INSTRUCTIONS:

Download REBOL/View from http://www.rebol.com/download-view.html (.6 MB)

Run REBOL, click the "Console" icon, and paste in any of these examples.

***** THAT'S EVERYTHING YOU NEED TO GET STARTED WITH THIS TEXT *****

To make REBOL automatically open to the console every time it starts, click the "User" link, and then uncheck "Open desktop at startup".

To use REBOL's built in text editor, type "editor none" at the console. Press the F5 key while in the editor to save and run any edited code. Be sure to type the header "REBOL []" at the top of any edited scripts. If you install REBOL, any file saved with a ".r" extension will run when its file icon is clicked (.r code files run just like .exe files).

Type "desktop" to reopen REBOL's default graphic desktop. See thousands of additional code examples by clicking the "REBOL" and "Public" icons.

See http://re-bol.com for a complete REBOL tutorial with line-by-line explanations of all the code in these examples. See the last example in this document for a useful quick reference manual.

Metapad (http://liquidninja.com/metapad/download.html) is a great editor for REBOL. Click Options -> Settings -> Primary External Viewer, and set your REBOL path (usually C:\Program Files\rebol\view\rebol.exe).

DEMO APPLICATION (10 apps in 2890 characters):

```
REBOL[title:"Demo"]p: :append kk: :pick r: :random y: :layout q: 'image
z: :if gg: :to-image v: :length? g: :view k: :center-face ts: :to-string
tu: :to-url sh: :show al: :alert rr: :request-date co: :copy g y[style h
btn 150 h"Paint"[g/new k y[s: area black 650x350 feel[engage: func[f a e][
z a = 'over[p pk: s/effect/draw e/offset sh s]z a = 'up[p pk 'line]]]
effect[draw[line]]b: btn"Save"[save/png %a.png gg s al"Saved 'a.png'"]btn
"Clear"[s/effect/draw: co[line]sh s]]]h"Game"[u: :reduce x: does[al join{
SCORE: }[v b]unview]s: gg y/tight[btn red 10x10]o: gg y/tight[btn tan
10x10]d: 0x10 w: 0 r/seed now b: u[q o(((r 19x19)* 10)+ 50x50)q s(((r
19x19)* 10)+ 50x50)]g/new k y/tight[c: area 305x305 effect[draw b]rate 15
feel[engage: func[f a e][z a = 'key[d: select u['up 0x-10 'down 0x10 'left
-10x0 'right 10x0]e/key]z a = 'time[z any[b/6/1 < 0 b/6/2 < 0 b/6/1 > 290
b/6/2 > 290][x]z find(at b 7)b/6[x]z within? b/6 b/3 10x10[p b u[q s(last
b)]w: 1 b/3:((r 29x29)* 10)]n: co/part b 5 p n(b/6 + d)for i 7(v b)1[
either(type?(kk b i)= pair!)[p n kk b(i - 3)][p n kk b i]]z w = 1[clear(
back tail n)p n(last b)w: 0]b: co n sh c]]]do[focus c]]]h"Puzzle"[al{
Arrange tiles alphabetically:}g/new k y[origin 0x0 space 0x0 across style
p button 60x60[z not find[0x60 60x0 0x-60 -60x0]face/offset - x/offset[
exit]tp: face/offset face/offset: x/offset x/offset: tp]p"O"p"N"p"M"p"L"
return p"K"p"J"p"I"p"H"return p"G"p"F"p"E"p"D"return p"C"p"B"p"A"x: p
white edge[size: 0]]]h"Calendar"[do bx:[z not(exists? %s)[write %s ""]rq:
rr g/new k y[h5 ts rq aa: area ts select to-block(find/last(to-block read
%s)rq)rq btn"Save"[write/append %s rejoin[rq" {"aa/text"} "]unview do bx]]
]]h"Video"[wl: tu request-text/title/default"URL:"join"http://tinyurl.com"
"/m54ltm"g/new k y[image load wl 640x480 rate 0 feel[engage: func[f a e][
z a = 'time[f/image: load wl show f]]]]h"IPs"[parse read tu join"http://"
"guitarz.org/ip.cgi"[thru<title>copy my to</title>]i: last parse my none
al ts rejoin["WAN: "i" -- LAN: "read join dns:// read dns://]]h"Email"[
g/new k y[mp: field"pop://user:pass@site.com"btn"Read"[ma: co[]foreach i
read tu mp/text[p ma join i"^/^/^/^/^/^/"editor ma]]]]h"Days"[g/new k y[
btn"Start"[sd: rr]btn"End"[ed: rr db/text: ts(ed - sd)show db]text{Days
Between:}db: field]]h"Sounds"[ps: func[sl][wait 0 rg: load sl wf: 1 sp:
open sound:// insert sp rg wait sp close sp wf: 0]wf: 0 change-dir
%/c/Windows/media do wl:[wv: co[]foreach i read %.[z %.wav = suffix? i[p
wv i]]]g/new k y[ft: text-list data wv[z wf <> 1[z error? try[ps value][al
"Error"close sp wf: 0]]]btn"Dir"[change-dir request-dir do wl ft/data: wv
sh ft]]]h{FTP}[g/new k y[px: field"ftp://user:pass@site.com/folder/"[
either dir? tu va: value[f/data: sort read tu va sh f][editor tu va]]f:
text-list[editor tu join px/text value]btn"?"[al{Type a URL path to browse
(nonexistent files are created). Click files to edit.}]]]]
```

Paste the code above into the REBOL interpreter to see how much computing can be accomplished with just a small amount of REBOL code.

LANGUAGE BASICS:

```
print "Hello world"
alert "Alert is a FUNCTION.  This text is its PARAMETER (or ARGUMENT)."
alert ("Parentheses can enclose parameters, but they aren't required.")
editor "This text is a parameter of the 'editor' function ... edit it."
browse http://rebol.com ; this function is 'browse' the parameter is a URL
request "Functions perform ACTIONS.  Parameters are DATA.  Make sense?"
; this is a comment
{
   This is a multi line comment.
   Comments don't do anything in a program.
   They just remind the programmer what's happening in the code.
}
request-text   ; These requester functions don't require any parameters,
request-date   ; but like most functions, they RETURN a useful value.
request-color
request-file
request-dir
request-pass

alert request-text  ; In all these examples, the 1st function uses a value
editor request-date ; RETURNED by the 2nd function, as its parameter.  To
print request-file  ; complete the 1st function, user response is needed.
editor request request-text         ; Here, return values are CASCADED.

alert "First function on this line" alert "Second function, same line"
alert "No"  alert "line"  alert "endings"  alert "are"  alert "required"
alert "functions are" editor "automatically grouped with their parameters"

request-text/default "Text"  ; many functions have "refinements" (options)
request-text/title  "The /title refinement sets this header text"
request-text/title/default  "Name:"  "John Smith"     ; 2 options together
request-text/title/offset "/offset repositions the requester" 10x100
request-pass/offset/title 10x100 "title" alert "Processing"  ; 2 functions
request-file/file %temp.txt          ; default file name
request-file/filter ["*.txt" "*.r"]  ; only show .txt and .r files
request-file/only  ; limit selection to a single file
request-file/save  ; save dialog (instead of open dialog)
request-file/save/file/filter %temp.txt ["*.txt" "*.r"]

print   10  +  12  /  2   ;  22 / 2 = 11  Math is ALWAYS evaluated left to
print  (10  +  12) /  2   ;  22 / 2 = 11  right.  The only exception:
print   10  + (12  /  2)  ;  10 + 6 = 16  Parentheses are evaluated first.

print "This is a string of text."
print {
    Curly braces are used for
    multi line text strings
    (instead of quotes).
}
alert {To use "quotes" in a text string, put them inside curly braces.}
alert "You can use {curly braces} inside quotes."
alert "'Single quotes' can go inside double quotes..."
alert {'...or inside curly braces'}
alert {"ANY quote symbol" {can actually be used within} 'curly braces'}
alert "In many cases"  alert {curly braces and quotes are interchangable.}

rejoin ["Hello " "World"]  ; "rejoin" CONCATENATES (joins together) values
rejoin [{Concatenate } {as } {many items } {as } {you } {want.}]
rejoin [request-date {  } request-color {  } now/time {  } $29.99]
alert rejoin ["You chose: " request "Choose one:"] ; CASCADE return values
join {"Join" only concatenates TWO items } {("rejoin" is more powerful).}
print rejoin ["This text is followed by a carriage return." newline]
```

```
print "This text is also followed by a carriage return.^/"
prin {'Prin' } prin {doesn't } prin {print } print {a carriage return.}
prin newpage  ; clear console screen

print read http://rebol.com  ; "read" retrieves the data from many sources
editor http://rebol.com   ; the built in editor can also read many sources
print read %./   ; the % symbol is used for local files and folders
editor %./
write %temp.txt "test"   ; write takes TWO parameters (file name and data)
editor %temp.txt
editor request-file/only   ; "only" refinement limits choice to 1 file
write clipboard:// (read http://rebol.com)  ; 2nd parameter in parentheses
editor clipboard://
print read dns://msn.com   ; REBOL can read many built in protocols
print read nntp://public.teranews.com
write/binary %/c/bay.jpg (read/binary http://rebol.com/view/bay.jpg)
write/binary %tada.wav (read/binary %/c/windows/media/tada.wav)
write/binary %temp.dat (compress read http://rebol.com)  ; COMPRESS DATA
print decompress read/binary %temp.dat                   ; DECOMPRESS DATA
print read ftp://user:pass@website.com/name.txt       ; user/pass required
write ftp://user:pass@website.com/name.txt "text"     ; user/pass required
editor ftp://user:pass@website.com/name.txt  ; can save changes to server!
editor pop://user:pass@website.com    ; read all emails in this account
send user@website.com "Hello"   ; send email
send user@website.com (read %file.txt)   ; email the text from this file
send/attach user@website.com "My photos" [%pic1.jpg %pic2.jpg pic3.jpg]

call/show "notepad.exe c:\config.sys"   ; run OS shell command
view layout [image %pic1.jpg]           ; view an image
view layout [image request-file/only]   ; view a user selected image
insert s: open sound:// load request-file/only wait s close s ; play sound
insert s: open sound:// load %/c/windows/media/tada.wav wait s close s

rename %temp.txt %temp2.txt   ; change file name
write %temp.txt read %temp2.txt   ; copy file
write/append %temp2.txt ""   ; create file (or if it exists, do nothing)
delete %temp2.txt
change-dir %../
what-dir
list-dir
make-dir %./temp
print read %./

x: 10     ; The COLON symbol assigns a value to a word label (a
"variable")
print x
x: x + 1    ; increment variable by 1 (add 1 to the current value of x)
print x
y: "hello"  z: " world"
print rejoin [y z]
probe rejoin [y z]   ; PROBE shows RAW DATA (PRINT formats nice output)
print join y z
person: "john"   ; variables (word labels) ARE ** NOT ** CASE SENSITIVE
print person   print PERSON   print PeRsOn
value1: value2: value3: "yes "   ; here, all 3 variables are set to "yes "
print rejoin [value1 value2 value3]
name: ask "Enter your name:  "   ; "ask" gets text input from the user
print rejoin ["Hello " name]
filename: request-file/only
alert rejoin ["You chose " filename]
osfile: to-local-file filename   ; REBOL uses its own multiplatform syntax
to-rebol-file osfile  ; Convert from native OS file notation back to REBOL
the-url: http://website.com/subfolder
split-path the-url   ; "split-path" breaks any file or URL into 2 parts
```

```
some-text:  {
    This is a multi line string of text.
    Strings are a native data type, delineated by
    quotes or curly braces, which enclose text.
    REBOL has MANY other built in data types, all
    delineated by various characters and text formats.
    The "type" function returns a value's data type.
    Below are just a few more native data types:
}
type? some-text

an-integer: 3874904                         type? an-integer
a-decimal: 7348.39                          type? a-decimal
web-site: http://musiclessonz.com           type? web-site
email-address: user@website.com             type? email-address
the-file: %/c/myfile.txt                    type? the-file
money-amount: $343.56                       type? money-amount
color-tuple: 123.54.212                     type? color-tuple
a-character: #"z"                           type? a-character
a-word: 'asdf                               type? a-word
html-tag: <br>                              type? html-tag
binary-info:  #{ddeedd}                     type? binary-info
image: load http://rebol.com/view/bay.jpg   type? image
a-sound: load %/c/windows/media/tada.wav    a-sound/type

to-decimal an-integer     ; Convert values TO different types ("cast")
to-string web-site        ; now the web site URL is surrounded by quotes
form web-site             ; "form" also converts various values to string
form $29.99
alert form $29.99         ; the alert function REQUIRES a string parameter
alert $29.99              ; (this throws an error)
5 + 6                     ; you can perform math operations with integers
"5" + "6"                 ; (error) you can't perform math with strings
(to-integer "5") + (to-integer "6")     ; this eliminates the math problem
to-pair [12 43]           ; creates a coordinate pair
as-pair 12 43             ; a better way to create a coordinate pair
to-binary 123.54.212      ; convert a REBOL color value to hex color value
to-binary request-color   ; convert the color chosen by the user, to hex
to-tuple #{00CD00}        ; convert a hex color value to REBOL color value
form to-tuple #{00CD00}   ; covert the hex color value to a string

6:30am + 00:37:19      ; REBOL computes values appropriately for type
now
now + 0:0:59
now - 10
23x54 + 19x31
22x66 * 2
22x66 * 2x3
192.168.1.1 + 0.0.0.37
11.22.33.44 * 9        ; note that each IP segment value is limited to 255
0.250.0 / 2            ; an easy way to adjust color values
$29.99 * 5
x: 12  y: 33  q: 18  p: 7
(as-pair x y) + (as-pair q p)  ; very common in graphics apps using coords

random/seed now/time   ; always use this line to get real random values
random 50              ; a random number between 0 and 50
random 50x100          ; left side is limited to 50, right limited to 100
random 222.222.222     ; each segment is limited to #s between 0 and 222
random $500
random "asdfqwerty"    ; a random mix of the given characters
```

```
FLOW CONTROL (Conditional Evaluations and Loops):

if now/time > 12:00 [alert "It's after noon."]      ; if (true) [do this]
if not now/time > 12:00 [alert "It's morning."]
unless now/time => 12:00 [alert "It's morning."]
if not now/time = 12:00 [alert "It's not noon."]
if now/time <> 12:00 [alert "It's not noon."]

if error? try [0 / 0] [alert "Divide by 0 error"]   ; if error [do this]
if attempt [1 / 1] [alert "Successful"]          ; if no error [do this]
attempt [0 / 0]                                  ; ignore error

either now/time > 8:00am [                       ; same as if/else
    alert "It's time to get up!"
][
    alert "You can keep on sleeping."
]

favorite-day:  request-text/title "What's your favorite day of the week?"
switch/default favorite-day [
    "Monday"    [alert "Monday is the worst!  The work week begins..."]
    "Tuesday"   [alert "Tuesdays and Thursdays are both ok, I guess..."]
    "Wednesday" [alert "The hump day - the week is halfway over!"]
    "Thursday"  [alert "Tuesdays and Thursdays are both ok, I guess..."]
    "Friday"    [alert "Yay!  TGIF!"]
    "Saturday"  [alert "Of course, the weekend!"]
    "Sunday"    [alert "Of course, the weekend!"]
] [alert "You didn't type in the name of a day!"]

name: "john"
case [
    find name "a" [print {Your name contains the letter "a"}]
    find name "e" [print {Your name contains the letter "e"}]
    find name "i" [print {Your name contains the letter "i"}]
    find name "o" [print {Your name contains the letter "o"}]
    find name "u" [print {Your name contains the letter "u"}]
    true [print {Your name doesn't contain any vowels!}]
]

name: "brian"
found: false
case/all [            ; /all evaluates ALL conditions, without breaking
    find name "a" [print {Your name contains the letter "a"} found: true]
    find name "e" [print {Your name contains the letter "e"} found: true]
    find name "i" [print {Your name contains the letter "i"} found: true]
    find name "o" [print {Your name contains the letter "o"} found: true]
    find name "u" [print {Your name contains the letter "u"} found: true]
    found = false [print {Your name doesn't contain any vowels!}]
]

value1: true   value2: false
if ((value1 = true) and (value2 = true)) [
    print "both true"
]
if ((value1 = true) or (value2 = true)) [
    print "either one OR the other is true"
]

alarm-time: now/time + :00:10
forever [if now/time = alarm-time [alert "10 seconds have passed" break]]

while [now/date < 21-dec-2012] [print "Press [ESC] to quit"]
until [now/date >= 21-dec-2012] [print "It's now December 21, 2012"]
```

```
loop 50 [print "REBOL is great!"]

count: 0
loop 50 [
    count: count + 1
    print rejoin ["This is loop #: " count]
]

repeat count 50 [print rejoin ["This is loop #: " count]]

for counter 1 10 1 [print counter]
for counter 10 1 -1 [print counter]
for counter 10 100 10 [print counter]
for counter 1 5 .5 [print counter]
for timer 8:00 9:00 0:05 [print timer]
for dimes $0.00 $1.00 $0.10 [print dimes]
for date 1-dec-2005 25-jan-2006 8 [print date]
for alphabet #"a" #"z" 1 [prin alphabet]

folder: read %.
foreach file folder [print file]
```

```
SERIES ("blocks", data lists):

new-block: copy []  ; a new, empty block
some-names: ["John" "Bill" "Tom" "Mike"]   ; a list of text strings
more-names: copy some-names   ; a copy of the above list
probe more-names
same? more-names some-names   ; NOT the exact same list, but a copy
same-names: some-names   ; THESE labels now refer to the EXACT SAME list
probe same-names
same? same-names some-names ; change some-names and same-names CHANGES TOO
sortednames: sort copy some-names  ; "copy" keeps some-names from changing
sortednames: sort some-names  ; here, the some-names block has been sorted
print first sortednames  ; here are 3 different ways to pick the 1st item:
print sortednames/1
print pick sortednames 1
print second sortednames   ; 3 different ways to pick the 2nd item:
print sortednames/2
print pick sortednames 2
find some-names "John"
first find some-names "John"
find/last some-names "John"
select some-names "John"   ; use series like dictionaries
reverse sortednames
length? sortednames
head sortednames
next sortednames
back sortednames
last sortednames
tail sortednames
at sortednames 2
skip sortednames 1
extract sortednames 3  ; every third item
index? sortednames
insert (at sortednames 3) "Lee"
append sortednames "George"
remove sortednames
remove find sortednames "Mike"
change sortednames "Phil"
change third sortednames "Phil"
poke sortednames 3 "Phil"
copy/part sortednames 2
replace/all sortednames "Lee" "Al"
probe form sortednames
probe mold sortednames
join some-names sortednames
intersect sortednames more-names
difference sortednames more-names
exclude sortednames more-names
union sortednames more-names
unique sortednames
clear sortednames
empty? sortednames
probe same-names
probe more-names

index-num: length? more-names
print pick more-names index-num
print pick more-names (index-num - 1)
print pick more-names (random length? more-names)
index-num: ((index? (find more-names "Tom")) - 1)
print pick more-names index-num ; 4 ways to pick items at a variable index
print more-names/:index-num
print compose [more-names/(index-num)]
print reduce [more-names/(index-num)]
```

```
; "READ" reads data byte-for-byte from file, "LOAD" performs a CONVERSION
; "WRITE" writes series byte-for-byte to file, "SAVE" performs CONVERSION

save %names.txt more-names          ; series data saved to a text file
loaded-names: load %names.txt       ; use "load" to read it into a variable
write %names2.txt mold more-names   ; series saved but WITH SQUARE BRACKETS
loaded-names2: load %names2.txt     ; "load" also correctly loads that file
read-names: to-block read %names.txt  ; "read" requires "to-block" convert
read-names2: to-block read %names2.txt  ; block within a block
probe read-names2                       ; [["John" "Phil" "Tom" "Mike"]]
first read-names2                       ; ["John" "Phil" "Tom" "Mike"]

write/binary %compressed.dat compress mold more-names  ; compress and save
probe load decompress read/binary %compressed.dat  ; read and decompress

foreach name more-names [print name]    ; FOR EACH item in list [do this]
foreach month system/locale/months [print month]
foreach file (read %./) [print file]

items: ["car" "boat" "house"]
foreach item items [print rejoin ["I like my " item]]

count: 0
foreach item items [
    count: count + 1
    print rejoin ["^/Item #" count ": " item]
]

big-block: [
    [may june july]
    [
        [1 2 3]
        [
            [yes no]
            [monday tuesday friday]
        ]
    ]
]

; Indentation makes the block easier to read, but is not required:
big-block: [[may june july][[1 2 3][[yes no][monday tuesday friday]]]]

probe first big-block          ; 3 ways to get the first item in the block
probe big-block/1
probe pick big-block 1
probe second big-block        ; 3 ways to get the second item in the block
probe big-block/2
probe pick big-block 2
probe first second big-block     ; 1st block in the 2nd block in big-block
probe big-block/2/1
probe second second big-block    ; 2nd block in the 2nd block in big-block
probe big-block/2/2
probe first second second big-block
probe big-block/2/2/1
probe second second second big-block
probe big-block/2/2/2
probe big-block/2/2/2/1
probe big-block/2/2/2/2
probe big-block/2/2/2/3
```

```
users: [
    "John Smith" "123 Tomline Lane Forest Hills, NJ" "555-1234"
    "Paul Thompson" "234 Georgetown Pl. Peanut Grove, AL" "555-2345"
    "Jim Persee" "345 Pickles Pike Orange Grove, FL" "555-3456"
    "George Jones" "456 Topforge Court Mountain Creek, CO" ""
    "Tim Paulson" "" "555-5678"
]

probe extract users 3           ; the name column (every 3rd item)
probe extract/index users 3 2   ; address column (skip 3, starting on 2)
probe extract/index users 3 3   ; phone column (skip 3, starting on 3)

probe form users  ; convert entire block to a string

foreach [name address phone] users [  ; get groups of 3 consecutive items
    print rejoin [
        "^/Name:     " name
        "^/Address:  " address
        "^/Phone:    " phone
    ]
]

foreach name (extract users 3) [
    if find name "a" [
        print pick users ((index? find users name) + 2)
    ]
]     ; prints phone numbers for all names that contain the letter "a"

do [
    prin newpage
    field: to-integer ask {Field to search (1=name, 2=address, 3=phone): }
    search-text: ask {Text to search for: }
    foreach [name address phone] users [
        if find (pick reduce [name address phone] field) search-text [
            print rejoin [
                newline
                "Name:     " name    newline
                "Address:  " address newline
                "Phone:    " phone   newline
            ]
        ]
    ]
]   ; "do" allows us to run some interactive console code such as "ask"

forever [
    prin newpage
    count: 1
    sorted-names: sort extract users 3
    foreach name sorted-names [
        print rejoin [newline count " - " name]
        count: count + 1
    ]
    choice: to-integer ask "^/^/Selection: "
    unless choice > (length? sorted-names) [
        chosen: index? find users (pick sorted-names choice)
        print newline
        for i chosen (chosen + 2) 1 [print pick users i]
        continue: ask "^/^/[ENTER] to continue, 'Q' to quit..."
        if continue = "q" [prin newpage halt]
    ]
]

append users ["Joe Thomas" "" "555-321-7654"]  ; append to end of list
```

```
name: "Alex Sharp"  address: "937 Boll Rd"  phone: "555-294-2834"
repend users [name address phone]              ; append variable values

insert (at users 4) [
    "Tom Adams" "321 Way Lane Villageville, AZ" "555-987-6543"
]

remove (at users 4)  ; remove 1 item
remove/part (at users 4) 2  ; remove 2 items
change (at users 1) "Jonathan Smith"
remove (at users 1) insert (at users 1) "Jonathan Smith"
foreach item users [
    replace item "John Smith" "Jonathan Smith"
]

new-users: copy []
foreach [name address phone] users [
    append/only new-users reduce [name address phone]
] ; append/only inserts blocks as blocks, instead of as individual items
editor new-users

field: 2  sort/compare new-users func [a b] [(at a field) < (at b field)]
editor new-users                    ; sorted by the 2nd field (by address)

users: copy []
foreach block new-users [append users reduce block]     ; "flatten" block
editor users

copy/part users 3
copy/part (at users 4) 3
copy at tail users -3
copy/part (at users 7) 3
copy/part (find users "Jim Persee") -3
copy/part (skip (find users "Jim Persee") -6) 3
alert form (copy/part users 3)

chosen: request-list "Choose a person: " (extract users 3)
alert form reduce [copy/part find users chosen 3]
alert reform [copy/part find users chosen 3]
chosen: request-list "Choose an address: " (extract/index users 3 2)
alert reform [copy/part at (find users chosen) -1 3]

x: ["one" "two" "three" "four" "five" "six"]
move/to (find x "five") 2
print x               ; item position changed

x: ["asdf" "qwer" "zxcv" "uiop" "hjkl" "vbnm"]
y: head clear skip tail x -2
probe y               ; last 2 items removed

data: [
    1 2 3 [4 5 6]
    7 8 9 [0 9 8 7 6 5 4 3 2 1]
    3 4 5 [6 3 1 7 8 0]
]
counter: 1
foreach [col1 col2 col3 col4] data [
    print rejoin [
        "Row: " counter newline
        "Column1: " col1 newline
        "Column2: " col2 newline
        "Column3: " col3 newline
        "Column4 (sorted): " (sort col4) newline newline
    ]
```

```
    counter: counter + 1
]

foreach file read %. [
    if (suffix? file) = %.tester [
        rename file to-file (replace to-string file ".tester" ".test")
    ]
]   ; directories are just lists of files - all series operations work
list-dir

for i 1 length? pp: open pop://user@site.com 1 compose [
    ask find pp/(i) "Subject:"
]

foreach line reverse copy system/console/history [print line]

replace/all replace/all replace/all replace/all form now/precise trim {
    /} "" ":" "x" "-" "q" "." ""       ; useful unique string generator

some-items: ["item1" "item2" "item3" "item4"]
an-image: load http://rebol.com/view/bay.jpg
append some-items an-image  ; block now contains 4 strings and an image
save/all %some-items.txt some-items  ; save it all to a simple text file
some-items: load %some-items.txt     ; load it back and use it later
view layout [image fifth some-items]

photo1: load http://rebol.com/view/bay.jpg
photo2: load http://rebol.com/view/demos/palms.jpg
photo-block: reduce [photo1 photo2]
; or  photo-block: compose [(photo1) (photo2)]
foreach photo photo-block [view layout [image photo]]
```

```
STRINGS (series of characters):

the-string: "abcdefghijklmnopqrstuvwxyz"
copy/part the-string 7
copy at tail the-string -7
copy/part (at the-string 12) 7
copy/part (find the-string "m") -7
copy/part (skip (find the-string "t") -12) 7
the-string/7
pick the-string 7
seventh the-string
replace the-string "cde" "123"
change (at the-string 7) "7"
poke the-string 7 #"7"
poke the-string 7 (to-char "7")
print the-string
remove/part (at the-string 3) 15
print the-string
insert (at the-string 3) "cdefghijklmnopq"
print the-string
insert/dup head the-string "-+ " 3
print the-string
replace/all the-string "-+ "  " "
print the-string
trim the-string
print the-string
extract the-string 3
to-integer third the-string
to-char 99
to-string to-char 99
to-string now
to-string $2344.44
to-string to-char 99
to-string system/locale/months
form now
form $2344.44
form to-char 99
form system/locale/months
the-block: copy []
foreach item the-string [append the-block item]
probe the-block
```

```
GUI WINDOWS (graphic user interfaces):

; VIEW LAYOUT [block of widgets]   ; basic syntax
;
; view layout [
;     widget1 properties [BLOCK OF ACTIONS TO BE PERFORMED BY WIDGET1]
;     widget2 properties [BLOCK OF ACTIONS TO BE PERFORMED BY WIDGET2]
; ]

view layout [area btn "Click Me"]   ; text area and button with some text

view layout [
    area
    btn "Click Me" [alert "You can type in the square area."]
]

view center-face layout [          ; center the GUI window on the screen
    a: area                        ; label this text area "a"
    btn "Save" [                   ; do this when the button is clicked:
        write %temp.txt a/text     ; write the text in the area to a file
        alert "Saved"              ; notify the user
    ]
]

view layout [
    f1: field "123"                  ; Text fields ONLY hold string values.
    f2: field "http://rebol.com"
    btn "Show Data Types" [
        print type? f1/text          ; Field text must be CONVERTED to
        print type? f2/text          ; expected REBOL data types for use:
    ]
    btn "Multiply" [print (to-integer f1/text) * 2]    ; convert to number
    btn "Browse" [browse (to-url f2/text)]             ; convert to URL
    btn "Error" [print f1/text * 2]    ; you can't perform math on strings
]

; ALWAYS use "show" to update ANY screen changes, as in this text editor:

view layout [
    a: area 600x350 wrap           ; 600 pixels wide by 350 pixels tall
    across                         ; layout next widgets horizontally
    f: field 500 "filename.txt"    ; 500 pixels across, with default text
    btn "Load" [                   ; do the following lines when clicked:
        f/text: request-file/file f/text  ; set text to selected file name
        show f                     ; TEXT CHANGED so widget must be updated
        a/text: read to-file f/text ; set area text to data read from file
        show a                     ; TEXT CHANGED so widget must be updated
    ]
    btn "Save" [                   ; do the following lines when clicked:
        write (request-file/only/save/file f/text) a/text     ; write the
        alert "Saved"              ; text in the area widget, to a file
    ]                              ; selected by the user, notify when done
]

view gui: layout [                 ; label the entire GUI layout
    size 600x400                   ; set the window size
    backdrop white                 ; set the window background color
    field1: field 560
    field2: field 560
    area1: area 560x235 wrap
    btn "Save" [
        save-block: copy []
        append save-block field1/text
        append save-block field2/text
```

```
        append save-block area1/text
        save %save.txt save-block
        alert {SAVED -- Now try running this script again, and load
            the data back into the fields.}
    ]
    btn "Load" [
        save-block: load %save.txt
        field1/text: save-block/1
        field2/text: save-block/2
        area1/text: save-block/3
        show gui                    ; UPDATE ENTIRE GUI to show changes
    ]
]

view layout[
    h1 "Send:"  ; big bold text widget (h1, h2, h3, ...)
    btn "Server settings" [  ; save user's email settings in REBOL:
        system/schemes/default/host: request-text/title "SMTP Server:"
        system/schemes/pop/host:     request-text/title "POP Server:"
        system/schemes/default/user: request-text/title "SMTP User Name:"
        system/schemes/default/pass: request-text/title "SMTP Password:"
        system/user/email:  to-email request-text/title "Email Address:"
    ]
    a: field "user@website.com"
    s: field "Subject"
    b: area
    btn "Send" [attempt [
        send/subject (to-email a/text) b/text s/text
        alert "Sent"
    ]]  ; "attempt" handles potential errors sending mail
    h1 "Read:"
    f: field "pop://user:pass@site.com"
    btn "Read" [editor to-url f/text]
]

REBOL [title: "My Title"]  ; this appears in the title bar of a GUI window
view layout [
    h3 "SOME POSITIONING EXAMPLES:"
    across
    btn "side" btn "by" btn "side"
    return
    btn "on the next line"
    tab
    btn "over a bit"
    tab
    btn "over more"
    below
    btn 160 "underneath" btn 160 "one" btn 160 "another"
    at 359x256
    btn "at 359x256"
]

write/append %save.txt ""            ; A useful data management program
view center-face gui: layout [
    text "Part Number:"
    x: field 560
    text "Part Name:"
    y: field 560
    text "Description:"
    z: area wrap 560x235
    across
    btn "Save" [
        do-face d 1   ; do all the delete button's code
        save %save.txt repend file [x/text y/text z/text]
```

```
    ]
    btn "Load" [
        chosen: request-list "Select:" extract (file: load %save.txt) 3
        if chosen = none [return]
        i: index? find file chosen
        x/text: pick file i
        y/text: pick file (i + 1)
        z/text: pick file (i + 2)
        show gui
    ]
    d: btn "Delete" [
        if true = request "Sure?" [
            remove/part (find (file: load %save.txt) x/text) 3
            save %save.txt file
            alert "Done"
        ]
    ]
    btn "Clear" [x/text: copy "" y/text: copy "" z/text: copy "" show gui]
    do [focus x]
]

write/append %recipes.txt ""  ; The program above, altered to hold recipes
view center-face gui: layout [
    text "Recipe Name:"
    x: field 560
    text "Ingredients:"
    y: area wrap 560x135
    text "Preparation Instructions:"
    z: area wrap 560x135
    across
    btn "Save" [
        do-face d 1   ; do all the delete button's code
        save %recipes.txt repend file [x/text y/text z/text]
    ]
    btn "Load" [
        chosen: request-list "Select:" extract (file: load %recipes.txt) 3
        if chosen = none [return]
        i: index? find file chosen
        x/text: pick file i
        y/text: pick file (i + 1)
        z/text: pick file (i + 2)
        show gui
    ]
    d: btn "Delete" [
        if true = request "Sure?" [
            remove/part (find (file: load %recipes.txt) x/text) 3
            save %recipes.txt file
            alert "Done"
        ]
    ]
    btn "Clear" [x/text: copy "" y/text: copy "" z/text: copy "" show gui]
    do [focus x]
]

write/append %s ""           ; A very compact version of the above program
view center-face g: layout [
    h3 "Name:"   x: field   h3 "Info:"   z: area wrap   across
    btn "Save" [do-face d 1  save %s  repend f [x/text z/text]]
    btn "Load" [
        c: request-list" Select:" extract (f: load %s) 2
        if c = none [return]
        x/text: first find f c  z/text: select f x/text  show g
    ]
    btn "New" [x/text: copy ""  z/text: copy ""  show g  focus x]
```

```
    d: btn "Delete" [
        if true = request "Sure?" [
            remove/part (find (f: load %s) x/text) 2 save %s f alert "ok"
        ]
    ]
]

s: ftp://user:pass@website.com/public_html/s.txt  ; edit this account info
write/append s ""               ; an Internet version of the above program
view center-face g: layout [
    h3 "Name:"   x: field   h3 "Info:"   z: area wrap   across
    btn "Save" [do-face d 1  save s  repend f [x/text z/text]]
    btn "Load" [
        c: request-list" Select:" extract (f: load s) 2
        if c = none [return]
        x/text: first find f c  z/text: select f x/text  show g
    ]
    btn "New" [x/text: copy ""  z/text: copy ""  show g  focus x]
    d: btn "Delete" [
        if true = request "Sure?" [
            remove/part (find (f: load s) x/text) 2 save s f alert "ok"
        ]
    ]
]
; browse s

; The word "value" refers to data contained in a currently active widget:

view layout [
    text "Some widgets with values and size/color properties.  Try them:"
    button red "Click Me" [alert "You clicked the red button."]
    f: field 400 "Type some text here, then press the [Enter] key" [
        alert value                 ; SAME AS alert f/text
    ]
    t: text-list 400x300 "Select this line" "Then this one" "Now this" [
        alert value                 ; SAME AS alert t/text
    ]
    check yellow [alert "You clicked the yellow check box."]
    button "Quit" [quit]
]

print "GUI Output:^/"
view layout [
    h1 "Some More GUI Widgets:"
    box red 500x2
    drop-down 200 data system/locale/months [
        a/text: join "Month:  " value show a
    ]
    a: field
    slider 200x18 [bar1/data: value show bar1]
    bar1: progress
    scroller 200x16 [bar2/data: value show bar2]
    bar2: progress
    across
    toggle "Click here" "Click again" [print value]
    rotary "Click" "Again" "And Again" [print value]
    choice "Choose" "Item 1" "Item 2" "Item 3" [print value]
    return
    x: radio y: radio z: radio
    btn "Get Radio States" [print [x/data y/data z/data]]
    return
    led
    arrow
    below
```

```
    code "Code text"
    tt "Typewriter text"
    text "Little Text" font-size 8
    title "Centered title" 500
]

; List Widget:

y: read %.   c: 0   x: copy []
foreach i y [append/only x reduce [(c: c + 1) i (size? to-file i)]]
slider-pos: 0
view center-face layout [
    across space 0
    the-list: list 400x400 [
        across  space 0x0
        text 50 purple
        text 250 bold [editor read to-file face/text]
        text 100 red italic
        return box green 400x1
    ] supply [
        count: count + slider-pos
        if none? q: pick x count [face/text: none exit]
        face/text: pick q index
    ]
    scroller 16x400 [
        slider-pos: (length? x) * value
        show the-list
    ]
]

view layout [
    area 400x400 load http://rebol.com/view/bay.jpg effect [
        Fit Flip Emboss              ; you can fit images on most widgets
    ]
]

effects:  [                          ; and there are MANY more effects:
  invert contrast 40 colorize 0.0.200 gradcol 1x1 0.0.255 255.0.0 tint 100
  luma -80 multiply 80.0.200 grayscale emboss flip 0x1 flip 1x0 rotate 90
  reflect 1x1 blur sharpen aspect tile tile-view
]

view layout [area effect [gradient red blue]]  ; gradients are color fades
view layout [
    size 500x400
    backdrop effect [gradient 1x1 tan brown]
    box effect [gradient 123.23.56 254.0.12]
    box effect [gradient blue gold/2]
]

view layout [
    btn "Right/Left Click Me" [alert "left click"] [alert "right click"]
]

svv/vid-face/color: white
alert "New global background color is now white."

; The word "offset" refers to a widget's coordinate position.
; The word "style" builds a new widget with the specified style & actions:

view center-face layout [
    origin 0x0 space 0x0 across
    style piece button 60x60 [
        if not find [0x60 60x0 0x-60 -60x0] (face/offset - empty/offset) [
```

```
            return  ; exit from the widget action (don't do anything else)
        ]
        temp: face/offset
        face/offset: empty/offset
        empty/offset: temp
    ]
    piece "1"   piece "2"   piece "3"   piece "4" return
    piece "5"   piece "6"   piece "7"   piece "8" return
    piece "9"   piece "10"  piece "11"  piece "12" return
    piece "13"  piece "14"  piece "15"
    empty: piece 200.200.200 edge [size: 0]
]

view center-face layout [
    size 600x440
    h3 "Press the left or right arrow key"
    key keycode [left]  [alert "You pressed the LEFT arrow key"]
    key keycode [right] [alert "You pressed the RIGHT arrow key"]
    btn #"a" "Click Me or Press the 'a' Key" [alert "clicked or pressed"]
]

; Here's a little program to show all key codes:
insert-event-func func [f e] [if e/type = 'key [print mold e/key] e]
view layout [text "Type keys to see their character/keycode"]

view gui: layout [       ; how to refer to the main layout window:
    btn1: btn "Button 1"
    btn2: btn "Remove all widgets from window" [
        foreach item system/view/screen-face/pane/1/pane [
            remove find system/view/screen-face/pane/1/pane item
        ]
        show gui
    ]
]

; "Feel" and "Engage" together detect events:

view layout [
    text "Mouse me." feel [
        engage: func [face action event] [
            if action = 'up [print "You just released the mouse."]
        ]
    ]
]

print "Click anywhere in the window, then click the text."
view center-face layout [
    size 400x200
    box 400x200 feel [
        engage: func [f a e] [     ; f a e = face action event
            print rejoin ["Mouse " a " at " e/offset]
        ]
    ]
    origin
    text "Click me" [print "Text clicked"] [print "Text right-clicked"]
    box blue [print "Box clicked"]
]

movestyle: [                    ; generic click and drag code
    engage: func [f a e] [
        if a = 'down [
            initial-position: e/offset
            remove find f/parent-face/pane f
            append f/parent-face/pane f
```

```
        ]
        if find [over away] a [
            f/offset: f/offset + (e/offset - initial-position)
        ]
        show f
    ]
]
view layout/size [
    style moveable-object box 20x20 feel movestyle
    at random 600x400 moveable-object (random 255.255.255)
    at random 600x400 moveable-object (random 255.255.255)
    at random 600x400 moveable-object (random 255.255.255)
    at random 600x400 moveable-object (random 255.255.255)
    at random 600x400 moveable-object (random 255.255.255)
    text "This text and all the boxes are movable" feel movestyle
] 600x440

; The following box code creates a repeating, multitasking loop in a GUI.
; The "within" function checks for graphic collisions:

view center-face layout [
    size 400x400
    btn1: btn red
    at 175x175 btn2: btn green
    box 0x0 rate 0 feel [engage: func [f a e] [if a = 'time [
        btn1/offset: btn1/offset + 5x5
        show btn1
        if within? btn1/offset btn2/offset 1x1 [alert "Collision" unview]
    ]]]
]

; Here's another, faster performing way to do GUI loops and multitasking:

gui: view/new center-face layout [
    size 400x400
    btn1: btn red
    at 175x175 btn2: btn green
]
forever [
    wait .02
    btn1/offset: btn1/offset + 5x5  show gui  ;  show btn1
    if within? btn1/offset btn2/offset 1x1 [alert "Collision"]
    if btn1/offset/1 > 300 [alert "done" quit]
]

view center-face layout [                    ; follow all mouse movements
    size 600x440
    at 270x209 b: btn "Click Me!" feel [
        detect: func [f e] [
            if e/type = 'move [
                if (within? e/offset b/offset 59x22) [
                    b/offset: b/offset + ((random 50x50) - (random 50x50))
                    if not within? b/offset -59x-22 659x462 [
                        b/offset: 270x209
                    ]
                    show b
                ]
            ]
            e
        ]
    ]
]
```

```
; To trap other events (this example traps and responds to close events):

insert-event-func [
    either event/type = 'close [
        really: request "Really close the program?"
        if really = true [unview]
    ] [event]  ; always return other events
]
view center-face layout [size 600x400]

insert-event-func [                    ; this example traps resize events
    either event/type = 'resize [
        fs: t1/parent-face/size
        t1/offset: fs / 2x2
        t2/offset: t1/offset - 50x25
        t3/offset: t1/offset - 25x50
        show gui  none
    ] [event]
]
svv/vid-face/color: white
view/options gui: layout [
    across
    t1: text "50x50"
    t2: text "- 50x25"
    t3: text "- 25x50"
] [resize]

; Use "to-image" to create a SCREEN SHOT of any layout:

picture: to-image layout [
    page-to-read: field "http://rebol.com"
    btn "Display HTML"
]
save/png %layout.png picture     ; save the image to a file
browse %layout.png

flash "Waiting..."  wait 3  unview         ; a simple info screen
inform layout [btn "Click Me" [flash "Waiting..." wait 3 unview]]

; Embed files (images, sounds, etc.) in code:

system/options/binary-base: 64
editor picture: compress to-string read/binary to-file request-file/only
view layout [image load (to-binary decompress picture)]

logo-pic: load to-binary decompress #{
789C018A0375FC89504E470D0A1A0A0000000D4948445200000640000001808
020000008360CFB900000013744558745374536F667477617265005245424F4C2F56
6965778FD9167800000332494441547 89CD599217402310C86F7CE6227B1481C
1637874362B1382C1687C4A15168240A89C5A2B058ECDEBE47DFFA429276DCEE
10FDCD582F97267FD33F2D7CF47ABDCF32D1ED76E7F3F9ED76FB4EE0743A8D46
A3B6A683A80FFE540562381C1E8FC7144D12DBEDB6951C3B9D4E91648DC7E34C
41B925465D349C14A2CA230BA65EA729E27C3E37CCB43CB228905A3525B1DBED
9A4CED93851C7C193088A0667C0D0603FB5640BFDFB7F648C0D0836B1C41C22E
11D7EBF57038F074BFDF534429BE2693891B4626CE1C59BC7CB95CDC99EEF7FB
66B349F922D65A4B4A8DE0D0B547B9DD85212B6B4CB4D3E994B055FEE8943566
30134626BBDA64052C974BD757A637B1DA2E599959A05EE61F4032D62C55EFBC
6EED01878954188DC80AE714C07126D24F91BBBE6265A129B3D96C2A4085BB64
459FEBF51A1B2692E5A9FA17A428B562EBE595A1F29650AD5C6B9525FD4621E0
A95D73491606F9046C94101A06178B4518E19122023655DA184B03ECA15BE98E
6D9D302E536E8D2C96A5FF0061458FEE9EAA045958720EDCFC82CF145A9E2C7C
52BC6CF0503B8C2B2200DAACD24698A4B710361E6421930E05A85E9484BE51B3
0885AE9727CB22A5591981B73D1AC6A58D2ABD5892DF46C5993DCFF25BC8828E
14538AACEB3390A43C59D890213B5D2AA3D2AC3C59ABD54ACE2E85C29E36DE42
```

```
162B8C0AC47F0942B512972CCCF0D91170ED6594ECC130288549ED44744DE52C
771381C571D5AFEDB14B2E79CB022F13C834A056049EFCE35C2A7449877A2B00
2D872635082FEA2D267D8BC047AD910D3875CE9247078A826259FC8234F264E1
9FAD4AAC52015465D973193B3755B611B417FB562A0C66C77EF7001F5463FD83
2CF20F83B2B8E0C22DAE760FA556B32AAF87B86A18C18259CFAA3567C250C7C3
1AE72CD95350531BD93FAE3B6CEADB33188174FCBBD77B7B7A0841DAB6C3EBEE
F13DE8696B6455E222ADCE23F162ECF644064709A47AA8FD3632BFAD78EA5E92
D947500C3BB04CAD419F3D5B05580DC127118E3D2866CAFB8AC6CAFCEB68F895
56796455CF47AAD741F5B957D4D751245980BD569729B723D742A964558FFB4D
EAB6A440BF6ACE54157EB028F7A730B695BDF749D05EA9C1B612C4CF0F396EDC
8E943F5C020000000049454E44AE426082CAEBA2D78A030000
}  ; this example embedded image was created with the script above
view layout [image logo-pic]

; Here are all the main GUI words that you should get to know:

VID-STYLES--GUI-WIDGETS: [
    face blank-face IMAGE BACKDROP BACKTILE BOX BAR SENSOR KEY BASE-TEXT
    VTEXT TEXT BODY TXT BANNER VH1 VH2 VH3 VH4 LABEL VLAB LBL LAB TITLE
    H1 H2 H3 H4 H5 TT CODE BUTTON CHECK CHECK-MARK RADIO CHECK-LINE
    RADIO-LINE LED ARROW TOGGLE ROTARY CHOICE DROP-DOWN ICON FIELD INFO
    AREA SLIDER SCROLLER PROGRESS PANEL LIST TEXT-LIST ANIM BTN BTN-ENTER
    BTN-CANCEL BTN-HELP LOGO-BAR TOG
]
LAYOUT-WORDS: [
    return at space pad across below origin guide tab tabs indent style
    styles size backcolor backeffect do
]
STYLE-FACETS--ATTRIBUTES: [
    edge font para doc feel effect effects keycode rate colors texts help
    user-data with bold italic underline left center right top middle
    bottom plain of font-size font-name font-color wrap no-wrap as-is
    shadow frame bevel ibevel
]
SPECIAL-STYLE-FACETS: [
    ARROW: [up right down left]  ROTARY: data  CHOICE: data  DROP-DOWN:
    [data rows]  FIELD: hide  INFO: hide  AREA: hide  LIST: [supply map
    data]  TEXT-LIST: data  ANIM: [frames rate]
]
```

```
FUNCTIONS:

; ANY string or block of data can be treated like a function:

some-actions: [
    alert "Here is one action."
    print "Here's a second action."
    write %/c/anotheraction.txt "Here's a third action."
]
do some-actions

write clipboard:// {alert "This code was run from the clipboard"}
do read clipboard://     ; copy, paste and run ANY code this way

write %some-code.r {
    REBOL [] ; executable code saved to a file must begin with this header
    print rejoin [newpage "The code in %some-code.r just ran." newline]
}
do %some-code.r

write ftp://user:pass@site.com/public_html/some-code.r {
    REBOL []
    print "The code in http://site.com/some-code.r just ran."
}
do http://site.com/some-code.r

the-word: to-word request-list "choose a word:" (first system/words)
do rejoin ["help " the-word]         ; you can DO any rejoined text

cls: does [prin "^(1B)[J"]           ; same as "prin newpage"
cls     ; no "do" required when a word is defined with "does"

x: 10
change-x-globally: func [y z] [x: y + z]
change-x-globally 10 20
print x

x: 10
change-x-locally: func [y z /local x] [x: y + z]
change-x-locally 10 20
print x

compute: func [x y /multiply /divide /subtract] [
    if multiply [return x * y]
    if divide   [return x / y]
    if subtract [return x - y]
    return x + y
]
compute/multiply 10 20
compute/divide 10 20
compute/subtract 10 20
compute 10 20

concatenate-string-or-num: func [
    "This function will only concatenate strings or integers."
    val1 [string! integer!] "First string or integer"
    val2 [string! integer!] "Second string or integer"
] [
    join val1 val2
]
help concatenate-string-or-num
concatenate-string-or-num "Hello " "there."  ; this works correctly
concatenate-string-or-num 10 20              ; this works correctly
concatenate-string-or-num 10.1 20.3          ; this creates an error
```

```
do [
    print "^/This example builds a line of code, and then executes it.^/"
    function: ask "Enter a function, such as 'print' or 'editor':  "
    parameter: ask "Enter a parameter, such as some random text:  "
    print rejoin [function { "} parameter {"}]
    do rejoin [function { "} parameter {"}]
    do compose [(to-word function) (parameter)]
    print "That's a very simple way to accomplish metaprogramming tasks."
]

write %imported-func.r {
    REBOL [title: "play-sound"]
    play-sound: func [sound-file] [
        wait 0
        insert sound-port: open sound:// load sound-file
        wait sound-port
        close sound-port
    ]
}
do %imported-func.r
play-sound %/C/WINDOWS/Media/chimes.wav
```

SEVERAL APPLICATION EXAMPLES:

```
REBOL [title: "Data Card File"]
write/append %data.txt ""   ; create the file if it doesn't exist
database: load %data.txt
view center-face gui: layout [
    text "Load an existing record:"
    name-list: text-list blue 400x100 data sort (extract database 4) [
        if value = none [return]
        marker: index? find database value
        n/text: pick database marker
        a/text: pick database (marker + 1)
        p/text: pick database (marker + 2)
        o/text: pick database (marker + 3)
        show gui
    ]
    text "Name:"       n: field 400
    text "Address:"    a: field 400
    text "Phone:"      p: field 400
    text "Notes:"      o: area  400x100
    across
    btn "Save" [
        if n/text = "" [alert "You must enter a name." return]
        if find (extract database 4) n/text [
            either true = request "Overwrite existing record?" [
               remove/part (find database n/text) 4
            ] [
               return
            ]
        ]
        save %data.txt repend database [n/text a/text p/text o/text]
        name-list/data: sort (extract copy database 4)
        show name-list
    ]
    btn "Delete" [
        if true = request rejoin ["Delete " n/text "?"] [
            remove/part (find database n/text) 4
            save %data.txt database
            do-face clear-button 1
            name-list/data: sort (extract copy database 4)
            show name-list
        ]
    ]
    clear-button: btn "New" [
        n/text: copy  ""
        a/text: copy  ""
        p/text: copy  ""
        o/text: copy  ""
        show gui
    ]
]
```

```
REBOL [title: "Calculator"]
prev-val: cur-val: 0   cur-eval: "+"   display-flag: false
print "0"
view center-face layout/tight [
    size 300x350 space 0x0 across
    display: field 300x50 font-size 28 "0" return
    style butn button 100x50  [
        if display-flag = true [display/text: "" display-flag: false]
        if display/text = "0" [display/text: ""]
        display/text: rejoin [display/text value]
        show display
```

```
        cur-val: display/text
    ]
    style eval button 100x50 brown font-size 13 [
        prev-val: cur-val
        display/text: "" show display
        cur-eval: value
    ]
    butn "1"  butn "2"  butn "3"  return
    butn "4"  butn "5"  butn "6"  return
    butn "7"  butn "8"  butn "9"  return
    butn "0"  butn "."  eval "+" return
    eval "-" eval "*" eval "/" return
    button 300x50 gray font-size 16 "=" [
        if display-flag <> true [
            if ((cur-eval = "/") and (cur-val = "0")) [
                alert "Division by 0 is not allowed." break
            ]
            prin rejoin [prev-val " " cur-eval " " cur-val " = "]
            print display/text: cur-val: do rejoin [
                prev-val " " cur-eval " " cur-val
            ]
            show display
            display-flag: true
        ]
    ]
]


REBOL [title: "Simple Search"]
phrase: request-text/title/default "Text to Find:" "the"
start-folder: request-dir/title "Folder to Start In:"
change-dir start-folder
found-list: ""
recurse: func [current-folder] [
    foreach item (read current-folder) [
        if not dir? item [  if error? try [
            if find (read to-file item) phrase [
                print rejoin [{"} phrase {" found in:  } what-dir item]
                found-list: rejoin [found-list newline what-dir item]
            ]] [print rejoin ["error reading " item]]
        ]
    ]
    foreach item (read current-folder) [
        if dir? item [
            change-dir item
            recurse %.\
            change-dir %..\
        ]
    ]
]
print rejoin [{SEARCHING for "} phrase {" in } start-folder "...^/"]
recurse %.\
print "^/DONE^/"
editor found-list
halt


REBOL [title: "Catch Game"]
alert "Arrow keys move left/right (up: faster, down: slower)"
random/seed now/time   speed: 11   score: 0
view center-face layout [
    size 600x440   backdrop white   across
    at 270x0 text "Score:"  t: text bold 100 (form score)
    at 280x20  y: btn 50x20 orange
```

```
    at 280x420 z: btn 50x20 blue
    key keycode [left] [z/offset: z/offset - 10x0  show z]
    key keycode [right]  [z/offset: z/offset + 10x0  show z]
    key keycode [up]  [speed: speed + 1]
    key keycode [down]  [if speed > 1 [speed: speed - 1]]
    box 0x0 rate 0 feel [engage: func [f a e] [if a = 'time [
        y/offset: y/offset + (as-pair 0 speed)  show y
        if y/offset/2 > 440 [
            y/offset: as-pair (random 550) 20   show y
            score: score - 1
        ]
        if within? z/offset (y/offset - 50x0) 100x20 [
            y/offset: as-pair (random 550) 20   show y
            score: score + 1
        ]
        t/text: (form score)  show t
    ]]]
]


REBOL [title: "Simple Data Retrieval App - Console"]
users: [
    ["John" "Smith" "123 Tomline Lane" "Forest Hills, NJ" "555-1234"]
    ["Paul" "Thompson" "234 Georgetown Pl." "Peanut Grove, AL" "555-2345"]
    ["Jim" "Persee" "345 Pickles Pike" "Orange Grove, FL" "555-3456"]
    ["George" "Jones" "456 Topforge Court" "Mountain Creek, CO" ""]
    ["Tim" "Paulson" "" "" "555-5678"]
]
a-line: copy [] loop 65 [append a-line "-"]
a-line: trim to-string a-line
print-all: does [
    foreach user users [
        print a-line
        print rejoin ["User:     " user/1 " " user/2]
        print a-line
        print rejoin ["Address:  " user/3 "  " user/4]
        print rejoin ["Phone:    " user/5]
        print newline
    ]
]
forever [
    prin "^(1B)[J"
    print "Here are the current users in the database:^/"
    print a-line
    foreach user users [prin rejoin [user/1 " " user/2 "  "]]
    print "" print a-line
    prin "Type the name of a user below "
    print "(part of a name will perform search):^/"
    print "Type 'all' for a complete database listing."
    print "Press [Enter] to quit.^/"
    answer: ask {What person would you like info about?  }
    print newline
    switch/default answer [
        "all"     [print-all]
        ""        [ask "Goodbye!  Press any key to end." quit]
        ][
        found: false
        foreach user users [
            if find rejoin [user/1 " " user/2] answer [
                print a-line
                print rejoin ["User:     " user/1 " " user/2]
                print a-line
                print rejoin ["Address:  " user/3 " " user/4]
                print rejoin ["Phone:    " user/5]
```

```
                print newline
                found: true
            ]
        ]
        if found <> true [
            print "That user is not in the database!^/"
        ]
    ]
    ask "Press [ENTER] to continue"
]


REBOL [title: "Simple Data Retrieval App - GUI"]
users: [
    ["John" "Smith" "123 Tomline Lane" "Forest Hills, NJ" "555-1234"]
    ["Paul" "Thompson" "234 Georgetown Pl." "Peanut Grove, AL" "555-2345"]
    ["Jim" "Persee" "345 Pickles Pike" "Orange Grove, FL" "555-3456"]
    ["George" "Jones" "456 Topforge Court" "Mountain Creek, CO" ""]
    ["Tim" "Paulson" "" "" "555-5678"]
]
user-list: copy []
foreach user users [append user-list user/1]
user-list: sort user-list
view display-gui: layout [
    h2 "Click a user name to display their information:"
    across
    list-users: text-list 200x400 data user-list [
        current-info: []
        foreach user users [
            if find user/1 value [
                current-info: rejoin [
                    "FIRST NAME:  " user/1 newline newline
                    "LAST NAME:   " user/2 newline newline
                    "ADDRESS:     " user/3 newline newline
                    "CITY/STATE:  " user/4 newline newline
                    "PHONE:       " user/5
                ]
            ]
        ]
        display/text: current-info
        show display show list-users
    ]
    display: area "" 300x400 wrap
]


REBOL [title: "Blogger"]
page: "blog.html"
ftp-url: ftp://user:pass@site.com/public_html/folder/ ; USER/PASS REQUIRED
html-url: join http://site.com/folder/ page
save/png %dot.png to-image layout/tight [box white 1x1]  ; blank image
view center-face gui: layout [
    h2 (form html-url)
    text "Title:"       t: field 400
    text "Link:"        l: field 400
    text "Image:"       i: btn 400 [i/text: request-file show i]
    text "Text:"        x: area  400x100
    across
    btn "Upload" [
        if error? try [existing-text: read html-url] [
            make-dir ftp-url
            write (join ftp-url page) ""
            existing-text: copy ""
        ]
```

```
        picture: last split-path to-file i/text
        write/binary (join ftp-url picture) (read/binary to-file i/text)
        write (join ftp-url page) rejoin [
            {<h1>} t/text {</h1>}
            {<img src="} picture {"><br><br>}
            now/date { } now/time {     }
            {<a href="} l/text {">} l/text {</a><br><br>}
            {<center><table width=80%><tr><td><pre><strong>}
                x/text
            {</strong></pre></td></tr></table></center><br><hr>}
            existing-text
        ]
        browse html-url
    ]
    btn "View" [browse html-url]
    btn "Edit" [editor (join ftp-url page)]
]


REBOL [title: "FTP Chat Room"]
webserver: to-url request-text/title/default {
    URL of text file on your server:} "ftp://user:pass@site.com/chat.txt"
name: request-text/title "Enter your name:"
cls: does [prin "^(1B)[J"]
write/append webserver rejoin [now ": " name " has entered the room.^/"]
forever [
    current-chat: read webserver
    cls
    print rejoin [
        "--------------------------------------------------"
        newline {You are logged in as: } name newline
        {Type "room" to switch chat rooms.} newline
        {Type "lock" to pause/lock your chat.} newline
        {Type "quit" to end your chat.} newline
        {Type "clear" to erase the current chat.} newline
        {Press [ENTER] to periodically update the display.} newline
        "--------------------------------------------------" newline
    ]
    print rejoin ["Here's the current chat text at: " webserver newline]
    print current-chat
    sent-message: copy rejoin [
        name " says: "
        entered-text: ask "You say:  "
    ]
    switch/default entered-text [
        "quit" [break]
        "clear" [
            if/else request-pass = ["secret" "password"] [
                write webserver ""
            ] [
                alert {
                    You must know the administrator password to clear
                    the room!
                }
            ]
        ]
        "room" [
            write/append webserver rejoin [
                now ": " name " has left the room." newline
            ]
            webserver: to-url request-text/title/default {
                New Web Server Address:} to-string webserver
            write/append webserver rejoin [
                now ": " name " has entered the room." newline
```

```
            ]
        ]
        "lock" [
            alert {The program will now pause for 5 seconds.
                You'll need the correct username and password
                to continue.
            }
            pause-time: now/time + 5
            forever [
                if now/time = pause-time [
                    while [
                        request-pass <> ["secret" "password"]
                    ][
                        alert "Incorrect password - look in the source!"
                    ]
                    break
                ]
            ]
        ]
    ][
        if entered-text <> "" [
            write/append webserver rejoin [sent-message newline]
        ]
    ]
]
cls print "Goodbye!"
write/append webserver rejoin [now ": " name " has closed chat." newline]
wait 1


REBOL [title: "Image Effector"]
effect-types: [
    "Invert" "Grayscale" "Emboss" "Blur" "Sharpen" "Flip 1x1" "Rotate 90"
    "Tint 83" "Contrast 66" "Luma 150" "None"
]
either exists? %/c/play_sound.r [
    do %/c/play_sound.r
    sound-available: true
][
    sound-available: false
]
image-url: to-url request-text/title/default {
    Enter the URL of an image to use:} trim {
    http://rebol.com/view/demos/palms.jpg}
gui: [
    across
    space -1
    at 20x2 choice 160 tan trim {
        Save Image} "View Saved Image" "Download New Image" trim {
            ------------} "Exit" [
        if value = "Save Image" [
            filename: to-file request-file/title/file/save trim {
                Save file as:} "Save" %/c/effectedimage.png
            wait 1 save/png filename to-image picture
        ]
        if value = "View Saved Image" [
            view-filename: to-file request-file/title/file {
                View file:} "" %/c/effectedimage.png
            view/new center-face layout [image load view-filename]
        ]
        if value = "Download New Image" [
            new-image: load to-url request-text/title/default trim {
                Enter a new image URL} trim {
                http://www.rebol.com/view/bay.jpg}
```

```
            picture/image: new-image
            show picture
        ]
        if value = "-------------" []  ; don't do anything
        if value = "Exit" [
            if sound-available = true [
                play-sound %/c/windows/media/tada.wav
            ]
            quit
        ]
    ]
    choice tan "Info" "About" [
        alert "Image Effector - Copyright 2005, Nick Antonaccio"
    ]
    below
    space 5
    pad 2
    box 550x1 white
    pad 10
    vh1 "Double click each effect in the list on the right:"
    return across
    picture: image load image-url
    text-list data effect-types [
        current-effect: value
        picture/effect: to-block form current-effect
        show picture
    ]
]
view/options center-face layout gui [no-title]


REBOL [Title: "Guitar Chord Diagram Maker"]
fretboard: load 64#{
iVBORw0KGgoAAAANSUhEUgAAAFUAAABkCAIAAAB4sesFAAAACXBIWXMAAAsTAAAL
EwEAmpwYAAAA2UlEQVR4nO3YQQqDQBAF0XTIwXtuNjfrLITs0rowGqbqbRWxEEL+
RFU9wJ53v8DN7Gezn81+NvvZXv3liLjmPX6n/4NL//72s9l/QGbWd5m53dbc8/kR
uv5RJ/QvzH42+9nsZ7OfzX42+nfOPzZzzyNUxxh8+qhfVHo94/rM49y+b/Wz2s9nP
Zj+b/WzuX/cvmfuXzX42+9nsZ7OfzX4296/7l8z9nsZ7OfzX4296/7l8z9y2Y/m/1s9rPZz2Y/m/vX/Uvm
/mWzn81+NvvZ7Gezn8396/4l2/n+y6N/f/vZ7Gezn81+tjenRWXD3TC8nAAAAABJ
RU5ErkJggg==
}
barimage: load 64#{
iVBORw0KGgoAAAANSUhEUgAAAEoAAAAFCAIAAABtvO2fAAAACXBIWXMAAAsTAAAL
EwEAmpwYAAAAHElEQVR4nGNsaGhgGL6AaaAdQFsw6r2hDIa59wCf/AGKgzU3RwAA
AABJRU5ErkJggg==
}
dot: load 64#{
iVBORw0KGgoAAAANSUhEUgAAAAoAAAAKCAIAAAACUFjqAAAACXBIWXMAAAsTAAAL
EwEAmpwYAAAAFElEQVR4nGNsaGhgwA2Y8HiNYGkA22EBlPG3fjQAAAAASUVORK5C
YII=
}
movestyle: [
    engage: func [f a e] [
        if a = 'down [
            initial-position: e/offset
            remove find f/parent-face/pane f
            append f/parent-face/pane f
        ]
        if find [over away] a [
            f/offset: f/offset + (e/offset - initial-position)
        ]
        show f
    ]
]
```

```
gui: [
    backdrop white
    currentfretboard: image fretboard 255x300
    currentbar: image barimage 240x15 feel movestyle
    text "INSTRUCTIONS:" underline
    text "Drag dots and other widgets onto the fretboard."
    across
    text "Resize the fretboard:"
    tab
    rotary "255x300" "170x200" "85x100" [
        currentfretboard/size: to-pair value show currentfretboard
        switch value [
            "255x300" [currentbar/size: 240x15 show currentbar]
            "170x200" [currentbar/size: 160x10 show currentbar]
            "85x100" [currentbar/size: 80x5 show currentbar]
        ]
    ]
    return
    button "Save Diagram" [
        filename: to-file request-file/save/file "1.png"
        save/png filename to-image currentfretboard
    ]
    tab
    button "Print" [
        filelist: sort request-file/title "Select image(s) to print:"
        html: copy "<html><body>"
        foreach file filelist [
            append html rejoin [
                {<img src="file:///} to-local-file file {">}
            ]
        ]
        append html [</body></html>]
        write %chords.html trim/auto html
        browse %chords.html
    ]
]
loop 50 [append gui [at 275x50 image dot 30x30 feel movestyle]]
loop 50 [append gui [at 275x100 image dot 20x20 feel movestyle]]
loop 50 [append gui [at 275x140 image dot 10x10 feel movestyle]]
loop 6 [append gui [at 273x165 text "X" bold feel movestyle]]
loop 6 [append gui [at 273x185 text "O" bold feel movestyle]]
view layout gui


REBOL [Title: "Thumbnail Maker"]
view center-face layout [
    text "Resize input images to this height:"
    height: field "200"
    text "Create output mosaic of this width:"
    width: field "600"
    text "Space between thumbnails:"
    padding-size: field "30"
    text "Color between thumbnails:"
    btn "Select color" [background-color: request-color/color white]
    text "Thumbnails will be displayed in this order:"
    the-images: area
    across
    btn "Select images" [
        some-images: request-file/title trim/lines {Hold
            down the [CTRL] key to select multiple images:} ""
        if some-images = none [return]
        foreach single-image some-images [
           append the-images/text single-image
           append the-images/text "^/"
```

```
        ]
        show the-images
    ]
    btn "Create Thumbnail Mosaic" [
        y-size: to-integer height/text
        mosaic-size: to-integer width/text
        padding: to-integer padding-size/text
        if error? try [background-color: to-tuple background-color][
            background-color: white
        ]
        images: copy parse/all the-images/text "^/"   ; see "PARSE" below
        if empty? images [alert "No images selected." break]
        mosaic: compose [
            backcolor (background-color) space (padding) across
        ]
        foreach picture images [
             flash rejoin ["Resizing " picture "..."]
             original: load to-file picture
             unview
             either original/size/2 > y-size [
                 new-x-factor: y-size / original/size/2
                 new-x-size: round original/size/1 * new-x-factor
                 new-image: to-image layout/tight [
                     image original as-pair new-x-size y-size
                 ]
                 append mosaic compose [image (new-image)]
             ][
                 append mosaic compose [image (original)]
             ]
             current-layout: layout/tight mosaic
             if current-layout/size/1 > mosaic-size [
                 insert back back tail mosaic 'return
             ]
        ]
        filename: to-file request-file/file/save "mosaic.png"
        save/png filename (to-image layout mosaic)
        view/new layout [image load filename]
    ]
]


REBOL [title: "Table Grid Example"]
x: copy [] for i 1 179 1 [append x reduce [i random "abcd"]]
update: does [q: copy [] foreach face g/pane/pane [append q face/text]]
do qq: [grid: copy [across space 0]
forskip x 2 [append grid compose [
    field (form x/1) 40 edge none
    field (form x/2) 260 edge [size: 1x1] return
]]
view center-face gui: layout [across space 0
    g: box 300x290 with [pane: layout/tight grid pane/offset: 0x0]
    slider 16x290 [
        g/pane/offset/y: g/size/y - g/pane/size/y * value show g
    ]
    return btn "Add" [
        row: (to-integer request-text/title "Insert at Row #:") * 2 - 1
        update insert at q row ["" ""] x: copy q unview do qq
    ]
    btn "Remove" [
        row: (to-integer request-text/title "Row # to delete:") * 2 - 1
        update remove/part (at q row) 2 x: copy q unview do qq
    ]
    btn "Col 1" [update editor extract q 2]
    btn "Col 2" [update editor extract/index q 2 2]
```

```
    btn "Save" [update save to-file request-file/save q]
    btn "Load" [x: load to-file request-file do qq]
    btn "History" [
        m: copy "ITEMS YOU'VE EDITED:^/^/" update for i 1 (length? q) 1 [
            if (to-string pick x i) <> (to-string pick q i) [
                append m rejoin [pick x i " " pick q i newline]
            ]
        ] editor m
    ]
]]
```

```
PARSE (goodbye regex):

my-text: {"apple","orange","pear"}
parsed-block: parse my-text none

code: "text1 <% replace this %> <% replace this %> text3"
parse/all code [
    any [thru "<%" copy new to "%>" (replace code new " text2 ")] to end
]
print code

some-text: {    First Name
           Last Name
           Street Address
           City, State, Zip}
parsed-block: parse/all some-text "^/"
foreach item parsed-block [trim item]
probe parsed-block

parse read http://guitarz.org/ip.cgi [
    thru <title> copy my-ip to </title>
]
parse my-ip [
    thru "Your IP Address is: " copy stripped-ip to end
]
alert to-string rejoin [
    "External: " trim/all stripped-ip "  "
    "Internal: " read join dns:// read dns://
]

html: read http://musiclessonz.com/ScreenShots.html
x: copy []
parse html [any [thru {src="} copy link to {"} (append x link)] to end]
make-dir %./downloaded-images
change-dir %./downloaded-images
foreach i x [attempt [
    print rejoin ["downloading:  " i]
    write/binary (to-file last split-path to-url i) (read/binary to-url i)
]]
foreach i read %. [view center-face layout [image (load i)]]

bb:  "some text http://guitarz.org http://yahoo.com"
bb_temp: copy bb
append bb_temp " "
append bb " "
parse bb [any [thru "http://" copy link to " " (
    replace bb_temp (rejoin [{http://} link]) (rejoin [
    {<a href="} {http://} link {" target=_blank>http://}
    link {</a>}]))] to end
]
bb: copy bb_temp
print bb

code: read to-file request-file
parse/all code [any [
    to #";" begin: thru newline ending: (
        remove/part begin ((index? ending) - (index? begin))) :begin
    ]
]
editor code  ; all comments removed

filename: %filename.csv
data: copy []
lines: read/lines filename
```

```
foreach line lines [
    append/only data parse/all line ","
]
info: copy ""
foreach line data [
    either find "Header" line/1 [
        info: line/1
    ][
        append line info
    ]
]
remove-each line data [find "Header" line/1/1]
remove-each line data [
    (line/3 = "TITLE") or (line/3 = "DESCRIPTION")
]
```

```
DLLs, SO, and DYLIB LIBRARIES:

lib: load/library %kernel32.dll
play-sound: make routine! [
    return: [integer!] pitch [integer!] duration [integer!]
] lib "Beep"
for hertz 37 3987 50 [
    print rejoin ["The pitch is now " hertz " hertz."]
    play-sound hertz 50
]
free lib

lib: load/library %winmm.dll
mciExecute: make routine! [c [string!] return: [logic!]] lib "mciExecute"
if not exists? %test.avi [
    flash "Downloading test video..."
    write/binary %test.avi read/binary http://re-bol.com/test.avi
    unview
]
video: to-local-file %test.avi
mciExecute rejoin ["OPEN " video " TYPE AVIVIDEO ALIAS thevideo"]
mciExecute "PLAY thevideo WAIT"
mciExecute "CLOSE thevideo"
free lib
quit

do [
    lib: load/library %winmm.dll
    mciExecute: make routine! [
        command [string!]
        return: [logic!]
    ] lib "mciExecute"
    file: to-local-file to-file request-file/save/title/file "Save as:" {
        } %rebol-recording.wav
    mciExecute "open new type waveaudio alias buffer1 buffer 6"
    mciExecute "record buffer1"
    ask "RECORDING STARTED (press [ENTER] when done)...^/"
    mciExecute "stop buffer1"
    mciExecute join "save buffer1 " file
    free lib
    print "Recording complete.  Here's how it sounds:^/"
    insert snd: open sound:// load to-rebol-file file wait snd close snd
    print "DONE.^/"
]

do [
    if not exists? %AutoItDLL.dll [
        write/binary %AutoItDLL.dll
        read/binary http://musiclessonz.com/rebol_tutorial/AutoItDLL.dll
    ]
    lib: load/library %AutoItDLL.dll
    move-mouse: make routine! [
        return: [integer!] x [integer!] y [integer!] z [integer!]
    ] lib "AUTOIT_MouseMove"
    print "Press the [Enter] key, and your mouse will move to the top"
    ask "corner of your screen, and then down diagonally to 200x200:  "
    for position 0 200 5 [
        move-mouse position position 10
    ]
    free lib
    print "^/Done.^/"
    halt
]
```

```
avicap32.dll: load/library %avicap32.dll
user32.dll: load/library %user32.dll
find-window-by-class: make routine! [
    ClassName [string!] WindowName [integer!] return: [integer!]
] user32.dll "FindWindowA"
sendmessage: make routine! [
    hWnd [integer!] val1 [integer!] val2 [integer!] val3 [integer!]
    return: [integer!]
] user32.dll "SendMessageA"
sendmessage-file: make routine! [
    hWnd [integer!] val1 [integer!] val2 [integer!] val3 [string!]
    return: [integer!]
] user32.dll  "SendMessageA"
cap: make routine! [
    cap [string!] child-val1 [integer!] val2 [integer!] val3 [integer!]
    width [integer!] height [integer!] handle [integer!]
    val4 [integer!] return: [integer!]
] avicap32.dll "capCreateCaptureWindowA"
view/new center-face layout/tight [
    image 320x240
    across
    btn "Take Snapshot" [
        sendmessage cap-result 1085 0 0
        sendmessage-file cap-result 1049 0 "scrshot.bmp"
        browse %scrshot.bmp
    ]
    btn "Exit" [
        sendmessage cap-result 1205 0 0
        sendmessage cap-result 1035 0 0
        free user32.dll   quit
    ]
]
hwnd: find-window-by-class "REBOLWind" 0
cap-result: cap "cap" 1342177280 0 0 320 240 hwnd 0
sendmessage cap-result 1034 0 0
sendmessage cap-result 1077 1 0
sendmessage cap-result 1075 1 0
sendmessage cap-result 1074 1 0
sendmessage cap-result 1076 1 0
do-events

tt: "Your Title"
user32.dll: load/library %user32.dll
gf: make routine![return:[int]]user32.dll"GetFocus"
sc: make routine![hw[int]a[string!]return:[int]]user32.dll"SetWindowTextA"
so: :show show: func[face][so[face]hw: gf sc hw tt]
view layout [text 400x400 "No 'REBOL -' in the title bar!"]
```

CGI:

```
#! /home/your_user_path/public_html/rebol/rebol -cs
REBOL [title: "Photo Viewer"]
print {content-type: text/html^/}
print {<HTML><HEAD><TITLE>Photos</TITLE></HEAD><BODY>}
; print read %template_header.html
folder: read %.
count: 0
foreach file folder [
    foreach ext [".jpg" ".gif" ".png" ".bmp"] [
        if find file ext [
            print [<BR> <CENTER>]
            print rejoin [{<img src="} file {">}]
            print [</CENTER>]
            count: count + 1
        ]
    ]
]
print {<BR>}
print rejoin [{Total Images: } count]
print {</BODY></HTML>}
; print read %template_footer.html
quit

#! ./rebol -cs
REBOL [title: "Generic CGI Application, With HTML Form"]
print {content-type: text/html^/}
submitted: decode-cgi system/options/cgi/query-string
print {<HTML><HEAD><TITLE>Page title</TITLE></HEAD><BODY>}
either empty? submitted [
    print {
        <FORM ACTION="http://yourwebserver.com/this_rebol_script.cgi">
        <INPUT TYPE="TEXT" NAME="username" SIZE="25">
        <INPUT TYPE="SUBMIT" NAME="Submit" VALUE="Submit">
        </FORM>
        </BODY></HTML>
    }
] [
    print rejoin [{Hello } second submitted {!}]
    print {</BODY></HTML>}
]

#! ./rebol -cs
REBOL [title: "CGI Logic"]
print {content-type: text/html^/}
submitted: decode-cgi system/options/cgi/query-string
if submitted/2 = "option1" [
    print "Now Doing Code For Option 1"
    quit
]
if submitted/2 = "option2" [
    print "Now Doing Code For Option 2"
    quit
]
options: ["option1" "option2"]
print {
    <FORM ACTION="./options.cgi">
    <select NAME="option">}
    foreach o options [prin rejoin [{<option>} o]]
    print {</option> </select>
    <INPUT TYPE="SUBMIT" NAME="Submit" VALUE="Submit">
    </FORM>
    </BODY></HTML>
```

```
}
quit

#! ./rebol -cs
REBOL [title: "Generic GET and POST Form Handler"]
write %form.html {
    <FORM action="form.cgi">
        Name:<BR><INPUT type="TEXT" name="name"><BR><BR>
        Email:<BR><INPUT type="TEXT" name="email"><BR><BR>
        Message:<BR>
            <TEXTAREA cols="75" rows="5" name="message">
            </TEXTAREA><BR><BR>
        <INPUT type="SUBMIT" name="Submit" value="Submit">
    </FORM>
}
print {content-type: text/html^/}
switch system/options/cgi/request-method [
    "POST" [
        cgi-data: copy ""  cgi-buffer: copy ""
        while [positive? read-io system/ports/input cgi-buffer 16380] [
            append cgi-data cgi-buffer clear cgi-buffer
        ]
    ]
    "GET" [cgi-data: system/options/cgi/query-string]
]
submitted: decode-cgi cgi-data
print {
    <HTML><HEAD><TITLE>Your Form Has Been Submitted</TITLE></HEAD>
    <BODY><CENTER><TABLE border=0 cellPadding=10 width="80%"><TR><TD>
}
entry: rejoin [{[^/ "Time Stamp:" } {"} form (now + 3:00) {"^/}]
foreach [title value] submitted [
    entry: rejoin [entry { } {"} mold title {" } mold value {^/}]
]
append entry {]^/}
write/append %submitted_forms.txt entry
html: copy ""
foreach [title value] submitted [
    repend html [
        <TR><TD width=20%> mold title </TD><TD> mold value </TD></TR>
    ]
]
print rejoin [
    {
         <FONT size=5>Thank You! The following information has been
         submitted: </FONT><BR><BR>Time Stamp:
    }
    now + 3:00  {<BR><BR><TABLE border=1 cellPadding=10 width="100%">}
    html  {</TABLE><BR>}
    {
        You'll hear from us shortly!<BR><BR><CENTER><FONT size=1>
        Copyright © 2009 This Web Site.  All rights reserved.</FONT>
        </CENTER></TD></TR></CENTER></BODY></HTML>
    }
]
quit

#! ./rebol -cs
REBOL [title: "Copy All Files to New Web Server"]
print "content-type: text/html^/"
print [<HTML><HEAD><TITLE>"wgetter"</TITLE></HEAD><BODY>]
foreach file (read ftp://user:pass@site.com/public_html/path/) [
    print file
    print <BR>
```

```
    write/binary (to-file file)
        (read/binary (to-url (rejoin [http://site.com/path/ file])))
]
print [</BODY></HTML>] quit
```

DRAW:

```
view layout [
    box 400x400 black effect [
        draw [
            pen red
            line 0x400 400x50
            pen white
            box 100x20 300x380
            fill-pen green
            circle 250x250 100
            pen blue
            fill-pen orange
            line-width 5
            spline closed 3 20x20 200x70 150x200
            polygon 20x20 200x70 150x200 50x300
        ]
    ]
]

view layout [
    box 400x220 effect [
        draw [
            fill-pen 200.100.90
            polygon 20x40 200x20 380x40 200x80
            fill-pen 200.130.110
            polygon 20x40 200x80 200x200 20x100
            fill-pen 100.80.50
            polygon 200x80 380x40 380x100 200x200
        ]
        gradmul 180.180.210 60.60.90
    ]
]

view layout [
    scrn: box black 400x400 feel [
        engage: func [face action event] [
            if find [down over] action [
                append scrn/effect/draw event/offset
                show scrn
            ]
            if action = 'up [append scrn/effect/draw 'line]
        ]
    ] effect [draw [line]]
]

pos: 300x300
view layout [
    scrn: box pos black effect [
        draw [image logo.gif 0x0 300x0 300x300 0x300]
    ]
    btn "Animate" [
        for point 1 140 1 [
            scrn/effect/draw: copy reduce [
                'image logo.gif
                (pos - 300x300)
                (1x1 + (as-pair 300 point))
                (pos - (as-pair 1 point))
                (pos - 300x0)
            ]
            show scrn
        ]
        scrn/effect/draw: copy [
            image logo.gif 0x0 300x0 300x300 0x300
```

```
        ]
        show scrn
    ]
]
```

```
HELP:

what
help call
? call
? "request"
? system
? system/console/history
? system/options
? system/locale/months
? system/network/host-address
? system/view/VID
? svv
editor extract svv/vid-styles 2
editor svv/vid-words
editor svv/facet-words
svv/vid-styles/TEXT-LIST/words

source for
editor mold :for

echo %words.txt what echo off   ; "echo" saves console activity to a file
echo %help.txt
foreach line read/lines %words.txt [
    word: first to-block line
    print "_____________________________________________________________^/"
    print rejoin ["word:  " uppercase to-string word]  print ""
    do compose [help (to-word word)]
]
echo off
editor at read %help.txt 4
```

```
PORTS:

help open

write/append %temp.txt "1234"  ; this line is the same as:
temp: open %temp.txt
append temp "1234"
close temp

; See ALL the properties of a port using "probe":

temp: open %temp.txt
probe temp
close temp

temp: open %temp.txt
print temp/date
print temp/path
print temp/size
close temp

my-file: open %temp.txt
set-modes my-file [
    world-read: true
    world-write: true
    world-execute: true
]
close my-file

write %temp.txt ""
print read %temp.txt
temp: open %temp.txt
append temp "1234"
print temp/state/inBuffer
print read %temp.txt
update temp
print read %temp.txt
temp: head temp
insert temp "abcd"
print temp/state/inBuffer
print read %temp.txt
close temp
print read %temp.txt
append temp "1q2w3e4r"  ; (error)

print second read pop://user:pass/site.com ; requires download of ALL mail
my-email: open pop://user:pass/site.com
print second my-email         ; port version only downloads single message

the-sound: load %/c/windows/media/tada.wav
insert s: open sound:// the-sound wait s close s
```

```
NETWORK PORTS:

; server (run this script first):

port: first wait open/lines tcp://:55555
print join "Received: "first wait port

; client (run this script second, in a separate instance of REBOL):

port: open/lines tcp://localhost:55555
insert port ask "Send:  "


REBOL [title: "Network Text Messenger"]
view layout [ across
    q: btn "Serve"[focus g p: first wait open/lines tcp://:8 z: 1]text"OR"
    k: btn "Connect"[focus g p: open/lines rejoin[tcp:// i/text ":8"]z: 1]
    i: field form read join dns:// read dns://  return
    r: area rate 4 feel [engage: func [f a e][if a = 'time and value? 'z [
        if error? try [x: first wait p] [quit]
        r/text: rejoin [x "^/" r/text] show r
    ]]]  return
    g: field "Type message here [ENTER]" [insert p value  focus face]
]


REBOL [title: "Network Binary File Transfer"]
; server/receiver - run first:
if error? try [port: first wait open/binary/no-wait tcp://:8] [quit]
mark: find file: copy wait port #""
length: to-integer to-string copy/part file mark
while [length > length? remove/part file next mark] [append file port]
view layout [image load file]
; client/sender - run after server (change IP address if using on 2 pcs):
save/png %image.png to-image layout [box blue "I traveled through ports!"]
port: open/binary/no-wait tcp://127.0.0.1:8  ; adjust this IP address
insert file: read/binary %image.png join l: length? file #""
insert port file


REBOL [title: "VOIP Intercom"] do [write %ireceive.r {REBOL []
if error? try [port: first wait open/binary/no-wait tcp://:8] [quit]
wait 0  speakers: open sound://
forever [
    if error? try [mark: find wav: copy wait port #""] [quit]
    i: to-integer to-string copy/part wav mark
    while [i > length? remove/part wav next mark] [append wav port]
    insert speakers load to-binary decompress wav
]} launch %ireceive.r
lib: load/library %winmm.dll
mci: make routine! [c [string!] return: [logic!]] lib "mciExecute"
if (ip: ask "Connect to IP (none = localhost):  ") = "" [ip: "localhost"]
if error? try [port: open/binary/no-wait rejoin [tcp:// ip ":8"]] [quit]
mci "open new type waveaudio alias wav"
forever [
    mci "record wav"  wait 2  mci "save wav r"  mci "delete wav from 0"
    insert wav: compress to-string read/binary %r join l: length? wav #""
    if l > 4000 [insert port wav]  ; squelch (don't send) if too quiet
]]

REBOL [Title: "UDP Group Chat"]
net-in: open udp://:9905  ; This is UDP, so NO known IP addresses required
net-out: open/lines udp://255.255.255.255:9905
set-modes net-out [broadcast: on]
```

```
name: request-text/title "Your name:"
gui: view/new layout [
    a1: area wrap rejoin [name ", you are logged in."]
    f1: field
    k1: at 0x0 key #"^M" [
        if f1/text = "" [return]
        insert net-out rejoin [name ":  " f1/text]
    ]
]
forever [
    focus f1
    received: wait [net-in]
    if not viewed? gui [quit]
    insert (at a1/text 1) copy received  show a1
]
```

```
OBJECTS:

account: make object! [
    first-name: last-name: address: phone: email-address: none
]

account/phone: "555-1234"
account/address: "4321 Street Place Cityville, USA 54321"
get in account 'first-name
get in account 'address
? account
next first account
foreach item (next first account) [print item]
foreach item (next first account) [print get in account item]

count: 0
foreach item (next first account) [
    count: count + 1
    print rejoin ["Item " count ":   " item]
    print rejoin ["Value:    " (get in account item) newline]
]

user1: make account [
    first-name: "John"
    last-name: "Smith"
    address: "1234 Street Place  Cityville, USA 12345"
    email-address: "john@hisdomain.com"
]

complex-account: make object! [
    first-name:
    last-name:
    address:
    phone:
    none
    email-address: does [
        return to-email rejoin [
            first-name "_" last-name "@website.com"
        ]
    ]
    display: does [
        print ""
        print rejoin ["Name:     " first-name " " last-name]
        print rejoin ["Address:  " address]
        print rejoin ["Phone:    " phone]
        print rejoin ["Email:    " email-address]
        print ""
    ]
]

user1: make complex-account []
user2: make complex-account [
    first-name: "John"
    last-name: "Smith"
    phone:  "555-4321"
]
user3: make complex-account [
    first-name: "Bob"
    last-name: "Jones"
    address: "4321 Street Place Cityville, USA 54321"
    phone:  "555-1234"
    email-address: "bob@mysite.net"  ; default function overwritten
]
user1/display   user2/display   user3/display
```

```
REBOL [title "Object Game"]
hidden-prize: random 15x15
character: make object! [
    position: 0x0
    move: does [
        direction: ask "Move up, down, left, or right:  "
        switch/default direction [
            "up" [position: position + -1x0]
            "down" [position: position + 1x0]
            "left" [position: position + 0x-1]
            "right" [position: position + 0x1]
        ] [print newline print "THAT'S NOT A DIRECTION!"]
        if position = hidden-prize [
            print newline
            print "You found the hidden prize.  YOU WIN!"
            print newline
            halt
        ]
        print rejoin [
            newline
            "You moved character " movement " " direction
            ".  Character " movement " is now "
            hidden-prize - position
            " spaces away from the hidden prize.  "
            newline
        ]
    ]
]
character1: make character []
character2: make character [position: 3x3]
character3: make character [position: 6x6]
character4: make character [position: 9x9]
character5: make character [position: 12x12]
loop 20 [
    prin "^(1B)[J"
    movement: ask "Which character do you want to move (1-5)?  "
    if find ["1" "2" "3" "4" "5"] movement [
        do rejoin ["character" movement "/move"]
        print rejoin [
            newline
            "The position of each character is now:  "
            newline newline
            "CHARACTER ONE:   " character1/position newline
            "CHARACTER TWO:   " character2/position newline
            "CHARACTER THREE: " character3/position newline
            "CHARACTER FOUR:  " character4/position newline
            "CHARACTER FIVE:  " character5/position
        ]
        ask "^/Press the [Enter] key to continue."
    ]
]

money: make object! [
    var: 1234.56
    bank: does [
        print ""
        print rejoin ["Your bank account balance is:  $" var]
        print ""
    ]
]
place: make object! [
    var: "Wabash"
    bank: does [
        print ""
```

```
        print rejoin [
            "Your favorite place is on the bank of the:  " var]
        print ""
    ]
]

money/bank
place/bank
money/var
place/var

deposit: make money [
    view layout [
        button "Deposit $10" [
            var: var + 10
            bank
        ]
    ]
]
travel: make place [
    view layout [
        new-favorite: field 300 trim {
            Type a new favorite river here, and press [Enter]} [
            var: value
            bank
        ]
    ]
]
```

```
THIRD PARTY LIBRARIES:

REBOL [title: "PDF example"]
do http://www.colellachiara.com/soft/Misc/pdf-maker.r
write/binary %example.pdf layout-pdf [[textbox ["Hello PDF world!"]]]
call %example.pdf

REBOL [title: "PDF example 2"]
do http://www.colellachiara.com/soft/Misc/pdf-maker.r
write/binary %example.pdf layout-pdf compose/deep [
    [
        page size 215.9 279.4  ; American Letter Size
        textbox ["Here is page 1.  It just contains this text."]
    ]
    [
        textbox 55 55 90 100 [
            {Here's page 2.  This text box contains a starting
             XxY position and an XxY size.  Coordinates are in
             millimeters and extend from the BOTTOM LEFT of the
             page (this box extends from starting point 55x55
             and is 90 mm wide, 100 mm tall).
             All the following page sizes are the default ISO A4,
             or 211×297 mm.  That is SLIGHTLY SMALLER than the
             standard American Letter page size.}
        ]
    ]
    [
        textbox 0 200 200 50 [
            center font Helvetica 10.5
            {This is page 3.  The text inside this box is centered
             and formatted using Helvetica font, with a character
             size of 10.5 mm.}
        ]
    ]
    [
        apply rotation 20 translation 35 150 [
            textbox 4 4 200 20 [
                {This is page 4.  The textbox is rotated 20 degrees
                 and translated (moved over) 35x150 mm.  Graphics
                 and images can also be rotated and translated.}
            ]
        ]
    ]
    [
        textbox 5 200 200 40 [
            {Here's page 5.  It contains this textbox and several
             images.  The first image is placed at starting point
             50x150 and is 10mm wide by 2.4mm tall.  The second
             image is 2x bigger and rotated 90 degrees.  The last
             image is placed at starting point 100x150 and is
             streched to 10x its size.  Notice that this ENTIRE
             layout block has been evaluated with compose/deep to
             evaluate the images in the following parentheses.}
        ]
        image 50 150 10 2.4 (system/view/vid/image-stock/logo)
        image 50 100 20 4.8 rotated 90
            (system/view/vid/image-stock/logo)
        image 100 150 100 24 (system/view/vid/image-stock/logo)
    ]
    [
        textbox [
            {This page contains this textbox and several generated
             graphics:  a line, a colored and filled box with a
             colored edge, and a circle.}
```

```
        ]
        line width 3  20 20 100 50   ; starting and ending XxY positions
        solid box edge width 0.2 edge 44.235.77 150.0.136 100 67 26 16
        circle 75 50 40   ; starting point 75x50, radius 40mm
    ]
]
call %example.pdf
; see http://www.colellachiara.com/soft/Misc/pdf-maker-doc.pdf

REBOL [title: "REBOL/Flash example"]
do http://box.lebeda.ws/~hmm/rswf/rswf_latest.r
make-swf/save/html http://tinyurl.com/yhex2cf
browse %starfield1.html
; see http://box.lebeda.ws/~hmm/rswf/
; and http://re-bol.com/rebol.html#section-9.17

REBOL [title: "RebGUI Example"]
do http://re-bol.com/rebgui.r
display/close "Tab Panel" [
    main-screen: tab-panel data [
        "Spreadsheet" [
            x: sheet data [
                A1 32 A2 12 A3 "=a1 + a2" A4 "=1.06 * to decimal! a3"
            ]
            a: area
            reverse
            button -1 " Show Data " [x/save-data set-text a x/data]
            button -1 " Quit! " [if true = question "Quit?" [quit]]
        ]
        "Pie Chart" [
            pie-chart data ["VID" yellow 19 "RebGUI" red 81]
                tip "Pie Chart!"
        ]
        "Menu" [
            menu #LHW data [
                "File" [
                     "Open" [x/text: read to-file request-file show x]
                     "Save" [write to-file request-file/save x/text]
                 ]
                 "About" [
                     "Info" [alert "RebGUI is great!"]
                 ]
            ]
            return
            x: area #LHW "[CTRL-Z: Undo CTRL-Y: Redo CTRL-S: Spell Check]"
        ]
        "VID style"  [
            style -1 data [text bold "Back to Spreadsheet" [
                main-screen/select-tab 1
            ]]
        ]
        action [wait .2 face/color: 230.230.230]  "Text" [
            text "Tabs are a great way to maximize screen real estate."
        ]
        action [wait .2 set-focus z]  "Fields" [
            y: field
            z: field "Edit me"
        ]
    ]
] [question "Really Close?"]
do-events
```

```
REBOL [title: "RebGUI Table Grid Example"]
do load-thru http://re-bol.com/rebgui.r
alert {Default username/password is "user1/pass1"}
unless exists? %snappmx.txt [
    save %snappmx.txt [
        "user1" "pass1" "Bill Jones" "bill@site.com" "Bill LLC"
        "user2" "pass2" "John Smith" "john@mail.com" "John LLC"
    ]
]
database: load %snappmx.txt
login: request-password
found: false
foreach [userid password name email company] database [
    either (login/1 = userid) and (login/2 = password) [found: true] []
]
if found = false [alert "Incorrect Login." quit]
add-record: does [
    display/dialog "User Info" [
        text 20 "User:" f1: field return
        text 20 "Pass:" f2: field return
        text 20 "Name:" f3: field return
        text 20 "Email:" f4: field return
        text 20 "Company:" f5: field reverse
        button -1 #XY " Clear " [clear-fields]
        button -1 #XY " Add " [add-fields]
    ]
]
edit-record: does [
    display/dialog "User Info" [
        text 20 "User:" f1: field (form pick t/selected 1) return
        text 20 "Pass:" f2: field (form pick t/selected 2) return
        text 20 "Name:" f3: field (form pick t/selected 3) return
        text 20 "Email:" f4: field (form pick t/selected 4) return
        text 20 "Company:" f5: field (form pick t/selected 5) reverse
        button -1 #XY " Delete " [
            t/remove-row t/picked
            save %snappmx.txt t/data
            hide-popup
        ]
        button -1 #XY " Save " [
            t/remove-row t/picked
            add-fields
            save %snappmx.txt t/data
            hide-popup
        ]
    ]
]
add-fields: does [
    t/add-row reduce [
        copy f1/text copy f2/text copy f3/text copy f4/text copy f5/text
    ]
    save %snappmx.txt copy t/data
]
clear-fields: does [
    foreach item [f1 f2 f3 f4 f5] [do rejoin [{set-text } item {""}]]
]
table-size: system/view/screen-face/size/1 / ctx-rebgui/sizes/cell
display/maximize "Users" [
    t: table table-size #LWH options [
        "" left .0  "" left .0   ; don't show the first 2 fields
        "Name" center .33  "Email" center .34  "Company" center .34
    ] data database [edit-record]
    reverse
    button -1 #XY " Add " [add-record]
```

```
]
do-events

REBOL [title: "Cyphre Menu and Tab Panel Example"]
do load-thru http://re-bol.com/cyphre_menu_and_tab_panel.r
insert-event-func [
    either event/type = 'resize [
        mn/size/1: system/view/screen-face/pane/1/size/1
        my-tabs/size: system/view/screen-face/pane/1/size - 15x30
        show [mn my-tabs]  none
    ] [event]
]
view/options center-face layout  [
    across space 0x0 origin 0x0
    mn: menu with [
        size: 470x20
        data: compose/deep [
            " File " [
                "Open" # "Ctrl+O" [request-file]
                "Save" # "Ctrl+S" [request-file/save]
                bar
                "Exit" [quit]
            ]
            " Options " [
                "Preferences" sub [
                    "Colors" [alert form request-color]
                    "Settings" [request-text/title "Enter new setting:"]
                ]
                "About" [alert "Menu Widget by Cyphre"]
            ]
        ]
    ]
    below
    at 10x25 my-tabs: tab-panel data [
        "Fields"   [
            h1 "Tab Panel by Cyphre" field field area area btn "Ok"
         ]
        "Data List"  [
            t1: text-list 400x430 data system/locale/months [alert value]
        ]
    ]
] [resize]

REBOL [title: "Listview Example"]
evt-close: func [face event] [
    either event/type = 'close [
        inform layout [
            across
            Button "Save Changes" [
                backup-file: to-file rejoin ["backup_" now/date]
                write backup-file read %database.db
                save %database.db theview/data quit
            ]
            Button "Lose Changes" [quit]
            Button "CANCEL" [hide-popup]
        ] none ] [
        event
    ]
]
insert-event-func :evt-close
if not exists? %list-view.r [write %list-view.r read
    http://www.hmkdesign.dk/rebol/list-view/list-view.r
]
do %list-view.r
```

```
if not exists? %database.db [write %database.db {[][]}]
database: load %database.db
view center-face gui: layout [
    h3 {To enter data, double-click any row, and type directly
        into the listview.  Click column headers to sort:}
    theview: list-view 775x200 with [
        data-columns: [
            Student Teacher Day Time Phone Parent Age Payments
            Reschedule Notes
        ]
        data: copy database
        tri-state-sort: false
        editable?: true
    ]
    across
    button "add row" [theview/insert-row]
    button "remove row" [
        if (to-string request-list "Are you sure?"
                [yes no]) = "yes" [
            theview/remove-row
        ]
    ]
    button "filter data" [
        filter-text: request-text/title trim {
            Filter Text (leave blank to refresh all data):}
        if filter-text <> none [
            theview/filter-string: filter-text
            theview/update
        ]
    ]
    button "save db" [
        backup-file: to-file rejoin ["backup_" now/date]
        write backup-file read %database.db
        save %database.db theview/data
    ]
]

REBOL [title: "Another Menu Module Example"]
if not exists? %menu-system.r [write %menu-system.r (
        read http://www.rebol.org/library/scripts/menu-system.r)]
do %menu-system.r
menu-data: [
    file: item "File" menu [item "Open" item "Save" item "Quit"]
    edit: item "Edit" menu [
        item "Item 1"
        item "Item 2" <ctrl-q>
        ---
        item "Submenu..." menu [
            item "Submenu Item 1"
            item "Submenu Item 2"
            item "Submenu Item 3" menu [
                item "Sub-Submenu Item 1"
                item "Sub-Submenu Item 2"
            ]
        ]
        ---
        item "Item 3"
    ]
    icons: item "Icons" menu [
        item "Icon Item 1" icons [help.gif stop.gif]
        item "Icon Item 2" icons [info.gif exclamation.gif]
    ]
]
basic-style: [item style action [
```

```
    switch item/body/text [
        case "Open" [
            the-file: request-file
            alert rejoin ["You opened: " the-file]
        ]
        case "Save" [
            the-file: request-file/save
            alert rejoin ["You saved to: " the-file]
        ]
        case "Quit" [
            if (request/confirm "Really Quit?") = true [quit]
        ]
        case "Item 1" [alert "Item 1 selected"]
        case "Item 2" [alert "Item 2 selected"]
        case "Item 3" [alert "Item 3 selected"]
        case "Submenu Item 1" [alert "Submenu Item 1 selected"]
        case "Submenu Item 2" [alert "Submenu Item 2 selected"]
        case "Submenu Item 3" [alert "Submenu Item 3 selected"]
        case "Sub-Submenu Item 1" [alert "Sub-Submenu Item 1 selected"]
        case "Sub-Submenu Item 2" [alert "Sub-Submenu Item 2 selected"]
        case "Icon Item 1" [alert "Icon Item 1 selected"]
        case "Icon Item 2" [alert "Icon Item 2 selected"]
    ]
]]
evt-close: func [face event] [either event/type = 'close [quit] [event]]
insert-event-func :evt-close
view center-face layout [
    size 400x500
    at 2x2 menu-bar menu menu-data menu-style basic-style
]

REBOL [title: "r3D Example"]
do http://re-bol.com/r3d.r
Transx:   Transy:   Transz:   300.0
Lookatx:  Lookaty:  Lookatz:  100.0
do update: does [
    world: copy []
    append world reduce [
        reduce [cube-model (r3d-scale 100.0 150.0 125.0) red]
    ]
    camera: r3d-position-object
        reduce [Transx Transy Transz]
        reduce [Lookatx Lookaty Lookatz]
        [0.0 0.0 1.0]
    RenderTriangles: render world camera r3d-perspective 250.0 400x360
    probe RenderTriangles    ; This line can be removed
]
view layout [
    scrn: box 400x360 black effect [draw RenderTriangles]  ; basic draw
    across return
    slider 60x16 [Transx: (value * 600 - 300.0) update show scrn]
    slider 60x16 [Transy: (value * 600 - 300.0) update show scrn]
    slider 60x16 [Transz: (value * 600) update show scrn]
    slider 60x16 [Lookatx: (value * 400 - 200.0) update show scrn]
    slider 60x16 [Lookaty: (value * 400 - 200.0) update show scrn]
    slider 60x16 [Lookatz: (value * 200 ) update show scrn]
]

REBOL [title: "MYSQL Example"]
; First, start MySQL and create a database named "Contacts"
unless exists? %mysql-protocol.r [
    write %mysql-protocol.r read http://re-bol.com/mysql-protocol.r
]
do %mysql-protocol.r
```

```
db: open mysql://root:root@localhost/Contacts
insert db {create table Contacts (
    name            varchar(100),
    address         text,
    phone           varchar(12),
    birthday        date
)}
insert db {INSERT into Contacts VALUES
    ('John Doe', '1 Street Lane', '555-9876', '1967-10-10'),
    ('John Smith', '123 Toleen Lane', '555-1234', '1972-02-01'),
    ('Paul Thompson', '234 Georgetown Pl.', '555-2345', '1972-02-01'),
    ('Jim Persee', '345 Portman Pike', '555-3456', '1929-07-02'),
    ('George Jones', '456 Topforge Court', '', '1989-12-23'),
    ('Tim Paulson', '', '555-5678', '2001-05-16')
}
insert db "DELETE from Contacts WHERE birthday = '1967-10-10'"
insert db "SELECT * from Contacts"
results: copy db
probe results
close db
halt
; see http://softinnov.org/rebol/mysql-usage.html

REBOL [title: "MySQL GUI Example"]
do %mysql-protocol.r
results: read/custom mysql://root:root@localhost/Contacts [
    "SELECT * from Contacts"
]
view layout [
    text-list 100x400 data results [
        string: rejoin [
            "NAME:      " value/1 newline
            "ADDRESS:   " value/2 newline
            "PHONE:     " value/3 newline
            "BIRTHDAY:  " value/4
        ]
        view/new layout [
            area string
        ]
    ]
]

REBOL [title: "SQLITE Example"]
unless exists? %sqlite3.dll [
    write/binary %sqlite3.dll read/binary http://re-bol.com/sqlite3.dll
]
unless exists? %sqlite.r [
    write %sqlite.r read http://re-bol.com/sqlite.r
]
do %sqlite.r
db: connect/create %contacts.db
SQL "create table contacts (name, address, phone, birthday)"
SQL {insert into contacts values
    ('"John Doe"', '"1 Street Lane"', '"555-9876"', '"1967-10-10"')
}
data: [
    "John Smith" "123 Toleen Lane" "555-1234" "1972-02-01"
    "Paul Thompson" "234 Georgetown Pl." "555-2345" "1972-02-01"
    "Jim Persee" "345 Portman Pike" "555-3456" "1929-07-02"
    "George Jones" "456 Topforge Court" "" "1989-12-23"
    "Tim Paulson" "" "555-5678" "2001-05-16"
]
SQL "begin"
foreach [name address phone bd] data [
```

```
    SQL reduce [
        "insert into contacts values (?, ?, ?, ?)" name address phone bd
    ]
]
SQL "commit"
SQL ["DELETE from Contacts WHERE birthday = ?" "1967-10-10"]
results: SQL "select * from contacts"
probe results
disconnect db
halt
; see http://www.dobeash.com/sqlite.html

REBOL [title: "SQLITE GUI Example"]
do %sqlite.r
connect %contacts.db
results: SQL "select * from contacts"
view layout [
    text-list 100x400 data results [
        string: rejoin [
            "NAME:      " value/1 newline
            "ADDRESS:   " value/2 newline
            "PHONE:     " value/3 newline
            "BIRTHDAY:  " value/4
        ]
        view/new layout [
            area string
        ]
    ]
]
disconnect

REBOL [title: "Captcha Example"]
write/binary %Caliban.caf read/binary http://re-bol.com/Caliban.caf
do http://re-bol.com/captcha.r
captcha/set-fonts-path %./
captcha/level: 4
write/binary %captcha.png captcha/generate
write %captcha.txt captcha/text
view center-face layout [
    image (load %captcha.png)
    text "Enter the captcha text:"
    f1: field [
        either f1/text = (read %captcha.txt) [
            alert "Correct"
        ] [
            alert "Incorrect"
        ]
    ]
]

REBOL [title: "Windows Screen Capture Example"]
do http://www.rebol.org/download-a-script.r?script-name=capture-screen.r
the-image: ftp://user:pass@site.com/path/current.png
view center-face gui: layout [
    button 150 "Upload Screen Shot" [
        unview gui
        wait .2
        save/png the-image capture-screen
        view center-face gui
    ]
]
```

```
REBOL [title: "XML-RPC Example"]
xmlrpc-exec http://betty.userland.com/RPC2 [examples.getStateName 41]
defh: func [orig-call] [do orig-call]
xmlrpc-serve/default [] 'defh
; see http://earl.strain.at/space/rebXR+Users+Guide
```

PLUGIN:

```
<HTML><HEAD><TITLE>REBOL Plugin for IE</TITLE></HEAD><BODY><CENTER>
<OBJECT ID="REBOL" CLASSID="CLSID:9DDFB297-9ED8-421d-B2AC-372A0F36E6C5"
    CODEBASE="http://www.rebol.com/plugin/rebolb7.cab#Version=1,0,0,0"
    WIDTH="500" HEIGHT="400">
    <PARAM NAME="LaunchURL" VALUE="http://yoursite.com/test_plugin.r">
    <PARAM NAME="BgColor" VALUE="#FFFFFF">
    <PARAM NAME="Version" VALUE="#FFFFFF">
    <PARAM NAME="BgColor" VALUE="#FFFFFF">
</OBJECT>
</CENTER></BODY></HTML>

<HTML><HEAD><TITLE>Plugin Mozilla, IE, Chrome</TITLE></HEAD><BODY><CENTER>
<OBJECT ID="REBOL_IE" CLASSID="CLSID:9DDFB297-9ED8-421d-B2AC-372A0F36E6C5"
    CODEBASE="http://re-bol.com/rebolb7.cab#Version=1,0,0,0"
    WIDTH="500" HEIGHT="400" BORDER="1" ALT="REBOL/Plugin">
    <PARAM NAME="bgcolor" value="#ffffff">
    <PARAM NAME="version" value="1.2.0">
    <PARAM NAME="LaunchURL" value="http://re-bol.com/test_plugin.r">
    <embed name="REBOL_Moz" type="application/x-rebol-plugin-v1"
        WIDTH="500" HEIGHT="400" BORDER="1" ALT="REBOL/Plugin"
        bgcolor="#ffffff"
        version="1.2.0"
        LaunchURL="http://re-bol.com/test_plugin.r"
    >
    </embed>
</OBJECT>
<script language="javascript" type="text/javascript">
var plugin_name = "REBOL/Plugin for Mozilla";
function install_rebol_plugin_mozilla() {
    if (navigator.userAgent.toLowerCase().indexOf("msie") == -1) {
        if (InstallTrigger) {
           xpi={'REBOL/Plugin for Mozilla':'http://re-bol.com/mozb2.xpi'};
           InstallTrigger.install(xpi, installation_complete);
        }
    }
}
function installation_complete(url, status) {
    if (status == 0) location.reload();
}
function is_mozilla_plugin_installed() {
    if (window.navigator.plugins) {
        for (var i = 0; i < window.navigator.plugins.length; i++) {
            if (window.navigator.plugins[i].name == plugin_name)
            return true;
        }
    }
    return false;
}
if (!is_mozilla_plugin_installed()) install_rebol_plugin_mozilla();
</script>
</CENTER></BODY></HTML>
```

```
MULTITASKING:

print "Obtaining keyboard input without blocking looping program flow:"
p: open/binary/no-wait console://
q: open/binary/no-wait [scheme: 'console]
count: 0
forever [
    count: count + 1
    if not none? wait/all [q :00:00.01] [
        wait q
        qq: to string! copy q
        probe qq
        print ["^/loop count incremented to" count "while waiting^/"]
    ]
]

alert {
GUI multitasking instructions:

    1)  Assign a rate of 0 to a GUI widget.
    2)  Assign a "feel" detection to that widget, and put the actions
        you want performed simultaneously inside the block that gets
        evaluated every time a 'time event occurs.
    3)  Stop and start the evaluation of concurrently active portions
        of code by assigning a rate of "none" or 0, respectively, to
        the associated GUI widget(s).
}
webcam-url: http://209.165.153.2/axis-cgi/jpg/image.cgi
view layout [
    btn "Start Video" [
        webcam/rate: 0
        webcam/image: load webcam-url
        show webcam
    ]
    btn "Stop Video" [webcam/rate: none show webcam]
    return
    webcam: image load webcam-url 320x240 rate 0 feel [
        engage: func [f a e][
            if a = 'time [
                f/image: load webcam-url show f
            ]
        ]
    ]
    clock: field to-string now/time/precise rate 0 feel [
        engage: func [f a e][
            if a = 'time [
                f/text: to-string now/time/precise show f
            ]
        ]
    ]
    h3 "Notice the delay in the timer as each image loads"
]

alert {
    This examples achieves true multitasking by simply writing the code
    for one process into a separate file and running it in a separate
    REBOL interpreter process using the "launch" function:
}
write %async.r {
    REBOL []
    view layout [
        origin 10x10
        clock: field to-string now/time/precise rate 0 feel [
            engage: func [face action event][
```

```
                if action = 'time [
                    face/text: to-string now/time/precise show face
                ]
            ]
        ]
    ]
}
launch %async.r
webcam-url: http://209.165.153.2/axis-cgi/jpg/image.cgi
view center-face layout [
    btn "Start Video" [
        webcam/rate: 0
        webcam/image: load webcam-url
        show webcam
    ]
    btn "Stop Video" [webcam/rate: none show webcam]
    return
    webcam: image load webcam-url 320x240 rate 0 feel [
        engage: func [face action event][
            if action = 'time [
                face/image: load webcam-url show face
            ]
        ]
    ]
]
```

```
VARIOUS ADDITIONAL APPLICATION EXAMPLES:

REBOL [title: "Ski Game"]
tree:  load to-binary decompress 64#{
eJzt18sNwjAQBFDTBSVw5EQBnLjQE1XRngmBQEj8Wa/3M4oYOZKBKHkaWwTO1/sh
jDkNx3N6HI7LcOzCfnz/9v5cMnEai7lj4mokT9C7XczUsrhvGSku6RkgDIbHAEP0
2EiIMBdMDuaOWZCSL91bQvCsSY4MHE9umXz7ydVi3xgltYvEKboexzVSlpTa614d
NonpUauIv176dX0ZTRgJlVgzNl25A3gkGwld1bkrNFqqedQfEI02AU9PjDeMpac/
ShKeTXylROqCImlXRFd9zkQoh4tp+GpqlSTnLnum4HTEzK/gjpmTpDxSASlHFqYU
EE/8nddG9n+9LIm8t9OeIEra2JZWDRSG4VEioa0UFCZFqv/aMQh2Rf790EnGgcJU
SVAer0Bhcp7/epVJvkHzBHjPfz+XSe6BwryC5gmQno3mAY3tpba2KAAA
}
skier-left: load to-binary decompress 64#{
eJyN0U8og2EcB/DvNrz+E5fJZSmRf9Ej76h3Ne1AIspyMQflpJDFU/KO1cQmSnGa
A3PYkvInB3kvuyzlgJolh+fCRUq5iBvP8+5lTvKrX33ep+/zp9/b2Tthhl6zvGt5
W3nX8TYhS1//MOGnSjNEa/AUxd0UVQ3raL9IYbBvA2OBI9Q0DqB6fAujl08Yi97D
Hr3F5EQYSss2OrrWEFo5xB+VO5Vx/skvnxmQbDCFvxcjMJ/b0s6LAZXGA3O0ZtTt
pW3WbJmDeMC8a1gE9o3bTBFI9YvGhrOKSueyEQpu9ri60vQFXFqPMx1K+sNWrdOh
73Y/uMr85fKdcIrJ0z6vxSfsYV5KCU2JEPNIlD9dFZ65AfXwD+HsKdAZiiLdqtvt
Hh65E5ZklTGmDvWLgxxKkjAivwt7XxhJEvIsrCY8ikLs0Tj3yGeCKaQtdsX9fv3G
N1jCJdyv84lHJkNriiM7Li29OIDV0jcU8kuIHaiPLEDEsG9DQYxiQTi0A8sBpEvh
OT65GmBYH9Jx5nf8TFFUFf5ZX2hFdG1uAgAA
}
skier-right: load to-binary decompress 64#{
eJxz8s1jYgCDMiDWAGIJINYCYkYGFrD4D0YGOBBAMBn4++Yz6HjVMSgY1oP5gWdu
M/gHTmCwNutlKJ2616F03VUGp3XnGGo+/mGILVnMoFkwhaHm7GcGz4m7GbABFwST
eQWSNXMQbM+3DAwlULbmEgaWXih75QUGzvkQJstMBwbPRRA2L1D5yS8QNudioNQF
qNYPDExAZRCtDg78c6Fa7wZK3Ycq94OO3L1fAcLWigpctUsZzHTSj5Jd+l7NAKS6
3HnXk6jHSiBF7sUmxi7Gl9VAZrqVOxsZuTirg8TTS0qAQs5FIPF0BhYXFkgog/zg
7gJlq5SXpaWVF4O9lZKuXl6eVl4AZLIfKS82LzYuB2nlOFxWXl5ubA6ytm1KWU65
cXExkMl09lNNR3q5eTFQPYfHE7YT6cXlJgcYGI7cPMAOMtKhgcH9wE8FBuPycgOG
BoYKtl8ODL4gjccY2HSAfr4BVMvgAwyazwwsXSA7ORgY2BQYeH+Cw+sAKPo5wEHj
kQAO/GZwIIHDgc0AaxQSBAAFOXD7bgIAAA==
}
random/seed now
the-score: 0
board: reduce ['image 300x20 skier-right black]
for i 1 20 1 [
    pos: random 600x540
    pos: pos + 0x300
    append board reduce ['image pos tree black]
]
view center-face layout/tight [
    scrn: box white 600x440 effect [draw board] rate 0 feel [
        engage: func [f a e] [
            if a = 'key [
                if e/key = 'right [
                    board/2: board/2 + 5x0
                    board/3: skier-right
                ]
                if e/key = 'left [
                    board/2: board/2 - 5x0
                    board/3: skier-left
                ]
                show scrn
            ]
            if a = 'time [
                new-board: copy []
                foreach item board [
                    either all [
                        ((type? item) = pair!)
                        ((length? new-board) > 4)
                    ] [
                        append new-board (item - 0x5)
```

```
                    ] [
                        append new-board item
                    ]
                    coord: first back back (tail new-board)
                    if ((type? coord) = pair!) [
                        if ((second coord) < -60) [
                            remove back tail new-board
                            remove back tail new-board
                            remove back tail new-board
                            remove back tail new-board
                        ]
                    ]
                ]
                board: copy new-board
                if (length? new-board) < 84 [
                    column: random 600
                    pos: to-pair rejoin [column "x" 440]
                    append board reduce ['image pos tree black]
                ]
                collision-board: remove/part (copy board) 4
                foreach item collision-board [
                    if (type? item) = pair! [
                        if all [
                          ((item/1 - board/2/1) < 15)
                          ((item/1 - board/2/1) > -40)
                          ((board/2/2 - item/2) < 30)
                          ((board/2/2 - item/2) > 5)
                        ] [
                            alert "Ouch - you hit a tree!"
                            alert rejoin ["Final Score: " the-score]
                            quit
                        ]
                    ]
                ]
                the-score: the-score + 1
                score/text: to-string the-score
                show scrn
            ]
        ]
    ]
    origin across h2 "Score:"
    score: h2 bold "000000"
    do [focus scrn]
]


REBOL [title: "Chord Accompaniment Player"]
play: false
insert-event-func [  ; standard method to trap GUI close events:
    either event/type = 'close [
        if play = true [play: false close sound-port]
        really: request "Really close the program?"
        if really = true [quit]
    ][
        event
    ]
]
flash "Downloading chord data (3.5 megabytes)..."
do load-thru http://musiclessonz.com/rebol_tutorial/wave_data.r
unview
view center-face layout [
    across
    h2 "Chords:"
    tab
```

```
chords: area 392x300 trim {
    bm bm bm bm
    gb7 gb7 gb7 gb7
    a a a a
    e e e e
    g g g g
    d d d d
    em em em em
    gb7 gb7 gb7 gb7
    g g g g
    d d d d
    gb7 gb7 gb7 gb7
    bm bm bm bm
    g g g g
    d d d d
    em em em em
    gb7 gb7 gb7 gb7
}
return
h2 "Delay:"
tab
tempo: field 50 "0.35" text "(seconds)"
tabs 40 tab
btn "PLAY" [
    play: true
    the-tempo: to-decimal tempo/text
    sounds: to-block chords/text
    wait 0
    sound-port: open sound://
    forever [
        foreach sound sounds [
            if play = false [break]
            do rejoin ["insert sound-port " reduce [sound]]
            wait sound-port
            wait the-tempo
        ]
        if play = false [break]
    ]
]
btn "STOP" [
    play: false
    close sound-port
]
btn "Save" [save to-file request-file/save chords/text]
btn "Load" [chords/text: load read to-file request-file show chords]
btn "HELP" [
    alert {
        This program plays chord progressions.  Simply type in
        the names of the chords that you'd like played, with a
        space between each chord.  For silence, use the
        underscore ("_") character.  Set the tempo by entering a
        delay time (in fractions of second) to be paused between
        each chord.  Click the start button to play from the
        beginning, and the stop button to end.  Pressing start
        again always begins at the first chord in the
        progression.  The save and load buttons allow you to
        store to the hard drive any songs you've created.
        Chord types allowed are major triad (no chord symbol -
        just a root note), minor triad ("m"), dominant 7th
        ("7"), major 7th ("maj7"), minor 7th ("m7"), diminished
        7th ("dim7"), and half diminished 7th ("m7b5").
        *** ALL ROOT NOTES ARE LABELED WITH FLATS (NO SHARPS)
        F# = Gb, C# = Db, etc...
     }
```

```
    ]
]


REBOL [title: "List Widget Example"]
x: copy []   random/seed now/time   ; generate 5000 rows of random data:
repeat i 5000 [
    append/only x reduce [random "asdfqwertyiop" form random 1000 form i]
]  y: copy x
Alert help-txt: {Be sure to try the following features:  1) Resize the GUI
    window to see the list automatically adjust to fit  2) Click column
    headers to sort by field  3) Use the arrow keys and page-up/page-down
    keys to scroll  4) Use the Insert, Delete and "M" keys to add, remove
    and move rows (by default, at the currently highlighted row)  5) Click
    the small "r" header button in the top right corner to reset the list
    back to its original values  6) Click any individual data cell to edit
    the selected value.}
sort-column: func [field] [
    either sort-order: not sort-order [
        sort/compare x func [a b] [(at a field) > (at b field)]
    ] [
        sort/compare x func [a b] [(at a field) < (at b field)]
    ]
    show li
]
key-scroll: func [scroll-amount] [
    s-pos: s-pos + scroll-amount
    if s-pos > (length? x) [s-pos: length? x]
    if s-pos < 0 [s-pos: 0]
    sl/data: s-pos / (length? x)
    show li  show sl
]
resize-grid: func [percentage] [
    gui-size: system/view/screen-face/pane/1/size ; - 10x0
    list-size/1: list-size/1 * percentage
    list-size/2: gui-size/2 - 95
    t-size: round (list-size/1 / 3)
    sl-size: as-pair 16 list-size/2
    unview/only gui view/options center-face layout gui-block [resize]
]
resize-fit: does [
    gui-size: system/view/screen-face/pane/1/size
    resize-grid (gui-size/1 / list-size/1 - .1)
]
insert-event-func [either event/type = 'resize [resize-fit none] [event]]
gui-size: system/view/screen-face/size - 0x50
list-size: gui-size - 60x95
sl-size: as-pair 16 list-size/2
t-size: round (list-size/1 / 3)
s-pos: 0  sort-order: true  ovr-cnt: none  svv/vid-face/color: white
view/options center-face gui: layout gui-block: [
    size gui-size  across
    btn "Smaller" [resize-grid .75]
    btn "Bigger" [resize-grid 1.3333]
    btn "Fit" [resize-fit]
    btn #"^~" "Remove" [attempt [
        indx: to-integer request-text/title/default "Row to remove:"
            form to-integer ovr-cnt
        if indx = 0 [return]
        if true <> request rejoin ["Remove: " pick x indx "?"] [return]
        remove (at x indx)  show li
    ]]
    insert-btn: btn "Add" [attempt [
        indx: to-integer request-text/title/default "Add values at row #:"
```

```
            form to-integer ovr-cnt
        if indx = 0 [return]
        new-values: reduce [
            request-text request-text (form ((length? x) + 1))
        ]
        insert/only (at x indx) new-values  show li
    ]]
    btn #"m" "Move" [
        old-indx: to-integer request-text/title/default "Move from row #:"
            form to-integer ovr-cnt
        new-indx: to-integer request-text/title "Move to row #:"
        if ((new-indx = 0) or (old-indx = 0)) [return]
        if true <> request rejoin ["Move: " pick x old-indx "?"] [return]
        move/to (at x old-indx) new-indx  show li
    ]
    btn "Save" [save to-file request-file/save x]
    btn "Load" [y: copy x: copy load request-file/only  show li]
    btn "Read Me" [alert help-txt]
    btn "View Data" [editor x]
    return  space 0x0
    style header button as-pair t-size 20 black white bold
    header "Random Text" [sort-column 1]
    header "Random Number" [sort-column 2]
    header "Unique Key" [sort-column 3]
    button black "r" 17x20 [if true = request "Reset?"[x: copy y show li]]
    return
    li: list list-size [
        style cell text t-size feel [
            over: func [f o] [
                if (o and (ovr-cnt <> f/data)) [ovr-cnt: f/data show li]
            ]
            engage: func [f a e] [
                if a = 'up [
                    f/text: request-text/default f/text show li
                ]
            ]
        ]
        across  space 0x0
        col1: cell blue
        col2: cell
        col3: cell red
    ] supply [
        either even? count [face/color: white] [face/color: 240.240.255]
        count: count + s-pos
        if none? q: pick x count [face/text: copy "" exit]
        if ovr-cnt = count [face/color: 200.200.255]
        face/data: count
        face/text: pick q index
    ]
    sl: scroller sl-size [s-pos: (length? x) * value  show li]
    key keycode [up] [key-scroll -1]
    key keycode [down] [key-scroll 1]
    key keycode [page-up] [key-scroll -20]
    key keycode [page-down] [key-scroll 20]
    key keycode [insert] [do-face insert-btn 1]
] [resize]


REBOL [title: "File-Explorer"]
closer: insert-event-func [either event/type = 'close [quit] [event]]
sort-column: func [field] [
    either sort-order: not sort-order [
        sort/compare x func [a b] [(at a field) > (at b field)]
    ] [
```

```
        sort/compare x func [a b] [(at a field) < (at b field)]
    ]
    show li
]
do-no-closer: func [the-code] [
    remove-event-func :closer
    do the-code
    closer: insert-event-func [either event/type = 'close [quit] [event]]
]
do file-request: [
    y: copy [] append y copy read %. insert head y "../"
    x: copy []  my-file: %none  sort-order: false
    foreach i y [
        append/only x reduce [
            i (size? to-file i) (modified? to-file i) (suffix? to-file i)
        ]
    ]
    slider-pos: 0
    view center-face layout [
        across  space 0x0  backdrop white
        h2 390 coal rejoin ["File: (" (length? y) - 1 ")"] [sort-column 1]
        h2 90 coal "Size:" [sort-column 2]
        h2 180 coal "Modified:" [sort-column 3]
        h2 right 50 coal "Type:" [sort-column 4] return
        li: list 750x360 [
            across  space 0x0
            text 390 bold [
                if face/text [
                    either #"/" = last (new-dir: copy face/text) [
                        unless %/ = clean-path to-file new-dir [
                            change-dir to-file new-dir
                            unview do file-request
                        ]
                    ] [
                        my-file: join what-dir to-file face/text unview
                    ]
                ]
            ]
            text 90 purple
            text 180 red italic
            text 50
            return box 0.0.240 750x1
        ] supply [
            count: count + slider-pos
            if none? q: pick x count [face/text: none exit]
            face/text: pick q index
        ]
        scroller 16x360 [
            lst-cnt: to-integer li/size/y / li/subface/size/y
            value: to-integer value * max 0 (length? y) - lst-cnt
            if slider-pos <> value [slider-pos: value show li]
        ]
        return
        h3 700 0.0.120 form what-dir  ; right
    ]
    ; probe my-file halt
    case [
        #"/" = (last my-file) []
        %.wav = (suffix? my-file) [
            if error? try [
                insert s: open sound:// load my-file wait s close s
            ] [close s alert "Incompatible wave file"]
        ]
        %.r = (suffix? my-file) [
```

```
            launch my-file
        ]
        find [%.html %.html] (suffix? my-file) [
            browse my-file
        ]
        find [%.jpg %.png %.gif %.bmp] (suffix? my-file) [
            do-no-closer [view center-face layout [image load my-file]]
        ]
        find [
            %.pdf %.exe %.com %.zip %.mp3 %.mid %.avi %.mov %.mpg %.mp4
            %.swf %.doc %.rtf %.xls %.ttf %.bat %.msi
        ] (suffix? my-file) [call my-file]
        find [%.dll %.bin %.sys] (suffix? my-file) [
            alert "This type of file should not be opened directly."
        ]
        true [do-no-closer [editor my-file]]
    ]
    do file-request
]


REBOL [title: "RebGUI Data Card File"]
do load-thru http://re-bol.com/rebgui.r
write/append %data.txt ""
database: load %data.txt
display "RebGUI Card File" [
    text 20 "Select:"
    names: drop-list #LW data (sort extract copy database 4) [
        marker: find database pick names/data names/picked
        set-text n copy first marker
        set-text a copy second marker
        set-text p copy third marker
        set-text o copy fourth marker
    ]
    after 2
    text 20 "Name:"        n: field #LW ""
    text 20 "Address:"     a: field #LW ""
    text 20 "Phone:"       p: field #LW ""
    after 1 text "Notes:"  o: area #LW ""
    after 3
    button -1 "Save" [
        if (n/text = "") [alert "You must enter a name." return]
        if find (sort extract copy database 4) copy n/text [
            either true = question "Overwrite existing record?" [
               remove/part (find database n/text) 4
            ] [return]
        ]
        database: repend database [
            copy n/text copy a/text copy p/text copy o/text
        ]
        save %data.txt database
        set-data names (sort extract copy database 4)
        set-text names copy n/text
    ]
    button -1 "Delete" [
        if true = question rejoin ["Delete " copy  n/text "?"] [
            remove/part (find database n/text) 4
            save %data.txt database
            set-data names (sort extract copy database 4)
            set-values face/parent-face ["" "" "" "" ""]
        ]
    ]
    button -1 "New" [
            set-values face/parent-face ["" "" "" "" ""]
```

```
    ]
]
do-events


REBOL [title: "RebGUI Text Editor"]
unless exists? %ui.dat [
    write %ui.dat read http://re-bol.com/ui-editor.dat
]
do load-thru http://re-bol.com/rebgui.r    ; Build#117
; do %rebgui.r
filename: %temp.txt
make-dir %./edit_history/
backup: does [
    if ((length? x/text) > 0) [
        write rejoin [
            %./edit_history/
             last split-path filename
             "_" now/date "_"
             replace/all form now/time ":" "-"
        ] x/text
    ]
]
ctx-rebgui/on-fkey/f5: does [
    backup
    write filename x/text
    launch filename
]
display/maximize/close "RebGUI Editor" [
    tight
    menu #LW data [
        "File" [
            "  New  " [
                if true = question "Erase Current Text?" [
                    backup
                    filename: %temp.txt set-text x copy ""
                ]
            ]
            "  Open  " [
                filetemp: to-file request-file/file filename
                if filetemp = %none [return]
                backup
                set-text x read filename: filetemp
            ]
            "  Save  " [
                backup
                write filename x/text
            ]
            "  Save As  " [
                filetemp: to-file request-file/save/file filename
                if filetemp = %none [return]
                backup
                write filename: filetemp x/text
            ]
            "  Save and Run  " [
                backup
                write filename x/text
                launch filename
            ]
            "  Print  "  [
                write %./edit_history/print-file.html rejoin [
                    {<}{pre}{>} x/text {<}{pre}{>}
                ]
                browse %./edit_history/print-file.html
```

```
            ]
            "  Quit  " [
                if true = question "Really Close?" [backup quit]
            ]
        ]
        "Options" [
            "  Appearance  " [request-ui]
        ]
        "Help" [
            "  Shortcut Keys  " [
                alert trim {
                    F5:       Save and Run
                    Ctrl+Z:   Undo
                    Ctrl+Y:   Redo
                    Esc:      Undo All
                    Ctrl+S:   Spellcheck
                }
            ]
        ]
    ] return
    x: area #LHW
] [
    if true = question "Really Close?" [backup quit]
]
do-events


REBOL [title: "Voice Alarms"]
lib: load/library %winmm.dll
mci: make routine! [c [string!] return: [logic!]] lib "mciExecute"
write %play-alarm.r {
    REBOL []
    wait 0
    the-sound: load %tmp.wav
    evnt: load %event.tmp
    if (evnt = []) [evnt: "Test"]
    forever [
        if error? try [
             insert s: open sound:// the-sound wait s close s
        ] [
             alert "Error playing sound!"
        ]
        delay: :00:07
        s: request/timeout [
            join uppercase evnt " alarm - repeats until you click 'stop':"
            "Continue"
            "STOP"
        ] delay
        if s = false [break]
    ]
}
current: rejoin [form now/date newline form now/time]
view center-face layout [
    c: box black 400x200 font-size 50 current rate :00:01 feel [
        engage: func [f a e] [
            if a = 'time [
                c/text: rejoin [form now/date newline form now/time]
                show c
                if error? try [
                    foreach evnt (to-block events/text) [
                        if any [
                             evnt/1 = form rejoin [
                                 now/date {/} now/time
                             ]
```

```
                        evnt/1 = form now/time
                    ] [
                        if error? try [
                            save %event.tmp form evnt/3
                            write/binary %tmp.wav
                            read/binary to-file evnt/2
                            launch %play-alarm.r
                        ] [
                            alert "Error playing sound!"
                        ]
                        ; request/timeout [(form evnt/3) "Ok"] :00:05
                    ]
                ]
            ] []  ; do nothing if user is manually editing events
        ]
    ]
]
h3 "Alarm Events (these CAN be edited manually):"
events: area  ; {[8:00:00am %alarm1.wav "Test Alarm - DELETE ME"]}
across
btn "Record Alarm Sound" [
    mci "open new type waveaudio alias wav"
    mci "record wav"
    request ["*** NOW RECORDING *** Click 'stop' to end:" "STOP"]
    mci "stop wav"
    if error? try [x: first request-file/file/save %alarm1.wav] [
        mci "close wav"
        return
    ]
    mci rejoin ["save wav " to-local-file x]
    mci "close wav"
    request [rejoin ["Here's how " form x " sounds..."] "Listen"]
    if error? try [
        save %event.tmp "test"
        write/binary %tmp.wav
        read/binary to-file x
        launch %play-alarm.r
    ] [
        alert "Error playing sound!"
    ]
]
btn "Add Event" [
    event-name: request-text/title/default "Event Title:" "Event 1"
    the-time: request-text/title/default "Enter a date/time:" rejoin [
        now/date {/} now/time
    ]
    if error? try [set-time: to-date the-time] [
        if error? try [set-time: to-time the-time] [
            alert "Not a valid time!"
            break
        ]
    ]
    my-sound: request-file/title/file ".WAV file:""" %alarm1.wav
    if my-sound = none [break]
    event-block: copy []
    append event-block form the-time
    append event-block my-sound
    append event-block event-name
    either events/text = "" [spacer: ""][spacer: newline]
    events/text: rejoin [events/text spacer (mold event-block)]
    show events
]
btn "Save Events" [
    write to-file request-file/file/save %alarm_events.txt events/text
```

```
    ]
    btn "Load Events" [
        if error? try [
            events/text: read to-file request-file/file %alarm_events.txt
        ] [return]
        show events
    ]
]


REBOL [title: "Little 3D Game"]
beep-sound: load to-binary decompress 64#{
eJwBUQKu/VJJRkZJAgAAV0FWRWZtdCAQAAAAAQABABErAAARKwAAAQAIAGRhdGEl
AgAA0d3f1cGadFQ+T2Z9jn1lSjM8T2uNsM/j7Midc05PWGh4eXVrXE5DQEZumsTn
4M2yk3hiVU9fcX+GcFU8KkNmj7rR3+HYroJbPUpfdoqAbldBP0ZWbpW62OvRrohk
WlleaHB2dW9bRzo1WYWy3OHbyrKObVNCVGp/jXpgRC48Vnievtfm6MCUaUVLWW1/
fXNkUkdCRlN7ps3r3cSkgm1fWFhmdH2AaVA6LElwnMja4dzNpHtXPUxje45/aVA5
PUtif6TG3uvMpHtXU1lkcnd2cGVURT0+ZJC84+HUvaGCZ1NIWm6AinVaQCtAX4Wu
yt3k37aJYEBKXXOHf3FdSEJET2KJsdPr1reUcGJbW2FsdXl2YUs5MFF7qdPe3tO+
mHNUP1Bnfo59ZEkyPFFukbTR5OvGm3BMTVlpent1aVpMQ0FJcZ3I6uHMsJB2YlZR
YXJ/hW5UOypEaJK90+Dg1qyBWjxKYHeLgG1WPz9HWXKYvNnr0KyFYVhZX2pydnVu
Wkc7N1yHtN3h2sivjGxTRFZrgI15X0MtPVh7osHZ5ua+kmdES1tvgn5zY1BGQ0hW
fqjO69vBoX9rXllaaHV9fmhPOi1Lcp/K2+DayaF4Vj1NY3uNfmhONjxLZIKnyODr
yqJ4VFFYZHN3dm5iUUM9QGaTv+Th0rqdf2VTS1tvgIl0WT4rQGCIssze5N60iF8/
Sl10h39vW0ZBRFFljLPU69W1kG1gWlxiYHkWb1ECAAA=
}
alert {
   Try to click the bouncing REBOLs as many times as possible in
   30 seconds.  The speed increases with each click!
}
do game: [
   speaker: open sound://
   g: 12 i: 5 h: i * g j: negate h  x: y: z: w: sc: 0  v2: v1: 1  o: now
   img1: to-image layout [backcolor brown box red center logo.gif]
   img2: to-image layout [backcolor aqua box yellow center logo.gif]
   img3: to-image layout [backcolor green box tan center logo.gif]
   cube: [[h h j][h h h][h j j][h j h][j h j][j h h][j j j][j j h]]
   view center-face layout/tight [
      f: box white 550x550 rate 15 feel [engage: func [f a e] [
         if a = 'time [
            b: copy []  x: x + 3  y: y + 3  ; z: z + 3
            repeat n 8 [
               if w > 500 [v1: 0]   if w < 50 [v1: 1]
               either v1 = 1 [w: w + 1] [w: w - 1]
               if j > (g * i * 1.4) [v2: 0]   if j < 1 [v2: 1]
               either v2 = 1 [h: h - 1] [h: h + 1]  j: negate h
               p: reduce pick cube n
               zx: p/1 * cosine z - (p/2 * sine z) - p/1
               zy: p/1 * sine z + (p/2 * cosine z) - p/2
               yx: (p/1 + zx * cosine y) - (p/3 * sine y) - p/1 - zx
               yz: (p/1 + zx * sine y) + (p/3 * cosine y) - p/3
               xy: (p/2 + zy * cosine x) - (p/3 + yz * sine x) - p/2 - zy
               append b as-pair (p/1 + yx + zx + w) (p/2 + zy + xy + w)
            ]
            f/effect: [draw [
               image img1 b/6 b/2 b/4 b/8
               image img2 b/6 b/5 b/1 b/2
               image img3 b/1 b/5 b/7 b/3
            ]]
            show f
            if now/time - o/time > :00:20 [
               close speaker
               either true = request [
                  join "Time's Up! Final Score: " sc "Again" "Quit"
```

```
            ] [do game] [quit]
        ]
    ]
    if a = 'down [
        xblock: copy [] yblock: copy []
        repeat n 8 [
            append xblock first pick b n
            append yblock second pick b n
        ]
        if all [
            e/offset/1 >= first minimum-of xblock
            e/offset/1 <= first maximum-of xblock
            e/offset/2 >= first minimum-of yblock
            e/offset/2 <= first maximum-of yblock
        ][
            insert speaker beep-sound wait speaker
            sc: sc + 1
            t1/text: join "Score: " sc
            show t1
            if (modulo sc 3) = 0 [f/rate: f/rate + 1]
            show f
        ]
    ]
  ]]
  at 200x0 t1: text brown "Click the bouncing REBOLs!"
]
]


Rebol [title: "Playing Card Framework - Freecell"]
flash "Downloading card images..."
do load-thru http://www.re-bol.com/playing-cards.r
unview
random/seed now
loop 156 [
    pos1: pick cards rnd1: (random 52) * 5
    pos2: pick cards rnd2: (random 52) * 5
    poke cards rnd1 pos2
    poke cards rnd2 pos1
]
movestyle: [
    engage: func [face action event] [
        if action = 'down [
            start-coord: face/offset
            face/data: event/offset
            remove find face/parent-face/pane face
            append face/parent-face/pane face
        ]
        if find [over away] action [
            unrounded-pos: (face/offset + event/offset - face/data)
            snap-to-x: (round/to first unrounded-pos 80) + 20
            snap-to-y: (round/to second unrounded-pos 20) + 20
            face/offset: (as-pair snap-to-x snap-to-y)
        ]
        if action = 'up [
            if any [
                (find cards face/offset)
                (face/offset/2 < 20)
            ] [
                if (face/offset/2 < 398) [face/offset: start-coord]
            ]
            replace cards start-coord face/offset
            arrange-cards
        ]
```

```
        show face
    ]
]
positions: does [
    temp: copy []
    foreach item cards [if ((type? item) = pair!) [append temp item]]
    return sort temp
]
arrange-cards: does [
    foreach position positions [
        foreach card system/view/screen-face/pane/1/pane [
            if (card/offset = position) and (position/2 < 398) [
                remove find system/view/screen-face/pane/1/pane card
                append system/view/screen-face/pane/1/pane card
            ]
        ]
    ]
    show system/view/screen-face/pane/1/pane
]
gui: [size 670x510 backdrop 0.150.0 across ]
foreach [card label num color pos] cards [
    append gui compose [
        at (pos) image load to-binary decompress (card) feel movestyle
    ]
]
box-pos: 18x398
loop 4 [
    append gui compose [
        at (box-pos) box green 72x2
        at (box-pos) box green 2x97
        at (box-pos + 320x0) box white 72x2
        at (box-pos + 320x0) box white 2x97
    ]
    box-pos: box-pos + 80x0
]
view/new center-face layout gui
arrange-cards
do-events


REBOL [title: "Number Verbalizer"]
verbalize: func [a-number] [
    if error? try [a-number: to-decimal a-number] [
        return "** Error **  Input must be a decimal value"
    ]
    if a-number = 0 [return "Zero"]
    the-original-number: round/down a-number
    pennies: a-number - the-original-number
    the-number: the-original-number
    if a-number < 1 [
        return join to-integer ((round/to pennies .01) * 100) "/100"
    ]
    small-numbers: [
        "One" "Two" "Three" "Four" "Five" "Six" "Seven" "Eight"
        "Nine" "Ten" "Eleven" "Twelve" "Thirteen" "Fourteen" "Fifteen"
        "Sixteen" "Seventeen" "Eighteen" "Nineteen"
    ]
    tens-block: [
        { } "Twenty" "Thirty" "Forty" "Fifty" "Sixty" "Seventy" "Eighty"
        "Ninety"
    ]
    big-numbers-block: ["Thousand" "Million" "Billion"]
    digit-groups: copy []
    for i 0 4 1 [
```

```
        append digit-groups (round/floor (mod the-number 1000))
        the-number: the-number / 1000
    ]
    spoken: copy ""
    for i 5 1 -1 [
        flag: false
        hundreds: (pick digit-groups i) / 100
        tens-units: mod (pick digit-groups i) 100
        if hundreds <> 0 [
            if none <> hundreds-portion: (pick small-numbers hundreds) [
                append spoken join hundreds-portion " Hundred "
            ]
            flag: true
        ]
        tens: tens-units / 10
        units: mod tens-units 10
        if tens >= 2 [
            append spoken (pick tens-block tens)
            if units <> 0 [
                if none <> last-portion: (pick small-numbers units) [
                    append spoken rejoin [" " last-portion " "]
                ]
                flag: true
            ]
        ]
        if tens-units <> 0 [
            if none <> tens-portion: (pick small-numbers tens-units) [
                append spoken join tens-portion " "
            ]
            flag: true
        ]
        if flag = true [
            commas: copy {}
            case [
                ((i = 4) and (the-original-number > 999999999)) [
                    commas: {billion, }
                ]
                ((i = 3) and (the-original-number > 999999)) [
                    commas: {million, }
                ]
                ((i = 2) and (the-original-number > 999)) [
                    commas: {thousand, }
                ]
            ]
            append spoken commas
        ]
    ]
    append spoken rejoin [
        "and " to-integer ((round/to pennies .01) * 100) "/100"
     ]
    return spoken
]
; HERE'S AN EXAMPLE OF HOW TO USE IT:
print verbalize ask "Enter a number to verbalize: "
halt


REBOL [Title: "Console Email"]
accounts: [
    ["pop.server" "smtp.server" "username" "password" you@site.com]
    ["pop.server2" "smtp.server2" "username" "password" you@site2.com]
    ["pop.server3" "smtp.server3" "username" "password" you@site3.com]
]
empty-lines: "^/"
```

```
loop 400 [append empty-lines "^/"]  ; # of lines it takes to clear screen
cls: does [prin {^(1B)[J}]
a-line:{--------------------------------------------------------------------}
select-account: does [
    cls
    print a-line
    forall accounts [
        print rejoin ["^/" index? accounts ":  " last first accounts]
    ]
    print join "^/" a-line
    selected: ask "^/Select an account #:  "
    if selected = "" [selected: 1]
    t: pick accounts (to-integer selected)
    system/schemes/pop/host:  t/1
    system/schemes/default/host: t/2
    system/schemes/default/user: t/3
    system/schemes/default/pass: t/4
    system/user/email: t/5
]
send-email: func [/reply] [
    cls
    print rejoin [a-line "^/^/Send Email:^/^/" a-line]
    either reply [
        print join "^/^/Reply-to:  " addr: form pretty/from
    ] [
        addr: ask "^/^/Recipient Email Address:  "
    ]
    either reply [
        print join "^/Subject:  " subject: join "re: " form pretty/subject
    ] [
        subject: ask "^/Email Subject:  "
    ]
    print {^/Body (when finished, type "end" on a seperate line):^/}
    print join a-line "^/"
    body: copy ""
    get-body: does [
        body-line: ask ""
        if body-line = "end" [return]
        body: rejoin [body "^/" body-line]
        get-body
    ]
    get-body
    if reply [
        rc: ask "^/Quote original email in your reply (Y/n)?  "
        if ((rc = "yes") or (rc = "y") or (rc = "")) [
            body: rejoin [
                body
                "^/^/^/--- Quoting " form pretty/from ":^/"
                form pretty/content
            ]
        ]
    ]
    print rejoin ["^/" a-line "^/^/Sending..."]
    send/subject to-email addr body subject
    cls
    print "Sent^/"
    wait 1
]
read-email: does [
    pretty: none
    cls
    print "One moment..."
    mail: open to-url join "pop://" system/user/email
    cls
```

```
    while [not tail? mail] [
        print "Reading...^/"
        pretty: import-email (copy first mail)
        either find pretty/subject "***SPAM***" [
            print join "Spam found in message #" length? mail
            mail: next mail
        ][
            print empty-lines
            cls
            prin rejoin [
                a-line
                {^/The following message is #} length? mail { from:  }
                system/user/email {^/} a-line {^/^/}
                {FROM:     } pretty/from {^/}
                {DATE:     } pretty/date {^/}
                {SUBJECT:  } pretty/subject {^/^/} a-line
            ]
            confirm: ask "^/^/Read Entire Message (Y/n):  "
            if ((confirm = "y") or (confirm = "yes") or (confirm = "")) [
                print join {^/^/} pretty/content
            ]
            print rejoin [
                {^/} a-line {^/}
                {^/[ENTER]:  Go Forward  (next email)^/}
                {^/    "b":  Go Backward (previous email)^/}
                {^/    "r":  Reply to current email^/}
                {^/    "d":  Delete current email^/}
                {^/    "q":  Quit this mail box^/}
                {^/  Any #:  Skip forward or backward this # of messages}
                {^/^/} a-line {^/}
            ]
            switch/default mail-command: ask "Enter Command:  " [
                ""  [mail: next mail]
                "b" [mail: back mail]
                "r" [send-email/reply]
                "d" [
                    remove mail
                    cls
                    print "Email deleted!^/"
                    wait 1
                ]
                "q" [
                    close mail
                    cls
                    print"Mail box closed^/"
                    wait 1
                    break
                ]
            ] [mail: skip mail to-integer mail-command]
            if (tail? mail) [mail: back mail]
        ]
    ]
]
select-account
forever [
    cls
    print a-line
    print rejoin [
        {^/"r":  Read Email^/}
        {^/"s":  Send Email^/}
        {^/"c":  Choose a different mail account^/}
        {^/"q":  Quit^/}
    ]
    print a-line
```

```
    response: ask "^/Select a menu choice:  "
    switch/default response [
        "r" [read-email]
        "s" [send-email]
        "c" [select-account]
        "q" [
            cls
            print "DONE!"
            wait .5
            quit
        ]
    ] [read-email]
]


REBOL [Title: "MP3 Jukebox"]
if not exists? %libwmp3.dll [
    write/binary %libwmp3.dll
    read/binary http://musiclessonz.com/rebol_tutorial/libwmp3.dll
]
lib: load/library %libwmp3.dll
Mp3_Initialize: make routine! [
    return: [integer!]
] lib "Mp3_Initialize"
Mp3_OpenFile: make routine! [
    return: [integer!]
    class [integer!]
    filename [string!]
    nWaveBufferLengthMs [integer!]
    nSeekFromStart [integer!]
    nFileSize [integer!]
] lib "Mp3_OpenFile"
Mp3_Play: make routine! [
    return: [integer!]
    initialized [integer!]
] lib "Mp3_Play"
Mp3_Stop: make routine! [
    return: [integer!]
    initialized [integer!]
] lib "Mp3_Stop"
Mp3_Destroy: make routine! [
    return: [integer!]
    initialized [integer!]
] lib "Mp3_Destroy"
Mp3_GetStatus: make routine! [
    return: [integer!]
    initialized [integer!]
    status [struct! []]
] lib "Mp3_GetStatus"
status: make struct! [
    fPlay [integer!]
    fPause [integer!]
    fStop [integer!]
    fEcho [integer!]
    nSfxMode [integer!]
    fExternalEQ [integer!]
    fInternalEQ [integer!]
    fVocalCut [integer!]
    fChannelMix [integer!]
    fFadeIn [integer!]
    fFadeOut [integer!]
    fInternalVolume [integer!]
    fLoop [integer!]
    fReverse [integer!]
```

```
] none
Mp3_Time: make struct! [
    ms [integer!]
    sec [integer!]
    bytes [integer!]
    frames [integer!]
    hms_hour [integer!]
    hms_minute [integer!]
    hms_second [integer!]
    hms_millisecond [integer!]
] none
TIME_FORMAT_SEC: 2
SONG_BEGIN: 1
SONG_CURRENT_FORWARD: 4
Mp3_Seek: make routine! [
    return: [integer!]
    initialized [integer!]
    fFormat [integer!]
    pTime [struct! []]
    nMoveMethod [integer!]
] lib "Mp3_Seek"
Mp3_PlayLoop: make routine! [
    return: [integer!]
    initialized [integer!]
    fFormatStartTime [integer!]
    pStartTime [struct! []]
    fFormatEndTime [integer!]
    pEndTime [struct! []]
    nNumOfRepeat [integer!]
] lib "Mp3_PlayLoop"
Mp3_GetSongLength: make routine! [
    return: [integer!]
    initialized [integer!]
    pLength [struct! []]
] lib "Mp3_GetSongLength"
Mp3_GetPosition: make routine! [
    return: [integer!]
    initialized [integer!]
    pTime [struct! []]
] lib "Mp3_GetPosition"
Mp3_SetVolume: make routine! [
    return: [integer!]
    initialized [integer!]
    nLeftVolume [integer!]
    nRightVolume [integer!]
] lib "Mp3_SetVolume"
Mp3_GetVolume: [
    initialized [integer!]
    pnLeftVolume [integer!]
    pnRightVolume [integer!]
    return: [integer!]
] lib "Mp3_GetVolume"
Mp3_VocalCut: make routine! [
    return: [integer!]
    initialized [integer!]
    fEnable [integer!]
] lib "Mp3_VocalCut"
Mp3_ReverseMode: make routine! [
    return: [integer!]
    initialized [integer!]
    fEnable [integer!]
] lib "Mp3_ReverseMode"
Mp3_Close: make routine! [
    return: [integer!]
```

```
    initialized [integer!]
] lib "Mp3_Close"
waves: []
foreach file read %. [
    if (%.mp3 = suffix? file) [append waves file]
]
append waves "(CHANGE FOLDER...)"
initialized: Mp3_Initialize
view center-face layout [
    vh2 "Click a File to Play:"
    file-list: text-list data waves [
        if value = "(CHANGE FOLDER...)" [
            new-dir: request-dir
            if new-dir = none [break]
            change-dir new-dir
            waves: copy []
            foreach file read %. [
                if (%.mp3 = suffix? file) [append waves file]
            ]
            append waves "(CHANGE FOLDER...)"
            file-list/data: waves
            show file-list
            break
        ]
        Mp3_GetStatus initialized status
        if (status/fPlay = 0) [
            file: rejoin [to-local-file what-dir "\" value]
            Mp3_OpenFile initialized file 1000 0 0
            Mp3_Play initialized
        ]
    ]
    across
    tabs 40
    text "Seek:   "
    tab slider 140x15 [
        plength: make struct! Mp3_Time compose [0 0 0 0 0 0 0 0]
        Mp3_GetSongLength initialized plength
        location: to-integer (value * plength/sec)
        ptime: make struct! Mp3_Time compose [0 (location) 0 0 0 0 0 0]
        Mp3_Seek initialized TIME_FORMAT_SEC ptime SONG_BEGIN
        Mp3_Play initialized
    ]
    return
    text "Volume: "
    tab slider 140x15 [
        volume: to-integer value * 100
        Mp3_SetVolume initialized volume volume
    ]
    return
    btn "Reverse" [
        Mp3_GetStatus initialized status
        either (status/fReverse > 0) [
            Mp3_ReverseMode initialized 0
        ] [
            Mp3_ReverseMode initialized 1
        ]
    ]
    btn "Vocal-Cut" [
        Mp3_GetStatus initialized status
        either (status/fVocalCut > 0) [
            Mp3_VocalCut initialized 0
        ] [
            Mp3_VocalCut initialized 1
        ]
```

```
    ]
    return
    tabs 50
    text "Loop Start:"
    tab start-slider: slider 120x15 []
    return
    text "Loop End:  "
    tab end-slider: slider 120x15 []
    return
    btn "Play Loop" [
        plength: make struct! Mp3_Time compose [0 0 0 0 0 0 0 0]
        Mp3_GetSongLength initialized plength
        s-loc: to-integer (start-slider/data * plength/sec)
        pStartTime: make struct! Mp3_Time compose [0 (s-loc) 0 0 0 0 0 0]
        end-loc: to-integer (end-slider/data * plength/sec)
        pEndTime: make struct! Mp3_Time compose [0 (end-loc) 0 0 0 0 0 0]
        ; TIME_FORMAT_SEC: 2
        Mp3_PlayLoop initialized 2 pStartTime 2 pEndTime 1000  ; 1000x
    ]
    btn 58 "Stop" [
        Mp3_GetStatus initialized status
        if (status/fPlay > 0) [Mp3_Stop initialized]
    ]
]
Mp3_Destroy initialized
free lib


REBOL [Title: "Textris"]
tui: func [commands [block!]] [
    string: copy ""
    cmd: func [s][join "^(1B)[" s]
    arg: parse commands [
        any [
            'clear (append string cmd "J") |
            'up    set arg integer! (append string cmd [
                arg "A"]) |
            'down  set arg integer! (append string cmd [
                arg "B"]) |
            'right set arg integer! (append string cmd [
                arg "C"]) |
            'left  set arg integer! (append string cmd [
                arg "D"]) |
            'at   set arg pair! (append string cmd [
                arg/x ";" arg/y "H" ]) |
            set arg string! (append string arg)
        ]
        end
    ]
    string
]
shape: [
    ["####"]
    ["#" down 1 left 1 "#" down 1 left 1 "#" down 1 left 1 "#"]
    ["###" down 1 left 2 "#"]
    [right 1 "#" down 1 left 2 "##" down 1 left 1 "#"]
    [right 1 "#" down 1 left 2 "###"]
    ["#" down 1 left 1 "##" down 1 left 2 "#"]
    ["###" down 1 left 3 "#"]
    ["##" down 1 left 1 "#" down 1 left 1 "#"]
    [right 2 "#" down 1 left 3 "###"]
    ["#" down 1 left 1 "#" down 1 left 1 "##"]
    ["###" down 1 left 1 "#"]
    [right 1 "#" down 1 left 1 "#" down 1 left 2 "##"]
```

```
    ["#" down 1 left 1 "###"]
    ["##" down 1 left 2 "#" down 1 left 1 "#"]
    ["##" down 1 left 1 "##"]
    [right 1 "#" down 1 left 2 "##" down 1 left 2 "#"]
    [right 1 "##" down 1 left 3 "##"]
    ["#" down 1 left 1 "##" down 1 left 1 "#"]
    ["##" down 1 left 2 "##"]
    ;
    ["    "]
    [" " down 1 left 1 " " down 1 left 1 " " down 1 left 1 " "]
    ["   " down 1 left 2 " "]
    [right 1 " " down 1 left 2 "  " down 1 left 1 " "]
    [right 1 " " down 1 left 2 "   "]
    [" " down 1 left 1 "  " down 1 left 2 " "]
    ["   " down 1 left 3 " "]
    ["  " down 1 left 1 " " down 1 left 1 " "]
    [right 2 " " down 1 left 3 "   "]
    [" " down 1 left 1 " " down 1 left 1 "  "]
    ["   " down 1 left 1 " "]
    [right 1 " " down 1 left 1 " " down 1 left 2 "  "]
    [" " down 1 left 1 "   "]
    ["  " down 1 left 2 " " down 1 left 1 " "]
    ["  " down 1 left 1 "  "]
    [right 1 " " down 1 left 2 "  " down 1 left 2 " "]
    [right 1 "  " down 1 left 3 "  "]
    [" " down 1 left 1 "  " down 1 left 1 " "]
    ["  " down 1 left 2 "  "]
]
floor:  [
    21x5 21x6 21x7 21x8 21x9 21x10 21x11 21x12 21x13 21x14 21x15
]
oc:  [
    [0x0 0x1 0x2 0x3] [0x0 1x0 2x0 3x0] [0x0 0x1 0x2 1x1]
    [0x1 1x0 1x1 2x1] [0x1 1x0 1x1 1x2] [0x0 1x0 1x1 2x0]
    [0x0 0x1 0x2 1x0] [0x0 0x1 1x1 2x1] [0x2 1x0 1x1 1x2]
    [0x0 1x0 2x0 2x1] [0x0 0x1 0x2 1x2] [0x1 1x1 2x0 2x1]
    [0x0 1x0 1x1 1x2] [0x0 0x1 1x0 2x0] [0x0 0x1 1x1 1x2]
    [0x1 1x0 1x1 2x0] [0x1 0x2 1x0 1x1] [0x0 1x0 1x1 2x1]
    [0x0 0x1 1x0 1x1]
]
width: [4 1 3 2 3 2 3 2 3 2 3 2 3 2 3 2 3 2 2]
score: 0
prin tui [clear]
a-line: copy [] loop 11 [append a-line " "]
a-line: rejoin ["   |"  to-string a-line "|"]
loop 20 [print a-line] prin "   " loop 13 [prin "+"] print ""
print tui compose [
    at 4x21 "TEXTRIS" at 5x21 "-------"
    at 7x20 "Use arrow keys" at 8x20 "to move/spin."
    at 10x20 "'P' = pause"
    at 13x20 "SCORE:  " (to-string score)
]
keys: open/binary/no-wait [scheme: 'console]
forever [
    random/seed now
    r: random 19
    xpos: 9
    for i 1 20 1 [
        pos: to-pair rejoin [i "x" xpos]
        do compose/deep [prin tui [at (pos)] print tui shape/(r)]
        old-r: r
        old-xpos: xpos
        if not none? wait/all [keys :00:00.30] [
            switch/default to-string copy keys [
```

```
        "p" [
            print tui [
                at 23x0 "Press [Enter] to continue"
            ]
            ask ""
            print tui [
                at 24x0 "                                "
                at 23x0 "                                "
            ]
        ]
        "^[[D" [if (xpos > 5) [
                xpos: xpos - 1
        ]]
        "^[[C" [if (xpos < (16 - compose width/(r))) [
                xpos: xpos + 1
        ]]
        "^[[A" [if (xpos < (16 - compose width/(r)))  [
                switch to-string r [
                    "1" [r: 2]
                    "2" [r: 1]
                    "3" [r: 6]
                    "4" [r: 3]
                    "5" [r: 4]
                    "6" [r: 5]
                    "7" [r: 10]
                    "8" [r: 7]
                    "9" [r: 8]
                    "10" [r: 9]
                    "11" [r: 14]
                    "12" [r: 11]
                    "13" [r: 12]
                    "14" [r: 13]
                    "15" [r: 16]
                    "16" [r: 15]
                    "17" [r: 18]
                    "18" [r: 17]
                    "19" [r: 19]
                ]
            ]
        ]
    ] []
]
do compose/deep [
    prin tui [at (pos)] print tui shape/(old-r + 19)
]
stop: false
foreach po compose oc/(r) [
    foreach coord floor [
        floor-y: to-integer first coord
        floor-x: to-integer second coord
        oc-y:  i + to-integer first po
        oc-x:  xpos + to-integer second po
        if (oc-y = (floor-y - 1)) and (floor-x = oc-x) [
            stop-shape-num: r
            stop: true
            break
        ]
    ]
]
foreach po compose oc/(old-r) [
    foreach coord floor [
        floor-y: to-integer first coord
        floor-x: to-integer second coord
        oc-y:  i + to-integer first po
```

```
            oc-x:  old-xpos + to-integer second po
            if (oc-y = (floor-y - 1)) and (floor-x = oc-x) [
                stop-shape-num: old-r
                stop: true
                break
            ]
        ]
    ]
    if stop = true [
        left-col: second pos
        width-of-shape: length? compose oc/(stop-shape-num)
        right-col: left-col + width-of-shape - 1
        counter: 1
        for current-column left-col right-col 1 [
            add-coord: compose oc/(stop-shape-num)/(counter)
            new-floor-coord: (pos + add-coord)
            append floor new-floor-coord
            counter: counter + 1
        ]
        break
    ]
]
do compose/deep [prin tui [at (pos)] print tui shape/(old-r)]
if (first pos) < 2 [
    prin tui [at 23x0]
    print "   GAME OVER!!!^/^/"
    halt
]
score: score + 10
print tui compose [at 13x28 (to-string score)]
for row 1 20 1 [
    line-is-full: true
    for colmn 5 15 1 [
        each-coord: to-pair rejoin [row "x" colmn]
        if not find floor each-coord [
            line-is-full: false
            break
        ]
    ]
    if line-is-full = true [
        remove-each cor floor [(first cor) = row]
        new-floor: copy [
            21x5 21x6 21x7 21x8 21x9 21x10 21x11 21x12 21x13
            21x14 21x15
        ]
        foreach cords floor [
            either ((first cords) < row) [
                append new-floor (cords + 1x0)
            ][
                append new-floor cords
            ]
        ]
        floor: copy unique new-floor
        score: score + 1000
        prin tui [clear]
        loop 20 [print a-line]
        prin "   " loop 13 [prin "+"] print ""
        print tui compose [
            at 4x21 "TEXTRIS" at 5x21 "-------"
            at 7x20 "Use arrow keys" at 8x20 "to move/spin."
            at 10x20 "'P' = pause"
            at 13x20 "SCORE:  " (to-string score)
        ]
        foreach was-here floor [
```

```
                if not ((first was-here) = 21) [
                    prin tui compose [at (was-here)]
                    prin "#"
                ]
            ]
        ]
    ]
]


#! ./rebol -cs
REBOL [title: "CGI Bulletin Board"]
print {content-type: text/html^/}
print read %template_header.html
bbs: load %bb.db
print {
    <center><table border=1 cellpadding=10 width=600><tr><td>
    <center><strong><font size=4>
    Please REFRESH this page to see new messages.
    </font></strong></center>
}
print-all: does [
    print {<br><hr><font size=5> Posted Messages: </font> <br><hr>}
    foreach bb (reverse bbs) [
        print rejoin [
            {<BR>Date/Time: } bb/2 {          }
            {"Name: } bb/1 {<BR><BR>} bb/3 {<BR><BR><HR>}
        ]
    ]
]
submitted: decode-cgi system/options/cgi/query-string
if submitted/2 <> none [
    entry: copy []
    append entry submitted/2
    append entry to-string (now + 3:00)
    append entry submitted/4
    append/only bbs entry
    save %bb.db bbs
    print {<BR><strong>Your message has been added: </strong><BR>}
]
print-all
print {
    <font size=5> Post A New Public Message:</font><hr>
    <FORM ACTION="http://website.com/bb/bb.cgi">
    <br> Your Name:  <br>
    <input type=text size="50" name="student"><BR><BR>
    Your Message: <br>
    <textarea name=message rows=5 cols=50></textarea><BR><BR>
    <INPUT TYPE="SUBMIT" NAME="Submit" VALUE="Post Message">
    </FORM>
    </td></tr></table></center>
}
print read %template_footer.html


#! ./rebol -cs
REBOL [title: "CGI Simple Search"]
print "content-type: text/html^/"
print [<HTML><HEAD><TITLE>"Search"</TITLE></HEAD><BODY>]
; print read %template_header.html
submitted: decode-cgi system/options/cgi/query-string
if not empty? submitted [
    phrase: submitted/2
    start-folder: to-file submitted/4
```

```
    change-dir start-folder
    found-list: ""
    recurse: func [current-folder] [
        foreach item (read current-folder) [
            if not dir? item [  if error? try [
                if find (read to-file item) phrase [
                    print rejoin [{"} phrase {" found in:  }
                        what-dir item {<BR>}]
                    found-list: rejoin [found-list newline
                        what-dir item]
                ]] [print rejoin ["error reading " item]]
            ]
        ]
        foreach item (read current-folder) [
            if dir? item [
                change-dir item
                recurse %.\
                change-dir %..\
            ]
        ]
    ]
    print rejoin [{SEARCHING for "} phrase {" in }
        start-folder {<BR><BR>}]
    recurse %.\
    print "<BR>DONE <BR>"
    ; save %found.txt found-list
    ; print read %template_footer.html
    quit
]
print [<CENTER><TABLE><TR><TD>]
print [<FORM ACTION="./search.cgi">]
print ["Text to search for:" <BR>
    <INPUT TYPE="TEXT" NAME="phrase"><BR><BR>]
print ["Folder to search in:" <BR>
    <INPUT TYPE="TEXT" NAME="folder" VALUE="../yourfolder/" ><BR><BR>]
print [<INPUT TYPE="SUBMIT" NAME="Submit" VALUE="Submit">]
print [</FORM>]
print [</TD></TR></TABLE></CENTER>]
; print read %template_footer.html
quit


#! ./rebol -cs
REBOL [title: "CGI Event Calendar"]
print "content-type: text/html^/"
print {<HTML><HEAD><TITLE>Event Calendar</TITLE></HEAD><BODY>}
bbs: load %bb.db
date: now/date
html: copy rejoin [
    {<CENTER><TABLE border=1 valign=middle width=99% height=99%>
        <TR><TD colspan=7 align=center height=8%><FONT size=5>}
    pick system/locale/months date/month {  } date/year
    {</FONT></TD></TR><TR>}
]
days: ["Sun" "Mon" "Tue" "Wed" "Thu" "Fri" "Sat"]
foreach day days [
    append html rejoin [
        {<TD bgcolor="#206080" align=center width=10% height=5%>
        <FONT face="courier new,courier" color="FFFFFF" size="+1">}
        day
        {</FONT></TD>}
    ]
]
append html {</TR><TR>}
```

```
sdate: date  sdate/day: 0
loop sdate/weekday // 7 + 1 [append html {<TD bgcolor=gray></TD>}]
while [sdate/day: sdate/day + 1 sdate/month = date/month][
    event-labels: {}
    foreach entry bbs [
        date-in-entry: 1-Jan-1001
        attempt [date-in-entry: (to-date entry/2)]
        if (date-in-entry = sdate) [
            event-labels: rejoin [
                {<font size=1>}
                event-labels
                "<strong><br><br>"
                {<a href="} to-string entry/3 {" target=_blank>}
                entry/1
                {</a>}
                "</strong>"
                {</font>}
            ]
        ]
    ]
    append html rejoin [
        {<TD bgcolor="#}
        either date/day = sdate/day ["AA9060"]["FFFFFF"]
        ; HERE, THE EVENTS ARE PRINTED IN THE APPROPRIATE DAY:
        {" height=14% valign=top>} sdate/day event-labels
        {</TD>}
    ]
    if sdate/weekday = 6 [append html {</TR><TR>}]
]
loop 7 - sdate/weekday [append html rejoin [{<TD bgcolor=gray></TD>}]]
append html {</TR></TABLE></CENTER></BODY></HTML>}
print html


#! /home/path/public_html/rebol/rebol276 -cs
REBOL [Title: "CGI Console"]
print "content-type: text/html^/"
print {<HTML><HEAD><TITLE>Console</TITLE></HEAD><BODY>}
selection: decode-cgi system/options/cgi/query-string
if ((selection/2 = none) or (selection/4 = none)) [
    print {
        <STRONG>W A R N I N G  -  Private Server, Login Required:</STRONG>
        <BR><BR>
        <FORM ACTION="./console.cgi">
        Username: <INPUT TYPE=text SIZE="50" NAME="name"><BR><BR>
        Password: <INPUT TYPE=text SIZE="50" NAME="pass"><BR><BR>
        <INPUT TYPE="SUBMIT" NAME="Submit" VALUE="Submit">
        </FORM>
    }
    quit
]
qq: [
    print {
        <CENTER><FORM METHOD="get" ACTION="./console.cgi">
        <INPUT TYPE=hidden NAME=submit_confirm VALUE="command-submitted">
        <TEXTAREA COLS="100" ROWS="18" NAME="contents"></TEXTAREA><BR><BR>
        <INPUT TYPE="SUBMIT" NAME="Submit" VALUE="Submit">
        </FORM></CENTER></BODY></HTML>
    }
]
if selection/2 = "command-submitted" [
    write %commands.txt join "REBOL[]^/" selection/4
    ; The "call" function requires REBOL version 2.76:
    call/output/error
```

```
        "/home/path/public_html/rebol/rebol276 -qs commands.txt"
        %conso.txt %conse.txt
    print rejoin [
        {<CENTER>Output: <BR><BR>}
        {<TABLE WIDTH=80% BORDER="1" CELLPADDING="10"><TR><TD><PRE>}
        read %conso.txt
        {</PRE></TD></TR></TABLE><BR><BR>}
        {Errors: <BR><BR>}
        read %conse.txt
        {</CENTER>}
    ]
    do qq
    quit
]
username: selection/2 password: selection/4
either (username = "user") and (password = "pass") [] [
    print "Incorrect Username/Password." quit
]
do qq


#!./rebol -cs
REBOL [title: "CGI Text Editor"]
print {content-type: text/html^/}
print {<HTML><HEAD><TITLE>Edit Text Document</TITLE></HEAD><BODY>}
read-cgi: func [/local data buffer][
    switch system/options/cgi/request-method [
        "POST" [
            data: make string! 1020
            buffer: make string! 16380
            while [positive? read-io system/ports/input buffer 16380][
                append data buffer
                clear buffer
            ]
        ]
        "GET" [data: system/options/cgi/query-string]
    ]
    data
]
submitted: decode-cgi read-cgi
if submitted/2 = "save" [
    ; save newly edited document:
    write to-file rejoin ["./" submitted/6 "/document.txt"] submitted/4
    print ["Document Saved."]
    print rejoin [
        {<META HTTP-EQUIV="REFRESH" CONTENT="0;
            URL=http://website.com/folder/}
        submitted/6 {">}
    ]
    quit
]
if ((submitted/2 = none) or (submitted/4 = none)) [
    print {
        <strong>W A R N I N G  -  Private Server, Login Required:
        </strong><BR><BR>
        <FORM ACTION="./edit.cgi">
        Username: <input type=text size="50" name="name"><BR><BR>
        Password: <input type=text size="50" name="pass"><BR><BR>
        <INPUT TYPE="SUBMIT" NAME="Submit" VALUE="Submit">
        </FORM>
    }
    quit
]
userlist: load %userlist.txt
```

```
folder: submitted/2
password: submitted/4
response: false
foreach user userlist [
    if ((first user) = folder) and (password = (second user)) [
        response: true
    ]
]
if response = false [print {Incorrect Username/Password.} quit]
cur-time: to-string replace/all to-string now/time {:} {-}       ; backup
document_text: read to-file rejoin [{./} folder {/document.txt}]
write to-file rejoin [
    {./} folder {/} now/date {_} cur-time {.txt}] document_text
prin {
    <strong>Be sure to SUBMIT when done:</strong><BR><BR>
    <FORM method="post" ACTION="./edit.cgi">
    <INPUT TYPE=hidden NAME=submit_confirm VALUE="save">
    <textarea cols="100" rows="15" name="contents">
}
prin replace/all document_text {</textarea>} {&lt\/textarea&gt;}
print {</textarea><BR><BR>}
print rejoin [{<INPUT TYPE=hidden NAME=folder VALUE="} folder {">}]
print {<INPUT TYPE="SUBMIT" NAME="Submit" VALUE="Submit">}
print {</FORM>}
print {</BODY></HTML>}


#! ./rebol -cs
REBOL [title: "CGI Event Signup"]
print {content-type: text/html^/}
print {<HTML><HEAD><TITLE>Event Sign-Up</TITLE></HEAD><BODY>}
events: load %event.db
a-line: [] loop 65 [append a-line "-"]
a-line: trim to-string a-line
print {
    <hr> <font size=5>" Sign up for an event:"</font> <hr><BR>
    <FORM ACTION="http://yourwebsite.com/cgi-bin/event.cgi">
    Student Name:
    <input type=text size="50" name="student"><BR><BR>
    ADD yourself to this event:              "
    <select NAME="add"><option>""<option>"all"
}
foreach event events [prin rejoin [{<option>} event/1]]
print {
    </option> </select> <BR> <BR>
    REMOVE yourself from this event:
    <select NAME="remove"><option>""<option>"all"
}
foreach event events [prin rejoin [{<option>} event/1]]
print {
    </option> </select> <BR> <BR>
    <INPUT TYPE="SUBMIT" NAME="Submit" VALUE="Submit">
    </FORM>
}
print-all: does [
    print [<br><hr><font size=5>]
    print " Currently scheduled events, and current attendance:"
    print [</font><br>]
    foreach event events [
        print rejoin [a-line {<BR>} event/1 {BR} a-line {<BR>}]
        for person 2 (length? event) 1 [
            print event/:person
            print {<BR>}
        ]
```

```
        print {<BR>}
    ]
    print {</BODY></HTML>}
]
submitted: decode-cgi system/options/cgi/query-string
if submitted/2 <> none [
    if ((submitted/4 = "") and (submitted/6 = "")) [
        print {
            <strong> Please try again. You must choose an event.</strong>
        }
        print-all
        quit
    ]
    if ((submitted/4 <> "") and (submitted/6 <> "")) [
        print {
            <strong> Please try again. Choose add OR remove.</strong>
        }
        print-all
        quit
    ]
    if submitted/4 = "all" [
        foreach event events [append event submitted/2]
        save %event.db events
        print {
            <strong> Your name has been added to every event:</strong>
        }
        print-all
        quit
    ]
    if submitted/6 = "all" [
        foreach event events [
            if find event submitted/2 [
                remove-each name event [name = submitted/2]
                save %event.db events
            ]
        ]
        print {
            <strong> Your name has been removed from all events:</strong>
        }
        print-all
        quit
    ]
    foreach event events [
        if (find event submitted/4) [
            append event submitted/2
            save %event.db events
            print {
                <strong> Your name has been added to the selected event:
                </strong>
            }
            print-all
            quit
        ]
    ]
    found: false
    foreach event events [
        if (find event submitted/6) [
            if (find event submitted/2) [
                remove-each name event [name = submitted/2]
                save %event.db events
                print {
                    <strong>
                    Your name has been removed from the selected event:
                    </strong>
```

```
                }
                print-all
                quit
                found: true
            ]
        ]
    ]
    if found <> true [
        print {
            <strong> That name is not found in the specified event!"
            </strong>
        }
        print-all
        quit
    ]
]
print-all


#! ./rebol -cs
REBOL [title: "CGI File Downloader"]
submitted: decode-cgi system/options/cgi/query-string
root-path: "/home/path"
if ((submitted/2 = none) or (submitted/4 = none)) [
    print "content-type: text/html^/"
    print [<STRONG>"W A R N I N G  -  "]
    print ["Private Server, Login Required:"</STRONG><BR><BR>]
    print [<FORM ACTION="./download.cgi">]
    print [" Username: " <INPUT TYPE=text SIZE="50" NAME="name"><BR><BR>]
    print [" Password: " <INPUT TYPE=text SIZE="50" NAME="pass"><BR><BR>]
    print [" File: "<BR><BR>]
    print [<INPUT TYPE=text SIZE="50" NAME="file" VALUE="/public_html/">]
    print [<BR><BR>]
    print [<INPUT TYPE="SUBMIT" NAME="Submit" VALUE="Submit">]
    print [</FORM>]
    quit
]
username: submitted/2 password: submitted/4
either (username = "user") and (password = "pass") [
    ; if user/pass is ok, go on
][
    print "content-type: text/html^/"
    print "Incorrect Username/Password." quit
]
print rejoin [
    "Content-Type: application/x-unknown"
    newline
    "Content-Length: "
    (size? to-file join root-path submitted/6)
    newline
    "Content-Disposition: attachment; filename="
    (second split-path to-file submitted/6)
    newline
]
data: read/binary to-file join root-path submitted/6
data-length: size? to-file join root-path submitted/6
write-io system/ports/output data data-length


#! ./rebol -cs
REBOL [Title: "CGI HTTP File Uploader"]
print {content-type: text/html^/}
print {<HTML><HEAD><TITLE>File Upload</TITLE></HEAD>}
print {<BODY><br><br><center><table width=95% border=1>}
```

```
print {<tr><td width=100%><br><center>}
read-cgi: func [/local data buffer][
    switch system/options/cgi/request-method [
        "POST" [
            data: make string! 1020
            buffer: make string! 16380
            while [positive? read-io system/ports/input buffer 16380][
                append data buffer
                clear buffer
            ]
        ]
        "GET" [data: system/options/cgi/query-string]
    ]
    data
]
submitted: read-cgi
if submitted/2 = none [
    print {
        <FORM ACTION="./upload.cgi"
        METHOD="post" ENCTYPE="multipart/form-data">
            <strong>Upload File:</strong><br><br>
            <INPUT TYPE="file" NAME="photo"> <br><br>
            <INPUT TYPE="submit" NAME="Submit" VALUE="Upload">
        </FORM>
        <br></center></td></tr></table></BODY></HTML>
    }
    quit
]
decode-multipart-form-data: func [
    p-content-type
    p-post-data
    /local list ct bd delim-beg delim-end non-cr non-lf non-crlf mime-part
] [
    list: copy []
    if not found? find p-content-type "multipart/form-data" [return list]
    ct: copy p-content-type
    bd: join "--" copy find/tail ct "boundary="
    delim-beg: join bd crlf
    delim-end: join crlf bd
    non-cr:     complement charset reduce [ cr ]
    non-lf:     complement charset reduce [ newline ]
    non-crlf:   [ non-cr | cr non-lf ]
    mime-part:  [
        ( ct-dispo: content: none ct-type: "text/plain" )
        delim-beg ; mime-part start delimiter
        "content-disposition: " copy ct-dispo any non-crlf crlf
        opt [ "content-type: " copy ct-type any non-crlf crlf ]
        crlf ; content delimiter
        copy content
        to delim-end crlf ; mime-part end delimiter
        ( handle-mime-part ct-dispo ct-type content )
    ]
    handle-mime-part: func [
        p-ct-dispo
        p-ct-type
        p-content
        /local tmp name value val-p
    ] [
        p-ct-dispo: parse p-ct-dispo {;="}
        name: to-set-word (select p-ct-dispo "name")
        either (none? tmp: select p-ct-dispo "filename")
               and (found? find p-ct-type "text/plain") [
            value: content
        ] [
```

```
            value: make object! [
                filename: copy tmp
                type: copy p-ct-type
                content: either none? p-content [none][copy p-content]
            ]
        ]
        either val-p: find list name
            [change/only next val-p compose [(first next val-p) (value)]]
            [append list compose [(to-set-word name) (value)]]
    ]
    use [ct-dispo ct-type content] [
        parse/all p-post-data [some mime-part "--" crlf]
    ]
    list
]
cgi-object: construct decode-multipart-form-data
    system/options/cgi/content-type copy submitted
the-file: last split-path to-file copy cgi-object/photo/filename
write/binary the-file cgi-object/photo/content
print {
    <strong>UPLOAD COMPLETE</strong><br><br>
    <strong>Files currently in this folder:</strong><br><br>
}
folder: sort read %.
foreach file folder [
    print [rejoin [{<a href="./} file {">} file {</a><br>}]]
]
print {<br></td></tr></table></BODY></HTML>}


#! ./rebol276 -cs
REBOL [title: "CGI WAP File Viewer"]
submitted: decode-cgi system/options/cgi/query-string
prin {Content-type: text/vnd.wap.wml^/^/}
prin {<?xml version="1.0" encoding="iso-8859-1"?>^/}
prin {<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">^/}
if submitted/2 = none [
    print {<wml><card id="1" title="Select Teacher"><p>}
    ; print {Name: <input name="name" size="12"/>}
    folders: copy []
    foreach folder read %./Teachers/ [
        if find to-string folder {/} [append folders to-string folder]
    ]
    print {Teacher: <select name="teacher">}
    foreach folder folders [
        folder: replace/all folder {/} {}
        print rejoin [
            {<option value="} folder {">} folder {</option>}
        ]
    ]
    print {</select>
    <anchor>
       <go method="get" href="wap.cgi">
           <postfield name="teacher" value="$(teacher)"/>
       </go>
       Submit
    </anchor>}
    print {</p></card></wml>}
    quit
]
count: 0
parse read join http://site.com/folder/ submitted/2 [
    thru submitted/2 copy p to "past students"
```

```
]
print {<wml>}
forskip p 130 [
    count: count + 1
    print rejoin [
        {<card id="} count {" title="} submitted/2 "-" count {"><p>}
    ]
    print rejoin [
        {<anchor>Next<go href="#} (count + 1) {"/></anchor>}
    ]
    print rejoin [{<anchor>Back<prev/></anchor>}]
    print copy/part p 130
    print {</p></card>}
]
print {</wml>}
quit


#!./rebol276 -cs
REBOL [title: "CGI WAP File Insert"]
submitted: decode-cgi system/options/cgi/query-string
prin {Content-type: text/vnd.wap.wml^/^/}
prin {<?xml version="1.0" encoding="iso-8859-1"?>^/}
prin {<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">^/}
if submitted/2 = none [
    print {<wml><card id="1" title="Select Teacher"><p>}
    print {Insert Text: <input name="thetext" size="12"/>}
    folders: copy []
    foreach folder read %./Teachers/ [
        if find to-string folder {/} [
            append folders to-string folder
        ]
    ]
    print {Teacher: <select name="teacher">}
    foreach folder folders [
        folder: replace/all folder {/} {}
        print rejoin [
            {<option value="} folder {">} folder {</option>}
        ]
    ]
    print {</select>
    <anchor>
       <go method="get" href="wapinsert.cgi">
           <postfield name="teacher" value="$(teacher)"/>
           <postfield name="thetext" value="$(thetext)"/>
       </go>
       Submit
    </anchor>}
    print {</p></card></wml>}
    quit
]
chosen-file: rejoin [%./Teachers/ submitted/2 "/schedule.txt"]
adjusted-file: read/lines chosen-file
insert next next next next adjusted-file submitted/4
write/lines chosen-file adjusted-file
count: 0
parse read join http://site.com/folders/ submitted/2 [
    thru submitted/2 copy p to "past students"
]
print {<wml>}
forskip p 130 [
    count: count + 1
    print rejoin [
```

```
        {<card id="} count {" title="} submitted/2 "-" count {"><p>}
    ]
    print rejoin [
        {<anchor>Next<go href="#} (count + 1) {"/></anchor>}
    ]
    print rejoin [{<anchor>Back<prev/></anchor>}]
    print copy/part p 130
    print {</p></card>}
]
print {</wml>}
quit


#!./rebol276 -cs
REBOL [title: "CGI WAP Mail Reader"]
submitted: decode-cgi system/options/cgi/query-string
prin {Content-type: text/vnd.wap.wml^/^/}
prin {<?xml version="1.0" encoding="iso-8859-1"?>^/}
prin {<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
"http://www.wapforum.org/DTD/wml_1.1.xml">^/}
accounts: [
    ["pop.server" "smtp.server" "username" "password" you@site.com]
    ["pop.server2" "smtp.server2" "username" "password" you@site2.com]
    ["pop.server3" "smtp.server3" "username" "password" you@site3.com]
]
if ((submitted/2 = none) or (submitted/2 = none)) [
    print {<wml><card id="1" title="Select Account"><p>}
    print {Account: <select name="account">}
    forall accounts [
        print rejoin [
            {<option value="} index? accounts {">}
            last first accounts {</option>}
        ]
    ]
    print {</select>
    <select name="readorsend">
        <option value="readselect">Read</option>
        <option value="sendinput">Send</option>
    </select>
    <anchor>
       <go method="get" href="wapmail.cgi">
           <postfield name="account" value="$(account)"/>
           <postfield name="readorsend" value="$(readorsend)"/>
       </go>
       Submit
    </anchor>}
    print {</p></card></wml>}
    quit
]
if submitted/4 = "readselect" [
    t: pick accounts (to-integer submitted/2)
    system/schemes/pop/host:  t/1
    system/schemes/default/host: t/2
    system/schemes/default/user: t/3
    system/schemes/default/pass: t/4
    system/user/email: t/5
    prin {<wml><card id="1" title="Choose Message"><p>}
    prin rejoin [{<setvar name="account" value="} submitted/2 {"/>}]
    prin {<select name="chosenmessage">}
    mail: read to-url join "pop://" system/user/email
    foreach message mail  [
        pretty: import-email message
        if (find pretty/subject "***SPAM***") = none [
            replace/all pretty/subject {"} {}
```

```
            replace/all pretty/subject {&} {}
            prin rejoin [
                {<option value="}
                pretty/subject
                {">}
                pretty/subject
                {</option>}
            ]
        ]
    ]
    print {</select>
    <anchor>
       <go method="get" href="wapmail.cgi">
           <postfield name="subroutine" value="display"/>
           <postfield name="chosenmessage" value="$(chosenmessage)"/>
           <postfield name="account" value="$(account)"/>
       </go>
       Submit
    </anchor>
    </p></card></wml>}
    quit
]
if submitted/2 = "display" [
    t: pick accounts (to-integer submitted/6)
    system/schemes/pop/host:  t/1
    system/schemes/default/host: t/2
    system/schemes/default/user: t/3
    system/schemes/default/pass: t/4
    system/user/email: t/5
    prin {<wml><card id="1" title="Display Message"><p>}
    mail: read to-url join "pop://" system/user/email
    foreach message mail  [
        pretty: import-email message
        if pretty/subject = submitted/4 [
            replace/all pretty/content {"} {}
            replace/all pretty/content {&} {}
            replace/all pretty/content {3d} {}
            strip: copy ""
            foreach item (load/markup pretty/content) [
                if ((type? item) = string!) [strip: join strip item]
            ]
            prin strip
        ]
    ]
    print {</p></card></wml>}
    quit
]


#! /home/path/public_html/rebol/rebol -cs
REBOL [title: "CGI Remove Unwanted Emails"]
print "content-type: text/html^/"
print [<HTML><HEAD><TITLE>"Remove Emails"</TITLE></HEAD><BODY>]
spam: [
    {Failure} {Undeliverable} {failed} {Returned Mail} {not be delivered}
    {mail status notification} {Mail Delivery Subsystem} {(Delay)}
]
print "logging in..."
mail: open pop://user:pass@site.com
print "logged in"
while [not tail? mail] [
    either any [
        (find first mail spam/1) (find first mail spam/2)
        (find first mail spam/3) (find first mail spam/4)
```

```
        (find first mail spam/5) (find first mail spam/6)
        (find first mail spam/7) (find first mail spam/8)
    ][
        remove mail
        print "removed"
    ][
        mail: next mail
    ]
    print length? mail
]
close mail
print [</BODY></HTML>]
quit


REBOL [title: "chmod777to555"]
start-dir: what-dir
all-files: to-file join start-dir %find777all.txt
write all-files ""
recurse: func [current-folder] [
    out-data: copy ""
    write/append all-files rejoin["CURRENT_DIRECTORY:  " what-dir newline]
    call/output {ls -al} out-data
    write/append all-files join out-data newline
    foreach item (read current-folder) [
        if dir? item [
            change-dir item
            recurse %.\
            change-dir %..\
        ]
    ]
]
recurse %.\
file-list: to-file join start-dir %found777.txt
write file-list ""
current-directory: ""
foreach line (read/lines all-files) [
    if find line "CURRENT_DIRECTORY:  " [
        current-directory: line
    ]
    if find line "rwxrwxrwx" [
        write/append file-list rejoin [
            (find/match current-directory "CURRENT_DIRECTORY:  ")
            (last parse/all line " ")
        ]
        write/append file-list newline
    ]
]
foreach file (read/lines file-list) [
    call rejoin [{chmod 755 } (to-local-file file)]
]


REBOL [title: "FTP Tool"]
Instructions: {

    Enter your username, password, and FTP URL in the text field, and
    hit [ENTER].

    BE SURE TO END YOUR FTP URL PATH WITH "/".

    URLs can be saved and loaded in multiple config files for future use.

    CONFIG FILES ARE STORED AS PLAIN TEXT, SO KEEP THEM SECURE.
```

```
    Click folders to browse through any dir on your web server.  Click
    text files to open, edit and save changes back to the server.
    Click images to view.  Also upload/download any type of file,
    create new files and folders, change file names, copy and delete
    files, change permissions, etc.

}
connect: does [
    either (to-string last p/text) = "/" [
    if error? try [
            f/data: sort append read to-url p/text "../" show f
        ][
            alert "Not a valid FTP address, or the connection failed."
        ]
    ][
        editor to-url p/text
    ]
]
view center-face layout [
    p: field 600 "ftp://user:pass@website.com/" [connect]
    across
    btn "Connect" [connect]
    btn "Load URL" [
        config: to-file request-file/file %/c/ftp.cfg
        either exists? config [
            if (config <> %none) [
                my-urls: copy []
                foreach item read/lines config [append my-urls item]
                if error? try [
                    p/text: copy request-list "Select a URL:" my-urls
                ] [break]
            ]
        ][
            alert "First, save some URLs to that file..."
        ]
        show p focus p
    ]
    btn "Save URL" [
        url: request-text/title/default "URL to save:" p/text
        if url = none [break]
        config-file: to-file request-file/file/save %/c/ftp.cfg
        if (url <> none) and (config-file <> %none) [
            if not exists? config-file [
                write/lines config-file ftp://user:pass@website.com/
            ]
            write/append/lines config-file to-url url
            alert "Saved"
        ]
    ]
    below
    f: text-list 600x350 [
        either (to-string value) = "../" [
            for i ((length? p/text) - 1) 1 -1 [
                if (to-string (pick p/text i)) = "/" [
                    clear at p/text (i + 1) show p
                    f/data: sort append read to-url p/text "../" show f
                    break
                ]
            ]
        ][
            either (to-string last value) = "/" [
                p/text: rejoin [p/text value] show p
                f/data: sort append read to-url p/text "../" show f
```

```
        ][
            if ((request "Edit/view this file?") = true) [
                either find [%.jpg %.png %.gif %.bmp] suffix? value [
                    view/new layout [
                        image load to-url join p/text value
                    ]
                ][
                    editor to-url rejoin [p/text value]
                ]
            ]
        ]
    ]
]
across
btn "Get Info" [
    p-file: to-url rejoin [p/text f/picked]
    alert rejoin ["Size: " size? p-file " Date: " modified? p-file]
]
btn "Delete" [
    p-file: to-url request-text/title/default "File to delete:"
        join p/text f/picked
    if ((confirm: request "Are you sure?") = true) [delete p-file]
    f/data: sort append read to-url p/text "../" show f
    if confirm = true [alert "File deleted"]
]
btn "Rename" [
    new-name: to-file request-text/title/default "New File Name:"
        to-string f/picked
    if ((confirm: request "Are you sure?") = true) [
        rename (to-url join p/text f/picked) new-name
    ]
    f/data: sort append read to-url p/text "../" show f
    if confirm = true [alert "File renamed"]
]
btn "Copy" [
    new-name: to-url request-text/title/default "New Path:"
        (join p/text f/picked)
    if ((confirm: request "Are you sure?") = true) [
        write/binary new-name read/binary to-url join p/text f/picked
    ]
    f/data: sort append read to-url p/text "../" show f
    if confirm = true [alert "File copied"]
]
btn "New File" [
    p-file: to-url request-text/title/default "New File Name:"
        join p/text "ENTER-A-FILENAME.EXT"
    if ((confirm: request "Are you sure?") = true) [
        write p-file ""
        ; editor p-file
    ]
    f/data: sort append read to-url p/text "../" show f
    if confirm = true [alert "Empty file created - click to edit."]
]
btn "New Dir" [
    make-dir x: to-url request-text/title/default "New folder:" p/text
    alert "Folder created"
    p/text: x show p
    f/data: sort append read to-url p/text "../" show f
]
btn "Download" [
    file: request-text/title/default "File:" (join p/text f/picked)
    l-file: next to-string (find/last (to-string file) "/")
    save-as: request-text/title/default "Save as..." to-string l-file
    write/binary (to-file save-as) (read/binary to-url file)
```

```
        alert "Download Complete"
    ]
    btn "Upload" [
        file: to-file request-file
        r-file: request-text/title/default "Save as..."
            join p/text (to-string to-relative-file file)
        write/binary (to-url r-file) (read/binary file)
        f/data: sort append read to-url p/text "../" show f
        alert "Upload Complete"
    ]
    btn "Chmod" [
        p-file: to-url request-text/default rejoin [p/text f/picked]
        chmod: to-block request-text/title/default "Permissions:"
            "read write execute"
        write/binary/allow p-file (read/binary p-file) chmod
        alert "Permissions changed"
    ]
    btn-help [inform layout[backcolor white text bold as-is instructions]]
    do [focus p]
]


REBOL [title: "Jeopardy"]
config: {

    REBOL []

    ;_____________________________________________________________________

    sizer: 4

    Category-1: "Category 1"
    Category-2: "Category 2"
    Category-3: "Category 3"
    Category-4: "Category 4"
    Category-5: "Category 5"

    answers: [
        "$100 Answer, Category 1"
        "$100 Answer, Category 2"
        "$100 Answer, Category 3"
        "$100 Answer, Category 4"
        "$100 Answer, Category 5"
        "$200 Answer, Category 1"
        "$200 Answer, Category 2"
        "$200 Answer, Category 3"
        "$200 Answer, Category 4"
        "$200 Answer, Category 5"
        "$300 Answer, Category 1"
        "$300 Answer, Category 2"
        "$300 Answer, Category 3"
        "$300 Answer, Category 4"
        "$300 Answer, Category 5"
        "$400 Answer, Category 1"
        "$400 Answer, Category 2"
        "$400 Answer, Category 3"
        "$400 Answer, Category 4"
        "$400 Answer, Category 5"
        "$500 Answer, Category 1"
        "$500 Answer, Category 2"
        "$500 Answer, Category 3"
        "$500 Answer, Category 4"
        "$500 Answer, Category 5"
    ]
```

```
    questions: [
        "$100 Question, Category 1"
        "$100 Question, Category 2"
        "$100 Question, Category 3"
        "$100 Question, Category 4"
        "$100 Question, Category 5"
        "$200 Question, Category 1"
        "$200 Question, Category 2"
        "$200 Question, Category 3"
        "$200 Question, Category 4"
        "$200 Question, Category 5"
        "$300 Question, Category 1"
        "$300 Question, Category 2"
        "$300 Question, Category 3"
        "$300 Question, Category 4"
        "$300 Question, Category 5"
        "$400 Question, Category 1"
        "$400 Question, Category 2"
        "$400 Question, Category 3"
        "$400 Question, Category 4"
        "$400 Question, Category 5"
        "$500 Question, Category 1"
        "$500 Question, Category 2"
        "$500 Question, Category 3"
        "$500 Question, Category 4"
        "$500 Question, Category 5"
    ]

    ;_____________________________________________________________________

}
do config
header: load to-binary decompress 64#{
eJyVj3s804v/xz/bzGyFucRGohgm1ziuuU7CNGxWuTaxbBpyLVJUpItLcyfXNIZI
jkuJboqFJsNodG9uuXRyya18ncfj93t8//4+3/+9X6/H6/1+bY1ufQKQzg5ODgAI
BAIo2wNsjQE4QFQEKgoVERWFisJgomIIJAIBhyNkJSR3IuVlUSh5WTk5BSWMsoKi
6m45ORVtFVV1DawWFq2so6+jqY/RxGr+GwKCwWAIMYQMAiGjqSCnoPk/s/UMkBID
dAEmBLQPAEuBIFKgrZeAIgBAQP8C/D/bT4qIbS+3xUNIAASGgiFQsNhOKPRfEQSG
ANtlpKT3wmQMxGT32br7X5JHHTAKiGCW75JTNbQjkk6GZzYOqGl63Hl++UrSA952
jAIIAEH/e2AbMACCbMdsa5pSABgMgkBEwJD/c4DAAERKRNpAZq8tdJ+7PzHc7lL5
gy0BsHPbAJaCSAHWwNFKVxP98V4tOkWdfD7FEjacdjSkarU//C/n4IDH8tIsGPlr
F8jSOHVpqLEgNs++1taBHGuekzY4eM/ojEII4R/1Q+ZIEH2QWTpkVWn+ntRBthgZ
kympJr8jt/eRTxvTS1RNGD3meRMcm4kAZ2CAnTULVlJflfeI8BVvJWeJaNZitC7j
HSzwe21RvYdfbQEZcgNoufW05RxBPLziaGaLDnMiIu5cgjXhKavuH/3ewXQVtg6Q
BLgRCGfHVnFmn6LPF/fJDI1/TejRG5nB2X8vi0llhhEm3Fb/XnK+KeG/sKxkl6Py
E3ZteGrG4qqpYSazc72YsFrY6n0ht0oo2EEs4dhdJjJvOThxbRVx3ruRD921c4ct
XXp89nPCL1Ikh1Zwlkpv LRz53sKF8WfDpRW0V7dePB2671umDPyzp0PJxckwXbHc
pTnSOjTC54hLNXHdWIVdxdi9pDKhqNWOrCdNu3LPTh8e6tlxg9pQZw/KFHWY2OGM
Z33g/Y140axOK5pkwP2FSiCj0RhPPofjhXBtt52xdJjhjMZC9IMH+GNXzhlmhij6
fGjVfqN1lIHPvdV686aDXaIb6tn6gH1kYn1bpyM5Fi6c6cIsFn39lXJXLybMmRTv
soo4Nv7eqf/Byy2ANn2FByPXt1Xt8KZQEjhR08GMzINfIqIfBVT7HI4YKYlv/MRP
bv94k1JMSbSZATZOlnwEa64bdZBL/dZ6iEmlv1FwwpnNoCoplqla0drmB70Sd/H1
sZ9ijEDcfSc3TrdMbtAs3ZOg0+s96CofOOEnUod2Xqo2TylA/5xW1LtJzLNtp8vS
1UwD/YqprdJffrIwcdFhwvmr+CKmQDfT+P2R2AvojAzZ4RItnkwq00z74hIujcx1
Mzh42OioCXKvxeU3Ze4LhSJNyZYt72K7XrxlYf0p1TZX2MXlrulNBYgLUc2umSfJ
9fPvBnVHGUEaCRImNyxX74XvEyg79i/XIB/bfxsP1DCbf7O03rV/mvFg3u344qJ6
8IDbp3PMP47rJyasP1ii1wgX3yT4Zzh8dPztIlE1r3yH+6TqFHKa0ULlS3gTzBPE
oj4lDr4sEL9lrb7w2lG5jL3+g8+L+V5QmpvPdnGOED5HQNzNxvqubX52yVriGraq
tO6Jrj6r7imhjdzFQqA3yqLzMDmjNeevW4rluGeg03qjg5oeh8z8EE0fVZ7+TelY
e3W27vblk7f1nIP9tWUx+PCfl50zdzXALUJVIesptLVwlles2qRi0C0KawvonDkS
0zL7bfgZvd+C5RxyaI+NmlSmpC36BtZYzzsVXrEgGDPrIzSG/uVbdKpq2DcrlPyi
N42GS+7PFLRyYffvkrYAZeof6SN31Ir1RJfPpgUQok53lbrErr+803Wn8lLXOyzc
bNkDgxT0n0PxnGhFDEeBBEs3nirrFOlQfMk1ZaKTK+lOZkzuhnKYwwE1xeMToRUa
m1188t34vrAtIIR9quLENPsjKebLtLCelN50PXG+jtoA74lKjV7Wr3Oc3Az4bqHH
```

```
fC+6atqPZVOY5d5DeF5My+8XnELv+nu97Rt08WoK2HySEbXy6rSNWuimAyaJS7A6
SX0ou0A3F0RV4b2KuCabPbAT8W5z54Vty8Izbu7qnrLNNLxwfLHB5liKIqWUNJ90
UfHge9e0PKqdZ/zrQxifKoPGRGGVylrq3QXxlbRaHMtvl1ezceO38aWZYhJPuMJH
ejSw/FUCKXPtlhuR899Au2LnD/fl/tDL75bbTwxWxWELbRTCUw+KEEO4+x4f+yK7
3NIWrBDsOyJweXaOzW5N2+4XZNVQ8GIlt3u37tUzZQKfQtWQdgLrxKIsAlpsf6tD
o/ZAtHwp++FX7iNdP1t88Jk0P6Rhaytfjl1r5fos47hqjFXgyu3S1PEfY9U/P058
QWn5uLJ7wslybYFUsvopb0qbc/Nnm7CRPWdKT7sQuDHZpfn5uRG+T86/YcEu3Hrt
qNEbH/QR/R0a0GSdJJ3lFmDdk2+Vrw2R/IrOT3wZ3zliXC5Qzz5s5/XnCsqU3Ryv
RyQmaybczOGgx6kGS0+DCivDV/LMjJpNtDiBbQ1wOFQtcMMkZZ21r+++T2P02ZpG
riura80pOBzlxXNgZhgO0q4p3T6ePqo/9ojap3RR5kcx41Jp9ustIK5ctUlT/Zsb
SqOttfW3g+LDtANDVTJz3ZW78YyWXH4z+5MatFYdaTpNZtaOeCzr0gsqN9pkqlnB
aJE5SAUx9MS4hQRy8gt+7QTuz+hSoPD+StBsxmHcpWWzji2A+PlddStmXqvTld7D
It5HXH6KH6UkVzP4LFPesu165Bzn7PU8M+Eeca0dcPOOR8hk2tBMbjM1GnG6Th4P
bTaewuxw9UHqx6WXdQZ1QgV1Wfvt0esmCRs35On1p7W1RHe+rRTwm3eeS/YJwTX1
7seY44NCcNbGVr4UV7kQr2W7n+w+J+LEqePRR6qbW9KcsOJ3lRuuNGmaedb2fnia
6r+SRnLzuwMnTD0nWOqHS+iGR0zPl9NCc3fVwrTvo/6EMaQHown3/SQZuBn+85z0
A5Qn0pwjxFWWaw5bnzpBTHGiBzh3eZuO3bPe4ffgx8rpjoETl/GxCwpXrQ0rUNzI
/GzFqw/+umAS0bYoPoTMSieKOYANUqVjVOt/dJl2R8VWFsLYeUcT6hG/LbnCQaWu
nYh71cfoGmhOulEmjHajoedAYNc79Jq4fcZ13veLs3U0nW8efhb3IANWlUaGEXz6
SOqrnr5OXQ9lGobK/1kkW0UBoaYMwLUk0i5sSk1kr21+Yb53rXgFBIOdMpShR70h
l/ejIBs3xoZzsiML7GcHNCtryNM5U3u1WxKJDfcCM+MSQKBzcHP5l27zTafCJWtB
ODP3kcH3oNnHd6pC/nY/mV99Rv1tdJ6ClirctOvPIGhOXqMuuK+OWNgib6At5uzn
hDUjZBUh45SPH50uEPDDVJsQc3AvvfEGL+magCbq2xiep1ZX3aYTbenXi9+g4dfh
73l5xFvLRDrUvMFD1vLgr+e/bK6GIV6m/XJ7pBB8iLQo/iu63OmYtppIog038vOe
M0qCnEPH7Ds4AirpYAGrL/tc0y4lDdT3jHiSLyb1w0K38rcmQRJwaRHrX9ZPT6Y/
kzCWO9z6mFnaBuOHBH/rVhqrMcQXkE0JBz7nD3ziZVdpWE35yFO9Tq4Fz8T51Wjk
U1qQiR0I1OBDbpzMpLjyzuVxCgmbtOHcvflM3HK5lPbPNYf1oj2/U72/zl+UOsEj
SsbQVq/tcZMeKOMHTdJIxKZ3KaxBN7veuhaPRHBWpofxGGz2ksnNHPPlfguPGhfj
r/NsDNaRd5VnDHf8s5byWOa6TG5atR7hkLEfVbytUc6X6PR6eHpj4CyLau5BBrL2
d1wP2QJsHD7ouDj3hiskzdk2zN5mlIep3NXEgk5zuwNas4+s5hwvI9ck90b4FtBD
iSSWDqbMQ2IS6FIsEbJ9RxI+Y4V7c3HQSmqh4l4Vmbb0GTMNnrRjD8cyUiouJjHT
kyuWUsMHnnAXTzE3nkbF6X/2ezD1aHCjwNdz691/ABEeZGXYCwAA
}
do-button: func [num] [
    alert pick answers num
    alert pick questions num
    if find [1 2 3 4 5] num [val: $100]
    if find [6 7 8 9 10] num [val: $200]
    if find [11 12 13 14 15] num [val: $300]
    if find [16 17 18 19 20] num [val: $400]
    if find [21 22 23 24 25] num [val: $500]
    correct: request-list "Select:" ["Player 1 answered correctly"
        "Player 1 answered incorrectly" "Player 2 answered correctly"
        "Player 2 answered incorrectly" "Player 3 answered correctly"
        "Player 3 answered incorrectly" "Player 4 answered correctly"
        "Player 4 answered incorrectly"
    ]
    switch correct  [
        "Player 1 answered correctly" [
            player1/text: to-string ((to-money player1/text) + val)
            show player1
        ]
        "Player 1 answered incorrectly" [
            player1/text: to-string ((to-money player1/text) - val)
            show player1
        ]
        "Player 2 answered correctly" [
            player2/text: to-string ((to-money player2/text) + val)
            show player2
        ]
        "Player 2 answered incorrectly"[
            player2/text: to-string ((to-money player2/text) - val)
            show player2
        ]
```

```
        "Player 3 answered correctly" [
            player3/text: to-string ((to-money player3/text) + val)
            show player3
        ]
        "Player 3 answered incorrectly" [
            player3/text: to-string ((to-money player3/text) - val)
            show player3
        ]
        "Player 4 answered incorrectly"[
            player4/text: to-string ((to-money player4/text) - val)
            show player4
        ]
        "Player 4 answered correctly" [
            player4/text: to-string ((to-money player4/text) + val)
            show player4
        ]
    ]
]
view center-face layout gui: [
    tabs (sizer * 20)
    backdrop effect [gradient 1x1 tan brown]
    style button button effect [gradient blue blue/2] (
        to-pair rejoin [(20 * sizer) "x" (13 * sizer)]
    ) font [size: (sizer * 6)]
    style box box brown (to-pair rejoin [(20 * sizer) "x" (7 * sizer
        )]) font [size: (sizer * 3)]
    image header (to-pair rejoin [(132 * sizer) "x" (8 * sizer)]) [
        contin: request/confirm {
            This will end the current game.  Continue?}
        if contin = false [break]
        loadoredit:  request/confirm "Load previously edited config file?"
        if loadoredit = true [
            do to-file request-file/title/file {
                Choose config file to use:} "File" %default_config.txt
            unview
            view center-face layout gui
            break
        ]
        alert {Edit carefully, maintaining all quotation marks.
            You can open a previously saved file if needed.
            When done, click SAVE-AS and then QUIT.
            Be sure choose a filename/folder
            location that you'll be able to find later.
        }
        write %default_config.txt config
        unview
        editor %default_config.txt
        alert {Now choose a config file to use (most likely the file
            you just edited).}
        do to-file request-file/title/file {
            Choose config file to use:} "File" %default_config.txt
        view center-face layout gui
    ]
    space (to-pair rejoin [(8 * sizer) "x" (2 * sizer)])
    pad (sizer * 2)
    across
    box (to-pair rejoin [(132 * sizer) "x" (2 * sizer)]
        ) effect [gradient 1x0 brown black]
    return
    box Category-1
    box Category-2
    box Category-3
    box Category-4
    box Category-5
```

```
    return
    box (to-pair rejoin [(132 * sizer) "x" (2 * sizer)]
        ) effect [gradient 1x0 brown black]
    return
    button "$100" [face/feel: none face/text: "" do-button 1]
    button "$100" [face/feel: none face/text: "" do-button 2]
    button "$100" [face/feel: none face/text: "" do-button 3]
    button "$100" [face/feel: none face/text: "" do-button 4]
    button "$100" [face/feel: none face/text: "" do-button 5]
    return
    button "$200" [face/feel: none face/text: "" do-button 6]
    button "$200" [face/feel: none face/text: "" do-button 7]
    button "$200" [face/feel: none face/text: "" do-button 8]
    button "$200" [face/feel: none face/text: "" do-button 9]
    button "$200" [face/feel: none face/text: "" do-button 10]
    return
    button "$300" [face/feel: none face/text: "" do-button 11]
    button "$300" [face/feel: none face/text: "" do-button 12]
    button "$300" [face/feel: none face/text: "" do-button 13]
    button "$300" [face/feel: none face/text: "" do-button 14]
    button "$300" [face/feel: none face/text: "" do-button 15]
    return
    button "$400" [face/feel: none face/text: "" do-button 16]
    button "$400" [face/feel: none face/text: "" do-button 17]
    button "$400" [face/feel: none face/text: "" do-button 18]
    button "$400" [face/feel: none face/text: "" do-button 19]
    button "$400" [face/feel: none face/text: "" do-button 20]
    return
    button "$500" [face/feel: none face/text: "" do-button 21]
    button "$500" [face/feel: none face/text: "" do-button 22]
    button "$500" [face/feel: none face/text: "" do-button 23]
    button "$500" [face/feel: none face/text: "" do-button 24]
    button "$500" [face/feel: none face/text: "" do-button 25]
    return
    box (to-pair rejoin [(132 * sizer) "x" (2 * sizer)]
        ) effect [gradient 1x0 brown black]
    return tab
    box "Player 1:" effect [gradient 1x1 tan brown]
    player1: box white "$0" font [color: black size: (sizer * 4)] [
        face/text: request-text/title/default "Enter Score:" face/text
    ]
    box "Player 2:" effect [gradient 1x1 tan brown]
    player2: box white "$0" font [color: black size: (sizer * 4)] [
        face/text: request-text/title/default "Enter Score:" face/text
    ]
    return tab
    box "Player 3:" effect [gradient 1x1 tan brown]
    player3: box white "$0" font [color: black size: (sizer * 4)] [
        face/text: request-text/title/default "Enter Score:" face/text
    ]
    box "Player 4:" effect [gradient 1x1 tan brown]
    player4: box white "$0" font [color: black size: (sizer * 4)] [
        face/text: request-text/title/default "Enter Score:" face/text
    ]
]


REBOL [title: "Guitar Chords"]
help: {
    This program creates guitar chord diagram charts for songs.  It was
    written to help students in high school jazz band quickly play all of
    the common extended, altered, and complex chord types.  It can also
    be used to create chord charts for any other type of music (with
    simpler chords):  folk, rock, blues, pop, etc.
```

To select chords for your song, click the root note (letter name: A, Bb, C#, etc.), and then the sonority (major, minor, 7(#5b9), etc.) of each chord. The list of chords you've selected will be shown in the text area below. When you've added all the chords needed to play your song, click the "Create Chart" button. Your browser will open, with a complete graphic rendering of all chords in your song. You can use your browser's page settings to print charts at different sizes.

Two versions of each chord are presented: 1 with the root note on the 6th string, and another with the root note on the 5th string. Chord lists can be saved and reloaded with the "Save" and "Load" buttons. The rendered images and the HTML that displays them are all saved to the "./chords" folder (a subfolder of wherever this script is run). You can create a zip file of all the contents of that folder to play your song later, upload it to a web server to share with the world, etc.

-- THEORY --

Here are the formulas and fingering patterns used to create chords in this program:

6th string notes:

```
0  1  3  5  7  8  10  12
E  F  G  A  B  C  D   E
```

5th string notes:

```
0  2  3  5  7  8  10  12
A  B  C  D  E  F  G   A
```

The sharp symbol ("#") moves notes UP one fret
The flat symbol ("b") moves notes DOWN one fret

Root 6 interval shapes:

```
_________
| | | | 4 |
| 3 6 9 | 7
1 | | | 5 1
| | 7 3 | |
| 5 1 4 6 9
| | | | | |
| | 9 | 7 |
```

Root 5 interval shapes:

```
_________
| | | | | |
| | | | | |
| | | | 1 4
| | 3 6 | |
5 1 4 | 9 5
| | | 7 | |
| | 5 1 3 6
```

To create any chord, slide either shape up the fretboard until the number "1" is on the correct root note (i.e., for a "G" chord, slide the root 6 shape up to the 3rd fret, or the root 5 shape up to the 10th fret). Then pick out the required intervals:

CHORD TYPE:	INTERVALS:				SYMBOLS:
Power Chord	1	5			5
Major Triad	1	3	5		none (just a root noot)
Minor Triad	1	b3	5		m, min, mi, -
Dominant 7	1	3	(5)	b7	7
Major 7	1	3	(5)	7	maj7, M7, (triangle) 7
Minor 7	1	b3	(5)	b7	m7, min7, mi7, -7
Half Diminished 7	1	b3	b5	b7	m7b5, (circle with line) 7
Diminished 7	1	b3	b5	bb7 (6)	dim7, (circle) 7
Augmented 7	1	3	#5	b7	7aug, 7(#5), 7(+5)

```
    Add these intervals to the above 7th chords to create extended chords:

    9 (is same as 2)   11 (is same as 4)   13 (is same as 6)

    Examples:              9          =    1   3  (5)  b7    9
                           min9       =    1  b3  (5)  b7    9
                           13         =    1   3   5   b7   13
                           9(+5)      =    1   3  #5   b7    9
                           maj9(#11)  =    1   3  (5)   7    9  #11


    Here are some more common chord types:

    "sus"       =  change 3 to 4
    "sus2"      =  change 3 to 2
    "add9"      =  1 3 5 9  (same as "add2", there's no 7 in "add" chords)
    "6,  maj6"  =  1 3 5 6
    "m6, min6"  =  1 b3 5 6
    "6/9"       =  1 3 5 6 9
    11          =  1 b7 9 11
    "/"         =  Bassist plays the note after the slash


    NOTE:  When playing complex chords (jazz chords) in a band setting,
    guitarists typically SHOULD NOT PLAY THE ROOT NOTE of the chord
    (the bassist or keyboardist will play it).  In diagrams created by
    this program, unnecessary notes are indicated by light circles, and
    required notes are indicated by dark circles.
}
root6-shapes: [
    "." "major triad, no symbol (just a root note)" [1 3 5 11 55 111]
    "m" "minor triad, min, mi, m, -" [1 b3 5 11 55 111]
    "aug" "augmented triad, aug, #5, +5" [1 3 b6 11 111]
    "dim" "diminished triad, dim, b5, -5" [1 b3 b5 11]
    "5" "power chord, 5" [1 55]
    "sus4" "sus4, sus" [1 4 5 11 55 111]
    "sus2" "sus2, 2" [1 99 5 11]
    "6" "major 6, maj6, ma6, 6" [1 3 5 6 11]
    "m6" "minor 6, min6, mi6, m6" [1 b3 5 6 11]
    "69" "major 6/9, 6/9, add6/9" [1 111 3 13 9]
    "maj7" "major 7, maj7, ma7, M7, (triangle) 7" [1 3 5 7 11 55]
    "7" "dominant 7, 7" [1 3 5 b7 11 55]
    "m7" "minor 7, min7, mi7, m7, -7" [1 b3 5 b7 11 55]
    "m7(b5)" "half diminished, min7(b5), (circle w/ line), m7(-5), -7(b5)"
        [1 b3 b5 b7 11]
    "dim7" "diminished 7, dim7, (circle) 7" [1 b3 b5 6 11]
    "7sus4" "dominant 7 sus4 (7sus4)" [1 4 5 b7 55 11]
    "7sus2" "dominant 7 sus2 (7sus2)" [1 b7 99 5 11]
    "7(b5)" "dominant 7 flat 5, 7(b5), 7(-5)" [1 3 b5 b7 11]
    "7(+5)" "augmented 7, 7(#5), 7(+5)" [1 3 b6 b7 11]
    "7(b9)" "dominant 7 flat 9, 7(b9), 7(-9)" [1 3 5 b7 b9]
    "7(+9)" "dominant 7 sharp 9, 7(#9), 7(+9)" [1 111 3 b77 b33]
    "7(b5b9)" "dominant 7 b5 b9, 7(b5b9), 7(-5-9)" [1 3 b5 b7 b9]
    "7(b5+9)" "dominant 7 b5 #9, 7(b5#9), 7(-5+9)" [1 3 b5 b7 b33]
    "7(+5b9)" "augmented 7 flat 9, aug7(b9), 7(#5b9)" [1 3 b6 b7 b9]
    "7(+5+9)" "augmented 7 sharp 9, aug7(#9), 7(#5#9)" [1 3 b6 b7 b33]
    "add9" "add9, add2" [1 3 5 999 55 11]
    "madd9" "minor add9, min add9, m add9, m add2" [1 b3 5 999 55 11]
    "maj9" "major 9, maj9, ma9, M9, (triangle) 9" [1 3 5 7 9]
    "maj9(+11)" "major 9 sharp 11, maj9(#11), M9(+11)" [1 3 7 9 b5]
    "9" "dominant 9, 9" [1 3 5 b7 9 55]
    "9sus" "dominant 9 sus4, 9sus4, 9sus" [1 4 5 b7 9 55]
    "9(+11)" "dominant 9 sharp 11, 9(#11), 9(+11)" [1 3 b7 9 b5]
```

```
    "m9" "minor 9, min9, mi9, m9, -9" [1 b3 5 b7 9 55]
    "11" "dominant 11, 11" [1 b7 99 44 11]
    "maj13" "major 13, maj13, ma13, M13, (triangle) 13" [1 3 55 7 11 13]
    "13" "dominant 13, 13" [1 3 55 b7 11 13]
    "m13" "minor 13, min13, mi13, m13, -13" [1 b3 55 b7 11 13]
]
root6-map:  [
    1 20x70 11 120x70 111 60x110 3 80x90 33 40x50 b3 80x70 5 100x70
    55 40x110 b5 100x50 7 60x90 b7 60x70 9 120x110 99 80x50 6 60x50
    13 100x110 4 80x110 44 100x30 999 60x150 b77 100x130 b33 120x130
    b9 120x90 b6 100x90 b55 40x90
]
root5-shapes: [
    "." "major triad, no symbol (just a root note)" [1 3 5 11 55]
    "m" "minor triad, min, mi, m, -" [1 b3 5 11 55]
    "aug" "augmented triad, aug, #5, +5" [1 3 b6 11 b66]
    "dim" "diminished triad, dim, b5, -5" [1 b3 b5 11]
    "5" "power chord, 5" [1 55]
    "sus4" "sus4, sus" [1 4 5 11 55]
    "sus2" "sus2, 2" [1 9 5 11 55]
    "6" "major 6, maj6, ma6, 6" [1 3 55 13 11]
    "m6" "minor 6, min6, mi6, m6" [1 b3 55 13 11]
    "69" "major 6/9, 6/9, add6/9" [1 33 6 9 5]
    "maj7" "major 7, maj7, ma7, M7, (triangle) 7" [1 3 5 7 55]
    "7" "dominant 7, 7" [1 3 5 b7 55]
    "m7" "minor 7, min7, mi7, m7, -7" [1 b3 5 b7 55]
    "m7(b5)" "half diminished, min7(b5), (circle w/ line), m7(-5), -7(b5)"
        [1 b3 b5 b7 b55]
    "dim7" "diminished 7, dim7, (circle) 7" [1 b33 b5 6 111]
    "7sus4" "dominant 7 sus4, 7sus4" [1 4 5 b7 55]
    "7sus2" "dominant 7 sus2, 7sus2" [1 9 5 b7 55]
    "7(b5)" "dominant 7 flat 5, 7(b5), 7(-5)" [1 33 b5 b7 111]
    "7(+5)" "augmented 7, 7(#5), 7(+5)" [1 33 b6 b7 111]
    "7(b9)" "dominant 7 flat 9, 7(b9), 7(-9)" [1 33 5 b7 b9]
    "7(+9)" "dominant 7 sharp 9, 7(#9), 7(+9)" [1 33 b7 b3]
    "7(b5b9)" "dominant 7 b5 b9, 7(b5b9), 7(-5-9)" [1 33 b5 b7 b9]
    "7(b5+9)" "dominant 7 b5 #9, 7(b5#9), 7(-5+9)" [1 33 b5 b7 b3]
    "7(+5b9)" "augmented 7 flat 9, aug7(b9), 7(#5b9)" [1 33 b6 b7 b9]
    "7(+5+9)" "augmented 7 sharp 9, aug7(#9), 7(#5#9)" [1 33 b7 b3 b6]
    "add9" "major add9, add9, add2" [1 3 5 99 55]
    "madd9" "minor add9, min add9, m add9, m add2" [1 b3 5 99 55]
    "maj9" "major 7, maj9, ma9, M9, (triangle) 9" [1 33 5 7 9]
    "maj9(+11)" "major 9 sharp 11, maj9(#11), M9(+11)" [1 33 b5 7 9]
    "9" "dominant 9, 9" [1 33 5 b7 9]
    "9sus" "dominant 9 sus4, 9sus4, 9sus" [1 44 5 b7 9]
    "9(+11)" "dominant 9 sharp 11, 9(#11), 9(+11)" [1 33 b5 b7 9]
    "m9" "minor 9, min9, mi9, m9, -9" [1 b33 5 b7 9]
    "11" "dominant 11, 11" [1 b7 9 44 444]
    "maj13" "major 13, maj13, ma13, M13, (triangle) 13" [1 3 55 7 13]
    "13" "dominant 13, 13" [1 3 55 b7 13]
    "m13" "minor 13, min13, mi13, m13, -13" [1 b3 55 b7 13]
]
root5-map:  [
    1 40x70 11 80x110 111 100x30 3 100x110 33 60x50 b33 60x30 5 120x70
    55 60x110 b5 120x50 7 80x90 b7 80x70 9 100x70 6 80x50 13 120x110
    4 100x130 44 60x70 444 120x30 99 80x150 b3 100x90 b9 100x50 b6 120x90
    b66 60x130 b55 60x90
]
root6-notes:  [
    "e" {12} "f" {1} "f#" {2} "gb" {2} "g" {3} "g#" {4} "ab" {4}
    "a" {5} "a#" {6} "bb" {6} "b" {7} "c" {8} "c#" {9} "db" {9} "d" {10}
    "d#" {11} "eb" {11}
]
root5-notes: [
```

```
    "a" {12} "a#" {1} "bb" {1} "b" {2} "c" {3} "c#" {4} "db" {4}
    "d" {5} "d#" {6} "eb" {6} "e" {7} "f" {8} "f#" {9} "gb" {9} "g" {10}
    "g#" {11} "ab" {11}
]
f: copy []
for n 20 160 20 [append f reduce ['line (as-pair 20 n) (as-pair 120 n)]]
for n 20 120 20 [append f reduce ['line (as-pair n 20) (as-pair n 160)]]
fretboard: to-image layout/tight [box white 150x180 effect [draw f]]
; spacer: to-image layout/tight [box white 20x20]
view center-face layout [
    across
    t1: text-list 60x270 data [
        "E" "F" "F#" "Gb" "G" "G#" "Ab" "A" "A#" "Bb" "B" "C" "C#" "Db"
        "D" "D#" "Eb"
    ]
    t2: text-list 330x270 data extract/index root6-shapes 3 2 [
        either empty? a/text [
            a/text: rejoin [
                copy t1/picked " "
                pick root6-shapes ((index? find root6-shapes value) - 1)
            ]
        ] [
            a/text: rejoin [
                a/text newline copy t1/picked " "
                pick root6-shapes ((index? find root6-shapes value) - 1)
            ]
        ]
        show a
    ]
    return
    a: area
    return
    btn "Create Chart" [if error? try [
        make-dir %chords
        delete/any %chords/*.*
        ; save/bmp %./chords/spacer.bmp spacer
        html: copy "<html><body bgcolor=#ffffffff>"
        foreach [root spacer1 spacer2 type] (parse/all form a/text " ") [
            diagram: copy [image fretboard]
            diagram2: copy [image fretboard]
            root1: copy root
            foreach itvl (third find root6-shapes type) [
                either find [1 55] itvl [
                    append diagram reduce [
                        'fill-pen white 'circle (select root6-map itvl) 5
                    ]
                ] [
                    append diagram reduce [
                        'fill-pen black 'circle (select root6-map itvl) 5
                    ]
                ]
            ]
            append diagram reduce ['text (trim/all join root1 type) 20x0]
            append diagram reduce [
                'text
                trim/all to-string (
                    select root6-notes trim/all to-string root1
                )
                130x65
            ]
            save/png
                to-file trim/all rejoin [
                    %./chords/ (replace/all root1 {#} {sharp}) type ".png"
                ]
```

```
            to-image layout/tight [
            box white 150x180 effect [draw diagram]
        ]
        append html rejoin [
            {<img src="./}
            trim/all rejoin [
                replace/all copy root1 {#} {sharp} type ".png"
            ]
            {">}
        ]

        foreach itvl (third find root5-shapes type) [
            either find [1] itvl [
                append diagram2 reduce [
                    'fill-pen white 'circle (select root5-map itvl) 5
                ]
            ] [
                append diagram2 reduce [
                    'fill-pen black 'circle (select root5-map itvl) 5
                ]
            ]
        ]
        append diagram2 reduce ['text (trim/all join root type) 20x0]
        append diagram2 reduce [
            'text
            trim/all to-string (
                select root5-notes trim/all to-string root
            )
            130x65
        ]
        save/png
            to-file trim/all rejoin [
                %./chords/ (replace/all root {#} {sharp})
                type "5th.png"
            ]
            to-image layout/tight [
            box white 150x180 effect [draw diagram2]
        ]
        append html rejoin [
            {<img src="./} (trim/all rejoin [
                replace/all root {#} {sharp} type "5th.png"
            ]) {">}
            ; {<img src="./spacer.bmp">}
        ]
    ]
    append html [</body></html>]
    write %./chords/chords.html trim/auto html
    browse %./chords/chords.html
] [alert "Error - please remove improper chord labels."]]
btn "Save" [
    savefile: to-file request-file/file/save %/c/mysong.txt
    if exists? savefile [
        alert "Please choose a file name that does not already exist."
        return
    ]
    if error? try [save savefile a/text] [alert "File not saved"]
]
btn "Load" [
    if error? try [
        a/text: load to-file request-file/file %/c/mysong.txt
        show a
    ] []
]
btn "Create Zip" [
```

```
if not exists? %chords/ [alert "Create A Chart First" return]
; rebzip by Vincent Ecuyer:
do to-string to-binary decompress 64#{
eJztW+uP20hy/+6/oldGsJ7bcEU2X00Zd4bX9iILXO4AY5N8EOYAjkTNMNaQOomy
PTbmf8+vqptkU3xIM4cgARICHmvYVdX1fnRrPn745a9/FsvrFy9W1VfnW75biFVZ
VNnXSixfCDyr/crZlsXtwvzeeVz885IkSsJEJYEQju96bhCHrhKOF8s4CGToSgKS
QeQHnh/jo1KRG8aRFzb0HD9O/CCMPA+fvciPwziUEX4RMkhkpEAIH90gAGFf0j6h
lCqUbhTRPkEcY3dftvx5kUwCMBYSCU+GIO1KIh64sR+5oAo8FUWhBDnwJIJEhX4U
Bgq4IJbIOIqilj/phknoRrSMz1GcuJGrSC5wBCRFvHoKLISQzSchVOwr0PMJHjyr
JArjlj8CVH6SsP4iBQFiLyR6TuCrIJRhABJJ7LsBRCQg3w8j3w8U6S+KPGI1avUn
HNggCsKA9ZdIpUJgJ1hIkhgSu0lEJlBeEvpgFkDYzvO8JKJ9wtiVkCgMGnqe60ZK
KS8I2XQkVBzR3k4cuxArimLCk6RJ6I32gd2TOJaQgei6RKC150Bb0JRL8jqeD39x
vTCCMgWwYdYgJP3BYSRkc0mIEDYIvdCnfQJP+lC+77XyRqEXwwqK9efHWPahENID
VOdJ2A94xHMko4T2CaMwdGNJvEKqIJJebNFzlAv7h2HC+oOWwFWSkFxR7IGnhPzP
C8mllMv+l4BXP/LZ/+BCkZcElrwiTkLlh6Ek1/IC2N93pQyJJwgC5cAXHUgVKC8G
HxxDgUdUJHuH54VQTEPPIWbJA2lrOBR4ChLaG3huHICviA0FVv1YerRPhF29wA3I
F32yrSv91v+kC6XFiBGXfwtJ867PsYQHvkN2dPA/wGQYUQz5McweRz4LRL8pz2/5
g/IilYRxzC4sOSEkCcUERE1CWJhihewRKNIZ4ilCRMBTSR4J51CBp1r/U1BO7MKY
gmMM1pCkJ0d6HqJAQgO8qx968EyyaUwBqRLONT4JLmPX0h+CN4CTUIoiwYIkSOKI
bBrBt0PF9AQ0CgvAiwnBlyQF0Yb34iUiuPUXz4e/+WHCoSkSFSSkb9JfGMR4i1RF
sSIhYBKHimMoAo8wNwUXtAd2PK/lLyQ4KJi3iwOIgqDlmIDh4bBuRLEC0jHUyj6n
wlj6yqd9EBmsY7flDyaCjwQR2QMxBu1CZxxLgfQofkOOMWDCIsQTlBr6fkIJDsGF
gEx87Nky6LnIXyCpONmQTZFWKdgD5B0w5VKyAZ4HHn1yQGTDIPDBIUUXIgRJTXmi
5RCxhtiCY3K4wEt8BY/kaFIhJQ+OZuI2Voqjzw0SlA6XRIopuCjFWhyCHx+lIuTs
66K6yMAnYJQkpCvgUkpAMqCADtlukfTJb8g8cF3lIUdJYdUkbAG2Q12TAs6Yih0S
kQXnTEhvKDsKviBp10C5LmKdc4JykbvAsUVQIANQRfRdjkCsJ+QJhIjcmniUvYEo
FZiXigMNsuMfBbFEdkXSQ9RaHEqsgRUv4qqEihdHiUfUkQ6hfeQ7iowIe8HzyFNg
fqRBN+RIgpeBhcAyigigGiklhx1kRe2OFZcRBEQEhmOXPikZIf0Q50IidFA3JaWi
GC4ETwttHcLACAJPG4WqgkpYocJHeUKyIYdHrZYI5Yh6A4QatQYB9w+hQlFBeCuL
QxSwGLHic0/hB1iOeHeH0lmAHEE6pOB2kd042FDIoE1FwQS3jVD/wsTiMIopMwfs
bjC2knBu9nKUeKgAlYKIJFAg0iRnxQAMIF592gh5TmFjyygCFHyJxKkzMrIkJSny
IfBBbQqXJo96IRmw90ewODZyyQ996BaeEUUWh/BYdFcel1+PyiYeyeEWI4uiDHFN
g02gXS/k2gRXcWPuIZDEAAAci8PYRz1CeQ04DkPqtJKQnNbF5hQSFL+UPShwyFho
xqjIUGNDXurD16SVvOC9iuwScdQgUSJsdXFC4kQajbmoUeeFxEuGjdDt+DCdx1GG
foBSXJsaoDwXJLmYQJ9YhqOSXWNoB4mUTOJQQGMXlxtDMIi2hPIRVSrIg+xnpQbl
I/BhK86uCFgoyKVWE0EVkOZZSPg1jImmh2IJFoQpKO84ktw+QuWykleM9EgFjgVD
oUYUwru5PCnkyDgMOakjn8Nu3BrCi9F3UkoXAedMFNOWQ9Rj1CBASM0wQiDi2ujF
HrWHIduKOl4F0Ti/oByhpYkSTn2o2S6ylcUhmm1k/5gbYAHTkBlc0mGETgcGcqkT
g1DcuLtc0UARfazkIKPONbECz6HyFlAAkQQOJXzEBHsekij6IF2t0PpT4eUCAGFg
4iCmCg/fRGsCh7I6TPS/KI+xrqA+Eheyp8spCnUT/knujnQCD+Yy5lGCg4dziUQd
hVfDjnaFgmIQooorFNIMFVD2Qm4GYBKSDGWBLBLSRtQ6QHpu19F9INTdxLdUGEgq
skFAoSYirsKBz30bumvEleIeE70cQjIgFYZU0PyYUomA3WDAGA2bVaGgnASxy85B
WQfr3PyFQEMB5pQBRqkwetysYSoj1H1+e1c5h7t8UzlqITbHYmUNY7OP7fJBVHeZ
+Jxuj5m4eRBK3OR4lxZrsc+q4744iLz6edagasBljnnvNtv/cC1mvzfoVSmYpIa+
bvfLcuyxF0V2m1b55+xNTaUzHTqe+FruzRLt71AfORf0kxawTm8pFVDH2uBen5Ah
eALUhBi/hX3R/uQfx906rTIHE+uAkoz8pB9ApTTXWprgV8u0eHAO1T4vbkkX7+kd
1LC6y1afDsf7Fhq4Ha39VuRVnm41lz2Fbcp9lq7uQCfdm426kzQz/KpjZXp3xZra
5atPzRDOO5M+UGp5lWn+JLyeUprZ3ZfjykjFu4/vUEtrCZ+lkL5rZPe76uGNIfHy
u2uexxPjbst0Lf6zzAsxe/19JpZV6dxlX2vXaY2pCeH17HF2PSzo60N+W6QQKju8
0KRX6dbZ5NvMwcoCTBsxxMvvCONfXEydj1pFWVHtp2ERqxo2K9ZOuXFqlGFoNK8a
mrh21tlhtc93VbkfAcdQ8GhkWJXF52x/yMuCDaYlAUKuj19OjfiOwTm8hfFFMk4q
tnlVQRhwm2OJkCdD/t/rcNf79wMeLNzkRbp/QBJhc3XNiGX2wjbW4ZyDECSLYfRV
C42WU7lXJ7F9Ho2zZKQxUQn9S3HnddKJrgYzCWEc7sp99WyNM/b/PpWfS6EAfaKj
3eS3/7CXrcuBFKDRKdw1a7dZdYFRhszAjDZs9zm0EpzYAX6KVUuXd1lKJZXCNbOM
tSp3D/Ndim01eWmxf063A2H7P8t80LjF/WFdHpwqv88mBGCPYEhBkH1u6e2lzq+N
bUJ2flce9+IPdACsrjpeXtZRPb/Pi2OVAciXfZDTIJgfMuy9pli46gaAlpQKz2WS
EmRfUnr7RElptvyDeHWffhWuYfIhQ/w6NJe4olKXRXWnhRZ1tzVfpw+WUOR7F9lv
sy/vz1jwKf7WxV1YIazf2Dqg3eCQ6+PqtI3E0ICGvtMAwgk6IJidui2ifQ8g6DTQ
WoUTDea+VktnbN/R0rD1/7u0RLuNaCmkQ2lbB+ROP7HrdAAD5V6qqvEu+lu+c6gD
emjU9P1deb9D63XIKI9RJ9UZOSxmuS/TEJSFMupdVzXymhdaYNNnafB1vs9WaKMe
BO+toa4fG+gihQ8tCZT0/Rf6rdwwaqentaLzX8t1vslXGGLQcPHSEMJ4E2zY1tBz
lqwzINxn1V25tsRzmBxU1r455N8ycfL7qWu8RgHYlEIziaKb0TzRIIF5e9dFN7vr
AYARG6geA4umyRTbrLit7nTr/mKyre84jhZ1IX48wEKW63ShXotv2/xGQJVZet9Z
OdHRonnR5dxoY2MLD+VkG5gxh1v8s+D92xkvLw4V1NEhYIR5ZYt6Jf7Uvjhh5mrg
```

```
        3rNm0hTM3jo9h0/5rmd8OQ5apfm2B++oHkKj63W22cJrOwDXfV5HbNOofktl5mTf
        Lk0rFYw5Zbqpsv2wTz7R1wYT3Gs7cVjxbx5uIF9+fxywVHcQ7C3r4fQRG5jJawIC
        2r499NaNO5lg/6NR89BdeYdYAz9slQE7Gnx1gj/kBUyh96bTyHHCm9OncTjOiQy3
        HtqBchw9r02W6QMM5DX76SWiIVZMNaydhdP8a3YE/qjfT1gt+4oassmz7XoMlumw
        HDXVQd1rkKK0KA7mBaOStqhNBJSBNecJKHHP8PDTA4zzPi4O6KdXfSn7gOiq/z8g
        /s8FhGVhUAVL93Cx88Dr/PCpON7fwPiHaqgqtqA0je0LJPS0Ql91g8mt70fWuaHm
        +okYpr8sN5tD1uflSUE/EfEtkNHTaOnUP/k/aihp2zfjJ7O/f/y3D3Rw3HbTx705
        Xz78PLNHjephh3bWNI14oZvg+lQeWObjtdkcLrRLqztmoLf/92bQAeIP1OZyR23a
        +fvyMzp8UiwdYM7ms7b9NgxpaMK9ZOjJdb89fIOAxZfYAhG9yfeH+lBiaQYl6r1E
        Qd9S08J1UPXgMTjHNLwukChW5TpzGr0265xs/2k2O2GoPosAL+Crs8Rq1+ukWzGb
        XY+tIzIw5HUgOr4BZxU/YsRqZqtfjvl2TXMVvUz3q7v8c1aPoewccBJMBWh0PtUD
        zOFnTbt2JhOVWKVakcOGENCQ+rm14Rfk7Y4NhT4Z+kG0089/MAxsekNsibxqFaGr
        ShefuCK0X4krQsuL1fa4ziwGWgLzdZbt6DKFQQ5aFII8HG/q8Q/MWwjwqpsSY87s
        z/mhqhG+3JFWml60uB2c0DgBcL3maiuKG6dWjiZTIOvDP9o0Y5VpM4kyoJ7hymMl
        mqJguTje10GmHbWuj00OqNXecZglJhdEjFk7cfGTWrhkoIWYRLFjrre1IVDusmKu
        pZtry8+/7HOUPF63mv9W7oWlt4Vwx3VlIhYtzIsTh9HDHllx+Uq/ubq2XJJsuSzK
        iuejNwbnRANky4VZmnvdJWPFhY5BTofa9Plef+pGKotmgHvJPt1uxbJBNNvid3IA
        frOn2ZuJLrtkT1u+/KARUkGi8WDdnnC8GZpX9d4UIA3GG83t9QCnr0W6gzXX+ru8
        RVVncrFFoAwWsfqSkqEaOUbaNnqMu/HYakKfMOaL7iHOoAIGujnbj6xYtK8260cf
        0RXllxcDcpNmqSyUhQlvU0OJtzdDpDDItyEpltrNmISRqg0X42FypBk1ZEzV2qYo
        FFqHteuP4LE2N2zX2l0n9N5qgA8tGZq3+fHeOtByBjvU+un3wvVDljfBL0a7oj76
        iGS6ctdxbin1gu5cR5fdrQwzBM3VZWC5Q5mqBqKantekr9PDRvjX4MGi9ZijzubU
        s6kcWX+y0/vopKlrM3efVrXrzwR3aXGb6SMgpj+XIpDtKUmbawd2Ose7HaP98mW2
        G6Db15LupnugVPY0mX6Y2lXCqpQ/NaPLEOJpomSfQcvUeT3lUC0B4m1kbu7poptx
        e9f8neXuxEPN1dRyfYiNrTpg7SRnZTvIe9P0aIT+NJQedOecbVrocw7XisXNqWmx
        piZDevrQwy0J3ZM07chqS8Fsmo4aqBW51y0fC7tffp+t7NsIu2vmi7vW/9vG2fTL
        fy22D+JLuf+EJhK5vHPMrM8XDvVxBg1E9Qms1ULTo6Xo9MFUQupe2Oqh1w2rnUaa
        Bes3091riLetTC2ZLo2p9rjFoVHuFSRJj9vq6oTA3485zSrvy+LHiluHCSKDXTYf
        EQ3chHAmGD/uoBTraLfS6badwOvXXNubg5sGs0OTYaxY4dmMnZEPr7P9vtwfTjt2
        /dZuZgl4pImnpkwriaq30fdpf16gFxnv3qlmma9AzJ7VtpsG9CReCIG4O8Xi7vEr
        mdK8OeHnPv2UUYrQwxhDnGOn9tU6Lds9hOnXB8KYNNy83qX7QzYnhmtiJ8lp6iD/
        UA42qtXd/nju/P/VeNt51QOWukqP3xVwn6ddXsMO5sRXptf7lu3LNxp8br4NKZbm
        6GKTbuFHfRZ4BxNOU1vUlz7WfbJ+NUKSA8nI90p/UaD7xQGGGEHmXqhGHuTH6le7
        B6mj0HPNxOB5bEuR6Tg0Csy/Ich6kH2Gg9qGo0e0u4deogomNN2732q+7HNKZkR7
        /fvgqe0GLm/bDXuLI1va6XXKiyy4jitZ70d26OfrqX160J3deqsTUnVEm3DIZrgY
        nyvo4TIxuNrnoS/yoMDd++3a5IOgw6wf0Jqs7uamYtfJYHxedM/Mkr2KuZgEp6f9
        ppj+JvWZSwv7YY3O/uYI50+i/ISaQd3U9d9OzlPtZ3xWVeen5Ha419m2H2zTJGpp
        +eBqCmhitK+fM9/pHHteflf014sfgviRmvD37lvvrfuWGvHmO88X0jGXI+8J97d/
        ef9xqnE/fWaEMAMiXTcsxsbBoQfzcL5GNPyxo3xx1lPqx3wt9YLLsaGnkdrTd0L0
        HXuw4qGbHagXIxQU4d7kcPdsd6G+aN9HnqS3GDNIadj1dp8+HNCPXM76ozWMt1PI
        k9CR4Sr7PrYYKpRTBIpS39dt04FzmzHE9h6uaP9e4EJ/eYJb1s5B33aLOt8lu9xF
        Zr+9f/t769qDhzpDz8vvsUrePdZfa3pSSLhsiNo2dM8Hvyie4pOuoD+OFL9++PVX
        AUVfhPYP5G175+fbtov9pPTz4S/vWxtlxeUx8Dxux6uOXp1crr8vR5PkG0Fnhktd
        i/gvXujjBeXCrpSbFG3Kms7/+cBhsl42TJg5Vv8/eJB++vRbgfPZ4gJJntFh0EOT
        aHMD+fTsv892WVqJ/Fk1nx778LQnwkUU6Kn/EIP/jIt/6G8mXmCOaR/ULHa7qUv9
        Y5zy8MrUxQWxsDzxUlNsZnXVmRFH45s+wVWf4qLDG1r3r3wkN3wBaz9D96uT3xgZ
        QrBPjCZtc3p9NJ1p9D3g5e55rU+jlj2MJxv+tQ7Q9jg15b+oQClr7z/GLT52VTY9
        PGz6x1fzxQWI9AwcbGnc50bK9JZ0Mr1sDhyvxWG3zSu+yzpv1WE5GfeZchLuM+U0
        DPUPVs+zwo5QnwReqvH6eXrKPTT3oU/zDHp6d6eL8fMp+znXJzxtpf+2e87QXa/K
        TiNknc6anHxSBfS3cObvc+TYY2EuRdYLpGnr5LOP1V6o0ARS5+pZna07CC0Pet62
        j9hfoGv6L0cgfyluTAAA
    }
    zipfile: to-file request-file/file/save %/c/mysong.zip
    if exists? zipfile [
        alert "Please choose a file name that does not already exist."
        return
    ]
    zip/deep zipfile %chords/
    ]
    btn "Help" [editor help]
]
```

```
REBOL [title: "PDF Barcode Generator"]
text-string: "item2342"
x-offset: 10
y-offset: 257
create-pdf-barcode: func [barcode-string xshift yshift] [
    barcode-width: .3  barcode-height: 12
    code39: first to-block decompress #{
    789C5D93490EC2400C04EF794514C10504D8EC1CD9F77D07F1FF6F30C9C4E3F6
    200529E54EA91D866F92BA4FC699BB989828FF6277EB793BE7EE3EE69D322F03
    E15D9F27629BEFA9DFE4FBEA377C103CC520F021F684FC087B0227EC037C2C9E
    F209E113F1447C1AF6F503E1B3D2CF517E1EFC36BF087ECB97E221BBEF0A7B42
    7E8D3D816FB00FF0AD7A8A89F09D7A0CDFC3BEF940F841FD267F847D317F827D
    919FC3BE6C3C17E889F92BF4447E833EC8EFDE43A212FE28F2C4317F4A9EED79
    7E95F9F83CBFD56FF21FF51BDE081EFBFB36B127E453EC09BC867D80578447E7
    B3051CDF4F5DFB185ED5FF9DE7C9EF0F6518AA1B22040000
    }
    convfrom: rejoin ["*" barcode-string "*"]
    pdf-dialect-out: copy []
    x: 0
    foreach char convfrom [
        pattern: select code39 form char
        foreach bit pattern [
            x: x + 1
            if bit = #"1" [
                append pdf-dialect-out compose [
                    line width (barcode-width)
                    line
                    ((x * barcode-width) + xshift) (yshift)
                    (
                        (x * barcode-width) + xshift
                    ) (yshift + barcode-height)
                ]
            ]
        ]
        x: x + 1
    ]
    return pdf-dialect-out
]
do load-thru http://www.colellachiara.com/soft/Misc/pdf-maker.r
barcode-layout: copy []
current-barcode-page: copy [page size 215.9 279.4 offset 0 0]
append current-barcode-page create-pdf-barcode text-string x-offset y-
offset
append current-barcode-page compose/deep [
    textbox
    (x-offset - 9.5) (y-offset - 8)
    56 8
    [
        center font Helvetica 3
        (mold text-string)
    ]
]
append/only barcode-layout current-barcode-page
write/binary %labels.pdf layout-pdf barcode-layout
call %labels.pdf
editor barcode-layout


REBOL [title: "VID Shooter"]
score: 0   speed: 10   lives: 5   fire: false   random/seed now/time
alert "[SPACE BAR: fire] | [K: move left] | [L: move right]"
do game: [
    view center-face layout [
        size 600x440
```

```
        backdrop black
        at 246x0 info: text tan rejoin ["Score: " score " Lives: " lives]
        at 280x440 x: box 2x20 yellow
        at (as-pair -50 (random 300) + 30) y: btn 50x20 orange
        at 280x420 z: btn 50x20 blue
        box 0x0 #"l" [z/offset: z/offset + 10x0 show z]
        box 0x0 #"k" [z/offset: z/offset + -10x0 show z]
        box 0x0 #" " [
            if fire = false [
                fire: true
                x/offset: as-pair (z/offset/1 + 25) 440
            ]
        ]
        box 0x0 rate speed feel [
            engage: func [f a e] [
                if a = 'time [
                    if fire = true [x/offset: x/offset + 0x-30]
                    if x/offset/2 < 0 [x/offset/2: 440  fire: false]
                    show x
                    y/offset: y/offset + as-pair
                        ((random 20) - 5) ((random 10) - 5)
                    if y/offset/1 > 600 [
                        lives: lives - 1
                        if lives = 0 [
                            alert join "GAME OVER!!! Final Score: " score
                            quit
                        ]
                        alert "-1 Life!"   unview   do game
                    ]
                    show y
                    if within? x/offset (y/offset - 5x5) 60x30 [
                        alert "Ka-blammmm!!!"
                        score: score + 1   speed: speed + 5  fire: false
                        unview   do game
                    ]
                ]
            ]
        ]
    ]
]


REBOL [Title: "Collect the Boxes"]
random/seed now
start-time: now/time
level: to-integer request-text/title/default "Number of Boxes?" "10"
gui: [
    size 600x440  backdrop white
    at -99x0 key keycode [up]    [p/offset: p/offset + 0x-10 show p]
    at -99x0 key keycode [down]  [p/offset: p/offset + 0x10  show p]
    at -99x0 key keycode [left]  [p/offset: p/offset + -10x0 show p]
    at -99x0 key keycode [right] [p/offset: p/offset + 10x0  show p]
    at -99x0 box rate 0 feel [engage: func [f a e][if a = 'time [
        foreach f system/view/screen-face/pane/1/pane [
            if  f <> p [
                if within? (f/offset + 10x10) p/offset 30x30 [
                    remove find system/view/screen-face/pane/1/pane f
                    show system/view/screen-face/pane/1/pane
                ]
            ]
            if (length? system/view/screen-face/pane/1/pane) < 8 [
                alert rejoin ["Your time: " now/time - start-time]
                quit
            ]
```

```
        ]
    ]]]
]
for counter 1 level 1 [
    append gui [at random 590x420 box 10x10 random 255.255.255]
]
append gui [p: btn red 20x20]
view center-face layout gui


REBOL [title: "Animated GIF Example"]
anim-frames-block: [64#{
eJxz93SzsEwMZwhn+M4AAg1g3ACmGsCsBjDnPxj/B1P/waz/YM4oGAXDBij+ZAHT
OiAClCfYOf4zCHLIeGxYcLCZQ1gr5sSGhYfbBZS95nhsXHS0W8I4686JjYuP9ys4
d8l4blpycrJG8KqYk5uWnp5ukHxqjufmZWdnWxS/OsPJcODoPLtKprUcW9TPLXJT
V7LdFZIZuMx/Ll/rrJCkC3NZ1ztd6SpVCG+L363EsXpCTvhmtovzVCWurr7R6jG7
rzZarKFpd8XTS77Z1/Xu7Qn+vunr6+/v725rqv6nm/Oj4Or2Ll7jvDUOa8+e6FX3
3uYjbPz0fN/RKjbeWcU+Z5do2qfN2lWaelnXfbveKwkz7ytLqu0qBK6Xed1cyfhG
TC58xeujhyuF422FXxQeOPybbR1nzbbP18+khtXvu/H95Ns7Gzdv5ZtfaVX64fjZ
crf/d6xPvV7XmJ7PZ1/x/ueXm/nXrOfVZKyZ+DL8nt85zhWzqu8LPosvPyYZEdW8
QrJjvjdj3TOFJuXQFVEVEl0iC9L49pVJJvZcnR7XLn/w+ux64XUpizrvbF0R1PFx
4QvB3s29OxLylB9tW9Cj9+vEol5NLk+5ia7vLB74GvxbETxZRklSqI+HyWNpR7ri
VbkJtreOp05nF1O/EeGW9C01/RqjmVrF3l7PZxnfPStv12qxsjBYAwBolvDW2AQA
AA==
} 64#{
eJxz93SzsEwMZwhn+M4AAg1g3ACmGsCsBjDnPxj/B1P/waz/YM4oGAXDBij+ZAHT
OiAClCfYOf4zMHLIeGxYcLCZQ1gr5sSGhYfbBZS95nhsXHS0W8I4686JjYuP9ys4
d8l4blpycrJG8KqYk5uWnp5ukHxqjufmZWdnW6hqBUwQfnxuvkPltxaJLSsuLOTt
ZWPdIPzSaal3vZUth6nWhZUsq7NsrUqzQ9f47K17qyWmdW1T2txFsreLdW/Pydu6
rXe2mHrsYuf3j86uLn95Z1/Qf6ZnWeUGD2e38V/3WVOh9viYkfzh3Fvmb1Iap+oq
P7OUKH64ocH2tsisGfkvTy7nXi6nG/n11dGZzLv3RQt8On3c19zY7e8stbyDCxtf
h0rLZBZuKjyYFrv6jsLdZ8xr99lGi3wueRLuGN6+zqSq7MW1700y/hHle4o/PhP8
5Xt+397f3z88Pj3ff/++v79/vGdnYbAGAJfEqNM/BAAA
} 64#{
eJxz93SzsEwMZwhn+M4AAg1g3ACmGsCsBjDnPxj/B1P/waz/YM4oGAXDBij+ZAHT
OiAClCfYOf4zMHLIeGxYcLCZQ1gr5sSGhYfbBZS95nhsXHS0W8I4686JjYuP9ys4
d8l4blri2cIVNC+GU2Hp6elcEX0tnsbLfPpNs++9mTE57fRcyepfJZxfFgUsdNWU
s51l8ihoma+8XatU6cOQVaHCca6zQh+GrYvlrWOVnvbgxrzUo/POzrz2JmpuLuu+
VuntT+9ML316T3VWuf79HXX/t/GuKTJIPBj5UW7bzB0fko75frwVGzP1ffIRa934
tpiQp88O9Zq3q84pL3qwq593uZ621dus61NCJ097K/714b7l3tf1bAv03jfNmv/v
264t3wu2Hn0r9973y6uiy2aql235hJeef35hovexONmK8jc3rzapXLeL03r+6cXl
1fHn9+39/f3D49Pz/ffv+/v7x+fX98/v3////1NWFgZrALxatNdHBAAA
} 64#{
eJxz93SzsEwMZwhn+M4AAg1g3ACmGsCsBjDnPxj/B1P/waz/YM4oGAXDBij+ZAHT
OiAClCfYOf4zMHLIeGxYcLCZQ1gr5sSGhYfbBZS95nhsXHS0W8I4686JjYuP9ys4
d8l4blpycrJG8KqYk5uWnp5ukHxKZMWCZWdnSuW+urOSId11nkP+rx6JLS8C2l0n
y6XO2PLyUovvXDtTCdNXV5pCl8YtnRn68tq6qOVNX6tKdW4uT+ud5sv9RTt6Xt79
Vz3a4Stu7Cq7+OitZ/i7i3tza5n4tCo+3JzWdniTz5oI1cfHNOVXt2pWqp87VaPv
LZf1413C3s7pdmKys0rSL88PZGbbe+vzva1rY3+/PV32+sCubRtnnd0rkJdwj/0h
0wyemh2p644UC7f17H778NGh3vO6fKbGX1/f2Jx9/9ze3d/fPzjczSvvv2/Pz88v
Lq+Oj7dTYLAGANdbpyswBAAA
} 64#{
eJxz93SzsEwMZwhn+M4AAg1g3ACmGsCsBjDnPxj/B1P/waz/YM4oGAXDBij+ZAHT
OiAClCfYOf4zMHLIeGxYcLCZQ1gr5sSGhYfbBZS95nhsXHS0W8I4686JjYuP9ys4
d8l4blpycrJG8KqYk5uWnp5ukHxqjufmZWdnWxS/unNy8/Lz8x2auWR/BTVeXOwi
Khe7y2Sl47KAiVamXApZV5b4rnWSXbVVO3RB3OF/PN7X1G9usjnfdXdl2dpz2/IK
D339VZZ3fVfZ2kdnd5uqx++t+/9tqvaMlWfXh3IrT7sZ/jHxaHim0zWtSqOnM6a9
FDtbU26cfkDPvrlNc1dm6kVTb22Lv5alaYfm5C+qu3OrNPfa+tzj13Ijv+XemZzI
zv9n+oq7Kye6f9+js2Fz5IFZx4PK+MR+JSy/sTn7/rm9u7+/f3C4m/m7pACDNQAX
yZ/iJgQAAA==
} 64#{
eJxz93SzsEwMZwhn+M4AAg1g3ACmGsCsBjDnPxj/B1P/waz/YM4oGAXDBij+ZAHT
OiAClCfYOf4zMHLIeGxYcLCZQ1gr5sSGhYfbBZS95nhsXHS0W8I4686JjYuP9ys4
d8l4blpycrJG8KqYk5uWendyiezhkdy8zHemsfm9O5LG6m7zHGqjWKRCMo7MY+h4
```

Z/IrYGXwMp65dq2rAl6FrGJbG3fUKuB12DrPvVqs2gFvwlelHZ/ku3qadvSilMP7
9kqW653fWvay6ezq67rxS6r/P1qjPWPDg4Nu/N+/rvyh9/iYt7zzNs0So6enpi2M
cuuRNLp3qJH/d6hNlEnY+eXS09l6w0qzLq+PPP7s98yy3N2Fp5+dvTtVN78lqf77
u5XTi3wfHpYVj5lTnX3xfsHkeDe98qrS11catc/PK7D+/u74fnNpHv19e35+fnF5
dfz5fXt/f//w+PR8//37/v5mYGJisAYARqapGj4EAAA=
} 64#{
eJxz93SzsEwMZwhn+M4AAg1g3ACmGsCsBjDnPxj/B1P/waz/YM4oGAXDBij+ZAHT
OiAClCfYOf4zMHLIeGxYcLCZQ1gr5sSGhYfbBZS95nhsXHS0W8I4686JjYuP9ys4
d8loBjYyMaj5ToqJNHjqOV0zsq/l56RnnjPlcq9t6Zy8+Nx8w+okFq8vywK6XDvl
ZGdNeR7Uyb9oUY7X55dH2INX7trCZbr62oIYSa+vv65mRDRHs05rrRR7GLU09+K+
v5LmD++sKuW/d3R2+YO4fbUn//G+MV+bsKpF9JzvnSKDx/vbhJ3DTkbo3j5coB2v
F72z4MzWubrBbLJWL25fWuZv7/d6y4q0bdMNj6udub7mzYnGuVV+v6qK8k/sl/We
l7Nb/+Ojyv5ytX0yFq/2LnRdfW3P79ef515b73/9nFRGSVPJ00c2fXwSf9685y1d
7B9ft/fu53ei/f3/5xnVtie8f33//P79wEKATeNBA4tYxoNGDrUVD5p4zF48aBZw
00h0ZGRksAYAd264o18EAAA=
} 64#{
eJxz93SzsEwMZwhn+M4AAg1g3ACmGsCsBjDnPxj/B1P/waz/YM4oGAXDBij+ZAHT
OiAClCfYOf4zMHLIeGxYcLCZQ1gr5sSGhYfbBZS95nhsXHS0W8I4686JjYuP9ys4
d8l4blpycrJG8KqYk5tUvVi5Yia1eG5edqbPtPjSnpWBy/0YDCvDvvwsXh7Q6TL5
kI1UYGbQMv65Wq2nAl6FrApd++vIrA8HmRc4smbxni59cH294d46Vu2tOQc3OzDO
cc2+ujZiZ9zjc6mvr+hFNGV+/rT31bUX9xuTTybFWllsTFzXI5uv6xO2yXe3m669
nrfIxrAzDaLqx9bc2Jx8aVZ90bWcWYZXr6xj39+W++NT4K1VuZ9LeqPfpM2cWHj8
ytmQHx/u79b9zSf3e9un5iOth/QkYnd9fHVy/fSydbWl5e8PBbYHLreJ+1Oyv1d1
cX5tVe2Li+94t/X7y9b9Wf5y4mx3u5919d/Orr1+s8jyovr9ZFYpjol1XGYvHjQL
uGk8bBEJy3jYKpG24mGbTNmLh+0KbRqPOoTYWBisAbfrxM90BAAA
} 64#{
eJxz93SzsEwMZwhn+M4AAg1g3ACmGsCsBjDnPxj/B1P/waz/YM4oGAXDBij+ZAHT
OiAClCfYOf4zMHLIeGxYcLCZQ1gr5sSGhYfbBZS95nhsXHS0W8I4686JjYuP9ys4
d8l4blpycrJG8KqYk5uWnp5ukHxqjufmZWdnWxRHhRwIfu46z6Hx1xSJLSsuLOTt
1XLdFfDy0mIfTqu5t4xfOayKWMt04NRVretrAvc3yWqVrTm/LnqlUuusba9Ct6aL
ctQ4mL+9syt3+jHWgO+Nd/fVPXxm88p8Q8y+Gl7/q5Il667sZjp7S0drqm7UHP/T
UrJ7LNc/2zFFOXudlNWyG9uzvs6yO1NgEj29V3RXH2/1tzfTthVv9lt52+zdvcXZ
zPZ/rb99OKfvLF+vu+d50Xaju3b3bSutnj+fsTx4/sra6pK3N9fed2Op/2uR/OZ5
+/pQf7GKiJ37tlb905I3LVw7s//St1W7NgW8f/l1+41qZr6O+MxvjuH3m3jMXjxo
FnDTeNgiEpbxsFUibUViGyMjgzUAhlm/D2kEAAA=
} 64#{
eJxz93SzsEwMZwhn+M4AAg1g3ACmGsCsBjDnPxj/B1P/waz/YM4oGAXDBij+ZAHT
OiAClCfYOf4zMHLIeGxYcLCZQ1gr5sSGhYfbBZS95nhsXHS0W8I4686JjYuP9ys4
d8l4blpycrJGcFnIAgdVr2kGybtEJDernZmpnfsqp9P48bn5tvr/ZKSuPApY4Koo
Fzvry8OgZb6Sdq1Sog9DZjJlh/16mLz2ZeDfU3c3SuClwzQm+RWsC6bqOC7JOrwo
Vnv72uht1gfbeK0n6MWtKW/8pbrj2/uI7QU/F9Vmf14XMbfnolxpjWlR3GGbyXZb
a3ZufLY619b5H8+vnNRL8z7K6ciWbnG80B7Y3SZrrZF7bVN+ee6q6uKr9/ZFM8/X
qfnx7s6xYPGrs+7oPXrWzex83qes6svaa+v/n9OrtUp9fX9ve7j/ux8fP3x61rjY
vLZ6b+iNdzsPre/9l5a86itjv21cXGXk5p+Wx+fVM3K9CK15v7MtwZlL74RCAp+b
xsMWkbCMh60SaSsetsmUvXjYrtCm8ahDZVrGo06NPFEBBmsAOJHArHoEAAA=
} 64#{
eJxz93SzsEwMZwhn+M4AAg1g3ACmGsCsBjDnPxj/B1P/waz/YM4oGAXDBij+ZAHT
OiAClCfYOf4zMHLIeGxYcLCZQ1gr5sSGhYfbBZS95nhsXHS0W8I4a8uKBYvd+6Wd
i/54bFp8YjKf9yqTzk2ph6ZqxZ4S4dj87Mw00+J7IjM3Pz/Xa1v674jElecXJrom
yq3NKFbwWC4/PSiE68FB5llMay/1aJkuClobLhqyV2pa9vUp8SeZBLjL1t7czDM7
S9ViukrMlpCNYj2V5YlB03x/7/uzu3RpQqsjL5tdjYFhyIF8yfehWT82Rmz3VxXf
9rvi0+VJs8zdv8lsLYo/NK2b699pqS93r20wLu/lrTbNvbYt3/rcWmv9x5f2prb7
lVZbvHxwrPO1n94u8+IzB/XV+/VsTEpfXl5pn+9Xbf3l6b2J1cHP+6psKhc/43zk
d99Cs/qrXW17eW3N17Jfp1aff17zb2/Rjz8/v8uWMf1aGt/IobbiQROP2YsHzQJu
Gg9bRMIyHrZKpK142CZT9uJhu0KbxqMOlWk7Eh0YrAGyBMCKdgQAAA==
} 64#{
eJxz93SzsEwMZwhn+M4AAg1g3ACmGsCsBjDnPxj/B1P/waz/YM4oGAXDBij+ZAHT
OiAClCfYOf4zMHLIeGxYcLCZQ1gr5sSGhYfbBZS95nhsXHS0W8I4686JjYuP9ys4
d8l4blpycrJG8KqYk5uWnp5ukHxqjufmZb79XEWvrlROfnRuvn21F4tXSOOFNptu
JttVBisuzfURtJsrdfXBleWhnHFLZ5VqX18V18lnImW6JmwT/yamD1ofHG9tZbi0
TLV6ytrbOwqeHkrNCtePaiypntX7u+z9rTml7OIxWiZrbhy2kbbm45IsTDrevTDu
GM/PgptrkzWj360qefhi9nLH+b09VUa3Z62zPN+zNkLt7fVt+eK21tHf8w40Jv7S
Oxv148Pxg73y1898t4h4Pnvh9rh5c9S+XjZbH/5+757K7y/22bc716+Lzn168ln4
db/1917kfwvbOH+6/zzLD8ez7p/X9/u1/d+fiEq2+Joe3owHjRxqKx408Zi9eNAs

```
4KbxsEUkLONhq0SaqACDNQAYMLy/ZgQAAA==
}]
count: 1
make-dir %./frames/
for count 1 length? anim-frames-block 1 [
    write/binary rejoin [
        %./frames/ "frame-" count ".gif"
    ] to-binary decompress pick anim-frames-block count
]
file-list: read %./frames/
animation-frames: copy []
for count 1 length? file-list 1 [
    append animation-frames rejoin [
        %./frames/ "frame-" count ".gif"
    ]
]
view center-face layout [
    size 625x415
    backcolor black
    anim: anim rate 10 frames load animation-frames
    btn "Run Animation" [
        for counter 0 31 1 [
            anim/offset: anim/offset + (as-pair counter 0)
            show anim wait .05
        ]
        for counter 0 24 1 [
            anim/offset: anim/offset + (as-pair 0 counter)
            show anim wait .05
        ]
        for counter 0 31 1 [
            anim/offset: anim/offset + (as-pair (negate counter) 0)
            show anim wait .05
        ]
        for counter 0 24 1 [
            anim/offset: anim/offset + (as-pair 0 (negate counter))
            show anim wait .05
        ]
    ]
]


REBOL [Title: "Simple Menu Example"]
view center-face gui: layout [
    size 400x300
    at 100x100 H3 "You selected:"
    display: field
    origin 2x2 space 5x5 across
    at -200x-200 file-menu: text-list "item1" "item2" "quit" [
        switch value [
            "item1" [
                face/offset: -200x-200
                show file-menu
                ; PUT YOUR CODE HERE:
                set-face display "File / item1"
            ]
            "item2" [
                face/offset: -200x-200
                show file-menu
                ; PUT YOUR CODE HERE:
                set-face display "File / item2"
            ]
            "quit" [quit]
        ]
    ]
```

```
    at 2x2
    text bold "File" [
        either (face/offset + 0x22) = file-menu/offset [
            file-menu/offset: -200x-200
            show file-menu
        ][
            file-menu/offset: (face/offset + 0x22)
            show file-menu
        ]
    ]
    text bold "Help" [
        file-menu/offset: -200x-200
        show file-menu
        ; PUT YOUR CODE HERE:
        set-face display "Help"
    ]
]


REBOL [title: "Days Between"]
sd: ed: now/date
view layout [
    btn "Select Start Date" [
        sd: request-date
        sdt/text: to-string sd
        show sdt
        db/text: to-string ((to-date edt/text) - sd)
        show db
    ]
    sdt: field to-string sd [
        either error? try [to-date sdt/text] [
            alert "Improper date format."
        ] [
            db/text: to-string ((to-date edt/text) - (to-date sdt/text))
            show db
        ]
    ]
    btn "Select End Date" [
        ed: request-date
        edt/text: to-string ed
        show edt
        db/text: to-string (ed - (to-date sdt/text))
        show db
    ]
    edt: field to-string ed [
        either error? try [to-date edt/text] [
            alert "Improper date format."
        ] [
            db/text: to-string ((to-date edt/text) - (to-date sdt/text))
            show db
        ]
    ]
    h1 "Days Between:"
    db: field "0" [
        either error? try [to-integer db/text] [
            alert "Please enter a number."
        ] [
            edt/text: to-string (
                (to-date sdt/text) + (to-integer db/text)
            )
        ]
        show edt
    ]
]
```

```
REBOL [title: "Update My Haxelibs"]
write %haxelibs.txt read http://lib.haxe.org/files/
the-list: read/lines %haxelibs.txt
clean: copy []
foreach line the-list [
    x: (parse/all form (find line ".zip") ">")
    if (length? x) > 2 [
        y: parse form second x "<"
        append clean first y
    ]
]
errors: copy []
make-dir %haxelibs
change-dir %haxelibs
save %list.txt clean
downloaded: read %.
if exists? %previously_downloaded.txt [
    append downloaded load %previously_downloaded.txt
]
save %previously_downloaded.txt unique downloaded
foreach file clean [
    if not (find downloaded (to-file file)) [
        either error? try [size? (join http://lib.haxe.org/files/ file)] [
            print join "ERROR:  " file
            append errors file
        ] [
            print rejoin [
                {Downloading:  } file {  (}
                size? (join http://lib.haxe.org/files/ file) { kb)}
            ]
            if error? try [
                write/binary
                    (to-file file)
                    (read/binary (join http://lib.haxe.org/files/ file))
            ] [
                print join "ERROR:  " file
                append errors file
            ]
        ]
    ]
]
save %haxe_lib_download_errors.txt errors
halt
foreach file clean [if find downloaded (to-file file) [print file]]


REBOL [title: "Embed a Folder Full of Files"]
system/options/binary-base: 64
sounds: copy []
foreach file load %./  [
    print file
    uncompressed: read/binary file
    compressed: compress to-string uncompressed
    if ((length? uncompressed) > 5000) [
        append sounds compressed
    ]
]
editor sounds


REBOL [title: "Add Spaces Before Comments"]
file: %/C/7-14-10/My_Docs/rebol/rebol_examples.txt
lines: read/lines file  newlines: ""  total-lines: length? lines  count: 1
foreach line lines [
```

```
    print rejoin ["Line:  " count " of " total-lines]
    line: to-string line
    linesize: length? line
    while [(length? line) < 76] [  ; currently, max length of line = 75
        if not find line #";" [break]
        print "updating..."
        replace/all line ";" " ;"
    ]
    newlines: rejoin [newlines line "^/"]
    count: count + 1
]
editor newlines
```

```
REBOL [title: "Generic CGI Form Submission Viewer"]
sort-column: 4  ; even numered cols contain data (2nd col is time stamp)
signups: load http://yoursite.com/submitted_forms.txt
do create-list: [
    name-list: copy []
    foreach item signups [append name-list (pick item sort-column)]
]
view center-face layout [
    the-list: text-list 600x150 data name-list [
        foreach item signups [
            if (pick item sort-column) = value [
                dt: copy ""
                foreach [label data] item [
                    dt: rejoin [
                        dt newline label "  " data
                    ]
                ]
                a/text: dt
                show a
            ]
        ]
    ]
    a: area 600x300 across
    btn "Sort by..." [
        sort-column: to-integer request-text/title/default {
            Data column to list:} "4"
        do create-list
        the-list/data: name-list
        show the-list
    ]
    tab text join (form length? signups) " entries."
]
```

```
REBOL [title: "Generic Form Submission Editor"]
submissions: load http://yoursite.com/submitted_forms.txt
do create-list: [
    data: copy []
    foreach block submissions [append/only data (extract/index block 2 4)]
    datastring: copy {}
    foreach block data [
        datastring: join datastring "[^/"
        foreach item block [datastring: rejoin [datastring item newline]]
        datastring: join datastring "]^/^/"
    ]
    editor datastring
]
```

```
REBOL [title: "Determine OS"]
switch system/version/4 [
    2 [print "OSX"]
    3 [print "Windows"]
    4 [print "Linux"]
    7 [print "FreeBSD"]
    8 [print "NetBSD"]
    9 [print "OpenBSD"]
    10 [print "Solaris"]
] [alert "Can't be dertermined"]


REBOL [title: "Insert HTML Into Web Page"]
file: %mp3.html
a-string: {<BODY bgcolor="#C8C8C8">}
; first way:
write file read http://re-bol.com/examples/mp3.html
content: read file
insert (skip find content a-string length? a-string) trim {
    <center><h1>MP3 Example!</h1></center>}
write file content
editor file
; second way:
write file read http://re-bol.com/examples/mp3.html
content: read file
begin: (index? find content a-string) + (length? a-string)
altered: rejoin [
    (copy/part content begin)
    {<center><h1>MP3 Example!</h1></center>}
    (at content begin)
]
write file altered
editor file


REBOL [title: "Find Long Lines"]
doc: read/lines to-file request-file
the-text: {}
foreach line doc [
    if ((find/part line "   " 4)) [
        if ((length? line) > 78) [
            print line
            the-text: rejoin [the-text newline line]
        ]
    ]
]
editor the-text


REBOL [title: "Sync Time to Web Server"]
write ftp://user:pass@site.com/public_html/time.cgi
    {#! /home/path/public_html/rebol/rebol -cs
    REBOL [title: "time"]
    print "content-type: text/html^/"
    print now}
dif: 7:00  ; difference between web server and your local time zone
date: (to-date trim read http://site.com/time.cgi) + dif
lib: load/library %kernel32.dll
set-clock: make routine! [
    systemtime [struct! []]
    return:    [integer!]
] lib "SetSystemTime"
current: make struct! [
    wYear          [short]
```

```
    wMonth        [short]
    wDayOfWeek    [short]
    wDay          [short]
    wHour         [short]
    wMinute       [short]
    wSecond       [short]
    wMilliseconds [short]
] reduce [
    date/year
    date/month
    date/weekday
    date/day
    date/time/hour
    date/time/minute
    to-integer date/time/second
    0
]
either ((set-clock current) = 1) [
    ask rejoin ["Time has been set to:  " now "^/^/[Enter]... "]
] [
    ask "Error setting time.  Please check your Internet connection."
]
free lib
; see http://www.fm.tul.cz/~ladislav/rebol/nistclock.r


REBOL [Title: "UDP Group Chat"]
net-in: open udp://:9905  ; This is UDP, so NO known IP addresses required
net-out: open/lines udp://255.255.255.255:9905
set-modes net-out [broadcast: on]
svv/vid-face/color: white
name: request-text/title "Your name:"
prev-message: ""
gui: view/new layout [
    a1: area wrap rejoin [name ", you are logged in."]  across
    f1: field
    btn "Save Chat" [write request-file/only/save/file %chat.txt a1/text]
    btn "?" [alert "Press [CTRL] + U to see who's online."]
    at 0x0 key #"^M" [
        if f1/text = "" [return]
        insert net-out rejoin [name {, } now/time {:    } f1/text]
    ]
    at 0x0 key #"^u" [
        insert net-out rejoin [name {, } now/time {:    Who's online?}]
    ]
]
forever [
    focus f1
    received: wait [net-in]
    if not viewed? gui [quit]
    if find (message: copy received) "Who's online" [
        insert net-out rejoin [name " is online."]
    ]
    if message <> prev-message [
        insert (at a1/text 1) message show a1
        attempt [
            insert s: open sound:// load %/c/windows/media/ding.wav
            wait s close s
        ]
    ]
    prev-message: copy message
]
```

```
REBOL [title: "Request Examples"]
request "Could this be useful?"
request ["Just some information."]
request ["Here are 2 buttons with altered text:" "Probably" "Not Really"]
request ["3 buttons with altered text:" "Probably" "Not Really" "Dunno"]
answer: form request ["Complex example:" "choice 1" "choice 2" "choice 3"]
switch/default answer [
    "true"  [the-answer: "choice 1"]
    "false" [the-answer: "choice 2"]
    "none"  [the-answer: "choice 3"]
] []
alert join "You chose: " the-answer
request/type ["Here's a better icon for information display."] 'info
request/type ["Altered title and button text go in a block:" "Good"] 'info
request/ok/type "This example is the EXACT same thing as 'alert'." 'alert
request/ok/type "Here's alert with a different icon." 'info
request/ok/type "Here's another icon!" 'stop


REBOL [title: "AutoIt DLL Example"]
if not exists? %AutoItDLL.dll [
    write/binary %AutoItDLL.dll
    read/binary http://musiclessonz.com/rebol_tutorial/AutoItDLL.dll
    write/binary %madplay.exe
    read/binary http://musiclessonz.com/rebol_tutorial/madplay.exe
]
lib: load/library %AutoItDLL.dll
move-mouse: make routine! [
    return: [integer!] x [integer!] y [integer!] z [integer!]
] lib "AUTOIT_MouseMove"
send-keys: make routine! [
    return: [integer!] keys [string!]
] lib "AUTOIT_Send"
winactivate: make routine! [
    return: [integer!] wintitle [string!] wintext [string!]
] lib "AUTOIT_WinActivate"
set-option: make routine! [
    return: [integer!] option [string!] param [integer!]
] lib "AUTOIT_SetTitleMatchMode"
set-option "WinTitleMatchMode" 2
call/show {madplay.exe -v *.mp3}
view layout [
    across
    btn "forward" [
        winactivate "\reb" ""
        send-keys "f"
    ]
    btn "back"[
        winactivate "\reb" ""
        send-keys "b"
    ]
    btn "volume up" [
        winactivate "\reb" ""
        send-keys "+"
    ]
    btn "volume-down" [
        winactivate "\reb" ""
        send-keys "-"
    ]
    btn "pause" [
        winactivate "\reb" ""
        send-keys "p"
    ]
    btn "quit" [
```

```
        winactivate "\reb" ""
        send-keys "q"
        quit
    ]
]


REBOL [title: "Word Count Example"]
view layout [
    i: info rate 0 feel [
        engage: func [f a e] [
            if a = 'time [
                l: length? parse m/text none
                i/text: join "Wordcount: " l
                show i
            ]
        ]
    ]
    m: area
]


REBOL [title: "FizzBuzz"]
repeat i 100 [
    j: ""
    if i // 3 = 0 [j: "fizz"]  ; modulo operator (can also use "mod")
    if i // 5 = 0 [j: join j "buzz"]
    if j = "" [j: i]
    print j
]


REBOL [title: "Test Script Loop"]
do test-script: [
    if error? try [previous-file: read %previous-file] [previous-file: ""]
    current-file: request-file/only/file previous-file
    if current-file = none [quit]
    write %previous-file current-file
    do current-file   ; launch current-file
    if true = request "Run again?" [do test-script]
]


REBOL [Title: "Quick Color Guide"]
echo %colors.txt ? tuple! echo off
lines: read/lines %colors.txt
colors: copy []
gui: copy [across space 1x1]
count: 0
foreach line at lines 2 [
    if error? try [append colors to-word first parse line none][]
]
foreach color colors [
    append gui [style box box [alert to-string face/color]]
    append gui reduce ['box 110x25 color to-string color]
    count: count + 1
    if count = 5 [append gui 'return count: 0]
]
view center-face layout gui


REBOL [title "Several List Widget Examples"]
x: copy []
for i 1 10 1 [
```

```
    append/only x reduce [form random 1000 form random 1000]
]
slider-pos: 0
view layout [
    across
    the-list: list 220x240 [across text 100 text 100] supply [
        count: count + slider-pos
        if none? q: pick x count [face/text: none exit]
        face/text: pick q index
    ]
    slider 16x240 [
        slider-pos: (length? x) * value
        show the-list
    ]
    return
    btn "Remove" [remove head x  show the-list
    ]
    btn "Add" [
        insert/only head x reduce [form random 1000 form random 1000]
        show the-list
    ]
]
x: copy [
    ["row 1, column 1" "row 1, column 2"]
    ["row 2, column 1" "row 2, column 2"]
    ["row 3, column 1" "row 3, column 2"]
    ["row 4, column 1" "row 4, column 2"]
]
do qq: [view gui: layout [
    the-list: list 304x100 [
        across space 0
        info 150 [face/text: request-text/default face/text show gui]
        info 150 [face/text: request-text/default face/text show gui]
    ] supply [
        either count > length? x [face/text: "" face/image: none] [
            the-list/set-it face x index count
        ]
    ]
    across
    btn "Save" [
        save to-file request-file/save x
    ]
    btn "Load" [
        x: copy load to-file request-file
        unview do qq
    ]
]]


REBOL [title: "REBOL/flash Build Tool"]
write %shape.rswf {
    REBOL [
        type: 'swf
        file: %shape.swf
        background: 230.230.230
        rate: 40
        size: 320x240
    ]
    a-rectangle: Shape [
        Bounds 0x0 110x50
        fill-style [color 255.0.0]
        box 0x0 110x50
    ]
    place [a-rectangle] at 105x100
```

```
    showFrame
    end
}
write %text.rswf {
    REBOL [
        type: 'swf
        file: %text.swf
        background: 255.255.255
        rate: 40
        size: 320x240
    ]
    fnt_Arial: defineFont2 [name "Arial"]
    some-text: EditText 'the-text 110x18 [
        Color 0.0.0
        ReadOnly
        NoSelect
        Font [fnt_Arial 12]
        Layout [align: 'center]
    ]
    place [some-text] at 105x100
    doAction [the-text: "Hello world!"]
    showFrame
    end
}
do [
    my-rswf-folder: %./
    change-dir my-rswf-folder
    do http://re-bol.com/rswf.r   ; do %rswf.r
    current-source: to-file request-file/filter/file """*.rswf" %text.rswf
    unset 'output-html
    do edit-compile-run: [
        editor current-source
        if error? err: try [make-swf/save/html current-source] [
            err: disarm :err
            alert reform [
                "The following compile error occurred: "
                err/id err/where err/arg1
            ]
            either true = request "Edit/Compile/Run Again?" [
                do edit-compile-run quit
            ] [
                quit
            ]
        ]
        unless value? 'output-html [
            output-html: to-file request-file/filter "*.html"
        ]
        browse output-html
        if true = request "Edit/Compile/Run Again?" [do edit-compile-run]
    ]
]


REBOL [title: "Generate from source and upload SWF and HTML to web site"]
write %mp3.rswf {
    REBOL [
        type: 'swf5
        file: %mp3.swf
        background: 200.200.200
        rate: 12
        size: 1x1
    ]
    mp3Stream http://re-bol.com/example.mp3
    finish stream
```

```
    end
}
source-file: %mp3.rswf
output-html: %mp3.html
output-swf:  %mp3.swf
inserted-html: {<center><h1>MP3 Example!</h1></center>}
insert-at: {<BODY bgcolor="#C8C8C8">}
my-ftp-info: ftp://username:password@site.com/public_html/folder/
destination-url: http://re-bol.com/examples/mp3.html
do http://box.lebeda.ws/~hmm/rswf/rswf_latest.r  ; do %rswf.r
make-swf/save/html source-file
content: read output-html
insert (skip find content insert-at length? insert-at) inserted-html
write output-html content
write (join my-ftp-info form output-html) (read output-html)
write/binary (join my-ftp-info output-swf) (read/binary output-swf)
browse destination-url
```

```
REBOL [title: "Little GUI Metaprogramming Example"]
view center-face layout [
    text "Select a button width, in pixels:"
    d: drop-down data [250 400 550]
    text "Enter any number of button labels (text separated by spaces):"
    f: field 475
    btn "Generate GUI" [
        created-buttons: copy compose [
            style new-btn btn (to-integer d/text) [
                alert join "This button's label is: " face/text
            ]
        ]
        foreach item to-block f/text [
            append created-buttons compose [
                new-btn (form item)
            ]
        ]
        view/new center-face layout created-buttons
    ]
]
```

```
REBOL [title: "Spelling Rules Pretty Printer"]
rules: [
    {"a" says uh} [ago alone ahead above agree]
    {"ai" says long a} [main rain paint wait afraid]
    {"air"} [chair airplane hair unfair repair]
    {"alk"} [walk talk]
    {ar says ar} [alarm farther charge garbage]
    {"ar" says "or"} [warm war]
]  ; rules: load http://re-bol.com/spelling_rules.txt
html: copy {<center><h1>Spelling Rules</h1>
    <table border=1 cellpadding=5 width=90%>}
count: 1
foreach [rule words] rules [
    if count = 1 [append html {<tr>}]
    append html rejoin [
        {<td width=33% valign=top><h3>} (uppercase rule) {</h3>}
    ]
    sort words
    foreach word words [
        append html join form word "<br>"
    ]
    append html {</td>}
    count: count + 1
```

```
    if count > 3 [
        append html {</tr>}
        count: 1
    ]
]
append html {</tr></table></center>}
write %spelling_table.html html
browse %spelling_table.html


REBOL [title: "Student Photo Database (variation on Data Card File)"]
write %StudentList.csv {STUDENTID,LASTNAME,FIRSTNAME,DOB,GRADE
111111,Doe,Steven D,6/16/1992,12
111112,Doe,Jonathan Daniel,12/16/1991,12
111113,Smith,Karen J,12/3/1991,12
111114,Jones,Michael J,6/4/1992,12
111115,Taylor,Ryan C,1/10/1992,12
111116,Adam,Kaitlan C,4/30/1992,12
111117,Washington,Gabryela,3/31/1992,12
111118,Travolta,Juan D,1/24/1992,12
111119,Cruise,Amber E,5/8/1992,12}
either exists? %data.txt [
    database: load %data.txt
][
    filename: %StudentList.csv
    database: copy []
    lines: read/lines filename
    foreach line lines [
        append database parse/all line ","
    ]
    remove/part database 5                    ; get rid of headers
    for counter 6 ((length? database) + 12) 6 [
        insert (at database counter) to-file rejoin [
            "/C/Photos/image_" (pick database (counter - 5)) ".jpg"
        ]
    ]
    save %data.txt database
]
update: func [marker] [
    n/text: pick database marker
    a/text: pick database (marker + 1)
    p/text: pick database (marker + 2)
    o/text: pick database (marker + 3)
    g/text: pick database (marker + 4)
    i/text: pick database (marker + 5)
    if error? try [photo/image: load to-file i/text] [
        ; alert "No image selected"
        photo/image: none
    ]
    photo/text: ""
    show gui
]
view center-face gui: layout [
    text "Load an existing record:"
    name-list: text-list blue 300x80 data sort (extract database 6) [
        if value = none [return]
        marker: index? find database value
        update marker
    ]
    text "ID:"       n: field 300
    text "Last:"     a: field 300
    text "First:"    p: field 300
    text "BD:"       o: field 300
    text "Grade:"    g: field 300
```

```
text "Image:"    i: btn 300 [
    i/text: to-file request-file
    photo/image: load to-file i/text
    show gui
]
at 340x20 photo: image white 300x300
across
btn "Save" [
    if n/text = "" [alert "You must enter a name." return]
    if find (extract database 6) n/text [
        either true = request "Overwrite existing record?" [
           remove/part (find database n/text) 6
        ] [
           return
        ]
    ]
    save %data.txt repend database [
        n/text a/text p/text o/text g/text i/text
    ]
    name-list/data: sort (extract copy database 6)
    show name-list
]
btn "Delete" [
    if true = request rejoin ["Delete " n/text "?"] [
        remove/part (find database n/text) 6
        save %data.txt database
        do-face clear-button 1
        name-list/data: sort (extract copy database 6)
        show name-list
    ]
]
clear-button: btn "New" [
    n/text: copy  ""
    a/text: copy  ""
    p/text: copy  ""
    o/text: copy  ""
    g/text: copy  ""
    i/text: copy  ""
    photo/image: none
    show gui
]
next-btn: btn "Next"  [
    if error? try [
        old-num: copy n/text
        n/text: form ((to-integer n/text) + 1)
        show n
        marker: index? find database n/text
        update marker
    ] [n/text: copy old-num show n alert "No more records"]
]
prev-btn: btn "Previous"  [
    if error? try [
        old-num: copy n/text
        n/text: form ((to-integer n/text) - 1)
        show n
        marker: index? find database n/text
        update marker
    ] [n/text: copy old-num show n alert "No more records"]
]
key keycode [down] [do-face next-btn 1]
key keycode [up] [do-face prev-btn 1]
at 340x340 d1: drop-down 300 data [
    "Last Name" "First Name" "Birthday" "Grade"
]
```

```
    at 340x380 f2: field 300 "Select field above, type search text here" [
        if d1/data = none [alert "Select a search field above" return]
        search-field: to-integer select [
            "Last Name" 2 "First Name" 3 "Birthday" 4 "Grade" 5
        ] d1/data
        results: copy []
        for counter search-field (length? database) 6 [
            if find (pick database counter) copy f2/text [
                append results pick database (counter - search-field + 1)
            ]
        ]
        t/data: copy results  show t
        if [] = results [alert "None found"]
    ]
    at 340x420 t: text-list 300x60 [
        name-list/picked: copy value
        show name-list
        if value = none [return]
        marker: index? find database value
        update marker
    ]
]


REBOL [title: "Space Invaders Shootup"]
alien1:  load to-binary decompress 64#{
eJx9UzFLQzEQjijUOognHTIVhCd0cXJ1kLe3g7SbFKcsWQoWZ7MFhNKxg0PpH3Cx
WbKUqpPoUNcOPim1Q+kPkCJekvf0NTx7cLl7d8l33+XywvL+FrFyhVpCPUY9QN0g
LnG7ScjjrtM98iedToeM3kbW7/f71k4/p6R+USe9Xo/UqjUbi94jMhgMrL/8XpLm
ZZP4spPyzxVTT35MM2Zir4vFYu4dM7GP2M483Fa8f8w0O/Vy24yzo8RXipfJmdb8
kJxwrdJ7K4gxiSs7/09czYpdW6vcsI+AtrEKQ7ScDPlLHO/aNQ8huzaVeSDaHrNi
3IlBjDI6mqVsWvIA0E5ZJ2OtlUIuAKHmqoS5kHOt9UPMP0sm3TU5PHdHQVIZMs3v
qZTPmrMAQAj6ZXOSUtkwPKRwloKQNlexCDOvR4fpclGq76KNzC2mQPiG681i5gAw
ZusVJEAh5JojBzrEGQYC2dncuh7+y83d7ASVAu8MpAQqkT9+3Gg3Q+wHI2AZSAFm
1+99FzMQkzllVUxeTFUrc4vC4Q4VV4wlLyaerjD1XPe+tLxK8SNbqTrJOIf/Bd4X
V+VU7AfjSm0ZEgQAAA==
}
ship1: load to-binary decompress 64#{
eJx1Us9L3EAU/rTbMdHE9VlYukSQFhUpvXjQXrfizR8XkYAnt0oQVsTLGlgEFdSL
l3jqpXgre6sEJAgDsidPIul/sHjopeIfILj0JdnV3ez0y7zJzLx533vvY2YXhzOI
scs2yfaF7QNbDxLHjzfAu4HEhpKryAgDfccC4rfAws0IjF/96HsWyH7XMNXIon9Z
QJvWsNQw8XZP4NPlKD73Whi/HcZO4z207fv7jyo8/jk4rlTdFQXcSrV+flEtq3x2
5amuB44lyU+BpHRKHq4dKXnZCbLkxl9kOF5BarPVDFWyBAWcEAVFsjrhENGmhyPK
UXe+XNHf9HqZW9GgyzUUoloqXcXE1wv6iSTHohSkQ8yJQ512RCiSvPIGbaVkTFuu
Ge5/erfdurb+wM3ETZHPyjaX5NzNHPATOHMsn894sMZJWX4uH78OYSvTrUU+paI2
q8nQl5JHMFaSOZLBbHPnoR2ndHUa5NtPwubfFKziT1YqRDdY2VV3JckT3X2ZIlwW
KQjmUxGhGQ0Ecm5OlhBvUsSi/NpXmjLRoFx4YWuL0789fN24m+jsK2x+wGE+JjLR
DePiqdbKZqZojf1qLZ2ptdO3ZrxXwjCODzuThK3Af4EF8jYSBAAA
}
alien-fire: load to-binary decompress 64#{
eJxz8o1jZACDMiDWAGI+IJYFYkYGFrD4CyAW5oZgAYhSBhZmFoaWphaG48eOMwQF
BDFoaGgwPH36lGHZsmUM4uLiDFk5WQyzZs1iuHHzBsOfv38Ydu7cyWBhZsFQXlrO
EBEVATTBaWlolAoDA/vp3bt37wHyZwPpTUCaedqpUBWGS6HLMj8AedpA0Z1QGqTK
KXrNtCdgF/BLtrCD6GywOAPDabA6BobCTAMwXTfzFMh8uM7ZUBpi/p3QZdMMwLp2
796GbH7omrR2sH6Omc+h5m4C09pQuiKzHWp+O1R+D1QeQjstPQINIwag+wBUhlwj
XgEAAA==
}
ship-fire: load to-binary decompress 64#{
eJxz8t3FAAFlQKwBxOxALAjEjAwsYHEXIBbmhmABqFo2FhYG9l4eBvajbAwKSTIM
/H8FGFjUOBg4tnEyGP1VYWAXZWOwadNg4KhiYdA5JMLAacbJIHNLhUFnkgiDIpMg
2IyDd2UYVMqdGNLLyxoOz7RpCJ5p2pDi4sYAwlFpSz+AcEoJkF8O5KstZWhUkvig
4uLEoAIUO7f7zQcA8m8lvboAAAA=
}
bottom: 270  end: sidewall: false  random/seed now
```

```
b: ['image 300x400 ship1 'line -10x270 610x270]
for row 60 220 40 [
    for column 20 380 60 [
        pos: to-pair rejoin [column "x" row]
        append b reduce ['image pos alien1]
    ]
]
view center-face layout/tight [
    scrn: box black 600x440 effect [draw b] rate 1000 feel [
        engage: func [f a e] [
            if a = 'key [
                if e/key = 'right [b/2: b/2 + 5x0]
                if e/key = 'left [b/2: b/2 - 5x0]
                if e/key = 'up [
                    if not find b ship-fire [
                        fire-pos: b/2 + 25x-20
                        append b reduce ['image fire-pos ship-fire]
                    ]
                ]
                system/view/caret: none
                show scrn
                system/view/caret: head f/text
            ]
            if a = 'time [
                if (random 1000) > 900 [
                    f-pos: to-pair rejoin [random 600 "x" bottom]
                    append b reduce ['image f-pos alien-fire]
                ]
                for i 1 (length? b) 1 [
                    removed: false
                    if ((pick b i) = ship-fire) [
                        for c 8 (length? head b) 3 [
                            if (within? (pick b c) (
                            (pick b (i - 1)) + -40x0) 50x35)
                            and ((pick b (c + 1)) <> ship-fire) [
                                removed: true
                                d: c
                                e: i - 1
                            ]
                        ]
                        either ((second (pick b (i - 1))) < -10) [
                            remove/part at b (i - 2) 3
                        ] [
                            do compose [b/(i - 1): b/(i - 1) - 0x9]
                        ]
                    ]
                    if ((pick b i) = alien1) [
                        either ((second (pick b (i - 1))) > 385) [
                            end: true
                        ] [
                            if ((first (pick b (i - 1))) > 550) [
                                sidewall: true
                                for item 4 (length? b) 1 [
                                    if (pick b item) = alien1 [
                                        do compose [
                                          b/(item - 1): b/(item - 1) + 0x2
                                        ]
                                    ]
                                ]
                                bottom: bottom + 2
                                b/5: to-pair rejoin [-10 "x" bottom]
                                b/6: to-pair rejoin [610 "x" bottom]
                            ]
                            if ((first (pick b (i - 1))) < 0) [
```

```
                                sidewall: false
                                for item 4 (length? b) 1 [
                                    if (pick b item) = alien1 [
                                        do compose [
                                          b/(item - 1): b/(item - 1) + 0x2
                                        ]
                                    ]
                                ]
                                bottom: bottom + 2
                                b/5: to-pair rejoin [-10 "x" bottom]
                                b/6: to-pair rejoin [610 "x" bottom]
                            ]
                            if sidewall = true [
                                do compose [b/(i - 1): b/(i - 1) - 2x0]
                            ]
                            if sidewall = false [
                                do compose [b/(i - 1): b/(i - 1) + 2x0]
                            ]
                        ]
                    ]
                    if ((pick b i) = alien-fire) [
                        if within? ((pick b (i - 1)) + 0x14) (
                            (pick b 2) + -10x0) 65x35 [
                            alert "You've been killed by alien fire!" quit
                        ]
                        either ((second (pick b (i - 1))) > 400) [
                            remove/part at b (i - 2) 3
                        ] [
                            do compose [b/(i - 1): b/(i - 1) + 0x3]
                        ]
                    ]
                    if removed = true [
                        remove/part (at b (d - 1)) 3
                        remove/part (at b (e - 1)) 3
                    ]
                ]
                system/view/caret: none
                show scrn
                system/view/caret: head f/text
                if not (find b alien1) [
                    alert "You killed all the aliens. You win!" quit
                ]
                if end = true [alert "The aliens landed! Game over." quit]
            ]
        ]
    ]
    do [focus scrn]
]


REBOL [title: "PDF Composing Values Example"]
do http://www.colellachiara.com/soft/Misc/pdf-maker.r
xpos: 50  ypos: 200  offset: 5
size: 5  width: (10 * size)  height: (2.4 * size)
page1: compose/deep [[
    image
        (xpos + offset) (ypos + offset)
        (width) (height)
        (system/view/vid/image-stock/logo)
]]
write/binary %example.pdf layout-pdf page1
call %example.pdf
```

```
REBOL [title: "PDF Guitar Fretboard Note Overlay Printer"]
chosen-scale: none
view center-face layout [
    h1 "Fretboard length:"
    text-list "25.5" "27.67" "30" [
        chosen-scale: join "scale" value
        unview
        alert rejoin [
            "Now printing "
            value
            " inch scale fretboard overlay to 'notes.pdf'"
        ]
    ]
]
notes: [
    [{F}{C}{ }{ }{ }{F}]
    [{ }{ }{A}{E}{B}{ }]
    [{G}{D}{ }{F}{C}{G}]
    [{ }{ }{B}{ }{ }{ }]
    [{A}{E}{C}{G}{D}{A}]
    [{ }{F}{ }{ }{ }{ }]
    [{B}{ }{D}{A}{E}{B}]
    [{C}{G}{ }{ }{F}{C}]
    [{ }{ }{E}{B}{ }{ }]
    [{D}{A}{F}{C}{G}{D}]
    [{ }{ }{ }{ }{ }{ }]
    [{E}{B}{G}{D}{A}{E}]
]

scale25.5: [
    36.35 70.66 103.05 133.62 162.47 189.71 215.41 239.67 262.58 284.19
    304.59 323.85 342.03 359.18 375.38 390.66 405.09 418.70 431.56 443.69
    455.14 465.95 476.15 485.77
]

scale27.67: [
    39.45 76.68 111.82 144.99 176.30 205.85 233.74 260.07 284.92 308.38
    330.51 351.41 371.13 389.75 407.32 423.91 439.56 454.34 468.28 481.45
    493.87 505.60 516.67 527.11
]

scale30: [
    42.77 83.14 121.24 157.20 191.15 223.18 253.43 281.97 308.91 334.34
    358.34 381.00 402.38 422.57 441.62 459.60 476.57 492.59 507.71 521.99
    535.46 548.17 560.17 571.50
]

x: 40  line-width: 30  text-width: 4  height: 5

make-overlay: does [
    page1: copy [
        textbox 40 0 4 6 [center font Helvetica 5 "E"]
        textbox 45 0 4 6 [center font Helvetica 5 "B"]
        textbox 50 0 4 6 [center font Helvetica 5 "G"]
        textbox 55 0 4 6 [center font Helvetica 5 "E"]
        textbox 60 0 4 6 [center font Helvetica 5 "A"]
        textbox 65 0 4 6 [center font Helvetica 5 "E"]
    ]
    output: copy []
    for i 1 10 1 [
        y: do compose [pick (to-word chosen-scale) i]
        notes-at-fret: pick notes i
        append page1 compose/deep [
            line width 4 (x) (y) (x + line-width) (y)
```

```
            textbox (x) (y - 10) (text-width) (height + 1) [
                center font Helvetica (height) (notes-at-fret/1)
            ]
            textbox (x + 5) (y - 10) (text-width) (height + 1) [
                center font Helvetica (height) (notes-at-fret/2)
            ]
            textbox (x + 10) (y - 10) (text-width) (height + 1) [
                center font Helvetica (height) (notes-at-fret/3)
            ]
            textbox (x + 15) (y - 10) (text-width) (height + 1) [
                center font Helvetica (height) (notes-at-fret/4)
            ]
            textbox (x + 20) (y - 10) (text-width) (height + 1) [
                center font Helvetica (height) (notes-at-fret/5)
            ]
            textbox (x + 25) (y - 10) (text-width) (height + 1) [
                center font Helvetica (height) (notes-at-fret/6)
            ]
        ]
    ]
    append/only output page1
    write/binary %notes.pdf layout-pdf output
    alert "Done"
]
do http://www.colellachiara.com/soft/Misc/pdf-maker.r
make-overlay


REBOL [title: "RebGUI Point of Sale System"]
write %posp.db {["username" "password"] ["username2" "password2"]} ; etc.
make-dir %./receipts/
write/append %./receipts/deleted.txt ""
unless exists? %scheme_has_changed [
    write %ui.dat decompress #{
        789C9552CB92A3300CBCE72BBCDCA18084995A7E652A078115F08EB129592C33
        7FBFC24E32CC2387A5EC2A49ED6EB56C267845E5BB3FD8F32FF57250F2CD3060
        ABEAA629E23E95B1CAF8C6AD7A3A1571A5D28813E6D60CA32055752AAAE67751
        97CF3B5003BDB6EA5817CF821E9B8804067E484BE04F34BFB035EE4EACCB5371
        DD9FE044AD8E4FC5751FCE6AFA3E648FD6B62A51516F035731BE78B7B9AAEF49
        3EE2D5693A3CC02CCD63B8F5DB8CC464021A8CBB49066B3492901EB4879E8D77
        B92C74BC1D7CD1E467992DB0D8319CA28B41ABE53D42583D691566E31C521438
        7F9161E844241276780F84BCC117DF2F410E480E7BFCBDB7A697FA407E99F3CE
        BF4937875685119195 88E631DF5146131F602FFA1F8645B1437D35A2BA85D93B
        F5317A8C9810BF5DC240E6A1F0CF374CE4D790B31F507E45B9E10BD8801122D0
        6633DEEC5E3CFB8BA4C14176AF6D936540066CA6B2DE2F649094C35532361386
        EC0B270D18660B61CC355A78BFFD53ECBD6533DF8A655BCA4AD08A9D366E905E
        4C4B72B71AA7FDDA2AE71D1ECEFF004BE40F38A0030000
    }
]
do http://re-bol.com/rebgui.r
do login: [
    userpass: request-password
    if (length? userpass) < 2 [quit]
    posp-database: to-block read %posp.db
    logged-in: false
    foreach user posp-database [
        if (userpass/1 = user/1) and (userpass/2 = user/2) [
            logged-in: true
        ]
    ]
    either logged-in = true [] [
        alert "Incorrect Username/Password"
        do login
    ]
```

```
]
calculate-totals: does [
    tax: .06
    subtotal: 0
    foreach [item booth price] pos-table/data [
        subtotal: subtotal + to decimal! price
    ]
    set-text subtotal-f subtotal
    set-text tax-f (round/to (subtotal * tax) .01)
    set-text total-f (round/to (subtotal + (subtotal * tax)) .01)
    set-focus barcode
]
add-new-item: does [
    if ("" = copy f1/text) or ("" = copy f2/text) or (error? try [
        to-decimal copy f3/text
    ]) [
        alert trim/lines {You must enter a proper Item Description,
            Booth Number, and Price.}
        return
    ]
    pos-table/add-row/position reduce [
        copy f1/text copy f2/text copy f3/text
    ] 1
    calculate-totals
]
print-receipt: does [
    if empty? pos-table/data [
        alert "There's nothing to print." return
    ]
    html: copy rejoin [
        {<html><head><title>Receipt</title></head><body>
        <table width=40% border=0 cellpadding=0><tr><td>
        <h1>Business Name</h1>
        123 Road St.<br>
        City, State 98765<br>
        123-456-7890
        </td></tr></table><br><br>
        <center><table width=80% border=1 cellpadding=5>
        <tr>
        <td width=60%><strong>Item</strong></td>
        <td width=20%><strong>Booth</strong></td>
        <td width=20%><strong>Price</strong></td></tr>}
    ]
    foreach [item booth price] pos-table/data [
        append html rejoin [
            {<tr><td width=60%>} item
            {</td><td width=20%>} booth
            {</td><td width=20%>} price {</td></tr>}
        ]
    ]
    append html rejoin [
        {<tr><td width=60%></td><td width=20%><strong>SUBTOTAL:
        </strong></td><td width=20%><strong>}
        copy subtotal-f/text
        {</strong></td></tr>}
    ]
    append html rejoin [
        {<tr><td width=60%></td><td width=20%><strong>TAX:
        </strong></td><td width=20%><strong>}
        copy tax-f/text
        {</strong></td></tr>}
    ]
    append html rejoin [
        {<tr><td width=60%></td><td width=20%><strong>TOTAL:
```

```
        </strong></td><td width=20%><strong>}
        copy total-f/text
        {</strong></td></tr>}
    ]
    append html rejoin [
        {</table><br>Date: <strong>} now/date
        {</strong>, Time: <strong>} now/time
        {</strong>, Salesperson: } userpass/1
        {</center></body></html>}
    ]
    write/append to-file saved-receipt: rejoin [
        %./receipts/
        now/date "_"
        replace/all copy form now/time ":" "-"
        "+" userpass/1
        ".html"
    ] html
    browse saved-receipt
]
save-receipt: does [
    if empty? pos-table/data [
        alert "There's nothing to save." return
    ]
    if allow-save = false [
        unless true = resaving: question trim/lines {
            This receipt has already been saved.  Save again?
        } [
            if true = question "Print another copy of the receipt?" [
                print-receipt
            ]
            return
        ]
    ]
    if resaving = true [
        resave-file-to-delete: copy ""
        display/dialog "Delete" compose [
            text 150 (trim/lines {
                IMPORTANT - DO NOT MAKE A MISTAKE HERE!
                Since you've made changes to an existing receipt,
                you MUST DELETE the original receipt.  The original
                receipt will be REPLACED by the new receipt (The
                original data will be saved in an audit history file,
                but will not appear in any future seaches or totals.)
                Please CAREFULLY choose the original receipt to DELETE:
            })
            return
            tl1: text-list 150 data [
                "I'm making changes to a NEW receipt that I JUST SAVED"
                "I'm making changes to an OLD receipt that I've RELOADED"
            ] [
                resave-file-to-delete: tl1/selected
                hide-popup
            ]
            return
            button -1 "Cancel" [
                resave-file-to-delete: copy ""
                hide-popup
            ]
        ]
        if resave-file-to-delete = "" [
            resaving: false
            return
        ]
        if resave-file-to-delete = trim/lines {
```

```
            I'm making changes to a NEW receipt that I JUST SAVED
        }  [
            the-file-to-delete: saved-file
        ]
        if resave-file-to-delete = trim/lines {
            I'm making changes to an OLD receipt that I've RELOADED
        } [
            the-file-to-delete: loaded-receipt
        ]
        if not question to-string the-file-to-delete [return]
        write %./receipts/deleted--backup.txt read %./receipts/deleted.txt
        write/append %./receipts/deleted.txt rejoin [
            newline newline newline
            to-string the-file-to-delete
            newline newline
            read the-file-to-delete
        ]
        delete the-file-to-delete
        alert "Original receipt has been deleted, and new receipt saved."
        resaving: false
    ]
    if true = question "Print receipt?" [print-receipt]
    saved-data: mold copy pos-table/data
    write/append to-file saved-file: copy rejoin [
        %./receipts/
        now/date "_"
        replace/all copy form now/time ":" "-"
        "+" userpass/1
        ".txt"
    ] saved-data
    splash compose [
        size: 300x100
        color: sky
        text: (rejoin [{^/      *** SAVED ***^/^/      } saved-file {^/}])
        font: ctx-rebgui/widgets/default-font
    ]
    wait 1
    unview
    allow-save: false
    if true = question "Clear and begin new receipt?" [clear-new]
]
load-receipt: does [
    if error? try [
        loaded-receipt: to-file request-file/file/filter %./receipts/
            ".txt" "*.txt"
    ] [
        alert "Error selecting file"
        return
    ]
    if find form loaded-receipt "deleted" [
        alert "Improper file selection"
        return
    ]
    if error? try [loaded-receipt-data: load loaded-receipt] [
        alert "Error loading data"
        return
    ]
    insert clear pos-table/data loaded-receipt-data
    pos-table/redraw
    calculate-totals
    allow-save: false
]
search-receipts: does [
    search-word: copy request-value/title "Search word:" "Search"
```

```
    ; if search-word = none [return]
    found-files: copy []
    foreach file read %./receipts/ [
        if find (read join %./receipts/ file) search-word [
            if (%.txt = suffix? file) and (file <> %deleted.txt) [
                append found-files file
            ]
        ]
    ]
    if empty? found-files [alert "None found" return]
    found-file: request-list "Pick a file to open" found-files
    if found-file = none [return]
    insert clear pos-table/data (
        load loaded-receipt: copy to-file join %./receipts/ found-file
    )
    pos-table/redraw
    calculate-totals
    allow-save: false
]
clear-new: does [
    if allow-save = true [
        unless (true = question "Erase without saving?") [return]
    ]
    foreach item [barcode f1 f2 f3 subtotal-f tax-f total-f] [
        do rejoin [{clear } item {/text show } item]
    ]
    clear head pos-table/data
    pos-table/redraw
    allow-save: true
]
change-appearance: does [
    request-ui
    if true = question "Restart now with new scheme?" [
        if allow-save = true [
            if false = question "Quit without saving?" [return]
        ]
        write %scheme_has_changed ""
        launch %pos.r  ; EDIT
        quit
    ]
]
title-text: "Point of Sale System"
if system/version/4 = 3 [
    user32.dll: load/library %user32.dll
    get-tb-focus: make routine! [return: [int]] user32.dll "GetFocus"
    set-caption: make routine! [
        hwnd [int]
        a [string!]
        return: [int]
    ] user32.dll "SetWindowTextA"
    show-old: :show
    show: func [face] [
        show-old [face]
        hwnd: get-tb-focus
        set-caption hwnd title-text
    ]
]
allow-save: true
resaving: false
saved-file: ""
loaded-receipt: ""
screen-size: system/view/screen-face/size
cell-width: to-integer (screen-size/1) / (ctx-rebgui/sizes/cell)
cell-height: to-integer (screen-size/2) / (ctx-rebgui/sizes/cell)
```

```
table-size: as-pair cell-width (to-integer cell-height / 2.5)
current-margin: ctx-rebgui/sizes/margin
top-left: as-pair negate current-margin negate current-margin
display/maximize/close "POS" [
    at top-left #L main-menu: menu data [
        "File" [
            "     Print      " [print-receipt]
            "     Save       " [save-receipt]
            "     Load       " [load-receipt]
            "     Search     " [search-receipts]
        ]
        "Options" [
            "     Appearance     " [change-appearance]
        ]
        "About" [
            "     Info     " [
                alert trim/lines {
                    Point of Sale System.
                    Copyright © 2010 Nick Antonaccio.
                    All rights reserved.
                }
            ]
        ]
    ]
    return
    barcode: field #LW tip "Bar Code" [
        parts: parse/all copy barcode/text " "
        set-text f1 parts/1
        set-text f2 parts/2
        set-text f3 parts/3
        clear barcode/text
        add-new-item
    ]
    return
    f1: field tip "Item"
    f2: field tip "Booth"
    f3: field tip "Price (do NOT include '$' sign)" [
        add-new-item
        set-focus add-button
    ]
    add-button: button -1 "Add Item" [
        add-new-item
        set-focus add-button
    ]
    button -1 #OX "Delete Selected Item" [
        remove/part find pos-table/data pos-table/selected 3
        pos-table/redraw
        calculate-totals
    ]
    return
    pos-table: table (table-size) #LWH options [
        "Description" center .6
        "Booth" center .2
        "Price" center .2
    ] data []
    reverse
    panel sky #XY data [
        after 2
        text 20 "Subtotal:" subtotal-f: field
        text 20 "     Tax:" tax-f: field
        text 20 "   TOTAL:" total-f: field
    ]
    reverse
    button -1 #XY "Lock" [do login]
```

```
    button -1 #XY "New" [clear-new]
    button -1 #XY "SAVE and PRINT" [save-receipt]
    do [set-focus barcode]
] [question "Really Close?"]
do-events


#!./rebol276 -cs
REBOL [title: "CGI Forum"]
print {content-type: text/html^/}
switch system/options/cgi/request-method [
    "POST" [
        cgi-data: copy ""  cgi-buffer: copy ""
        while [positive? read-io system/ports/input cgi-buffer 16380] [
            append cgi-data cgi-buffer clear cgi-buffer
        ]
    ]
    "GET" [cgi-data: system/options/cgi/query-string]
]
submitted: decode-cgi cgi-data
if submitted/2 = "rss" [
    write/append %bb.db ""
    bbs: load %bb.db
    reverse bbs
    stickycount: 0
    foreach topic bbs [foreach item topic [
        if find item {<i>STICKY:} [stickycount: stickycount + 1]
    ]]
    print trim {<?xml version="1.0"?>
        <rss version="2.0">
        <channel>
        <title>REBOL Forum</title>
        <description>Recent REBOL Forum Topics</description>
        <link>http://rebolforum.com</link>
    }
    count: 1
    foreach item (at bbs (stickycount + 1)) [
        rss-title: copy item/1
        rss-title: replace/all rss-title {&} {&amp;}
        rss-title: replace/all rss-title {"} {&quot;}
        rss-title: replace/all rss-title {'} {&apos;}
        rss-title: replace/all rss-title {<} {&lt;}
        rss-title: replace/all rss-title {>} {&gt;}
        rss-description: rejoin [
            (copy/part (pick item ((length? item) - 2)) 4096)
            ", Posted by: " (pick item ((length? item) - 1))
            "  " (pick item (length? item))
        ]
        rss-description: replace/all rss-description {&} {&amp;}
        rss-description: replace/all rss-description {"} {&quot;}
        rss-description: replace/all rss-description {'} {&apos;}
        rss-description: replace/all rss-description {<} {&lt;}
        rss-description: replace/all rss-description {>} {&gt;}
        print trim rejoin [
            {<item>
            <title>} rss-title {</title>
            <description>} rss-description {</description>
            <link>http://rebolforum.com/index.cgi?
                f=printtopic&amp;topicnumber=}
            ((length? bbs) + 1 - count - stickycount)
            {&amp;archiveflag=new</link></item>}
        ]
        count: count + 1
        if count > 10 [break]
```

```
    ]
    print trim {
        </channel>
        </rss>
    }
    quit
]
print {<HTML><HEAD><TITLE>REBOL Forum</TITLE>
    <link rel="alternate" type="application/rss+xml"
        title="rebolforum.com rss feed" href="./index.cgi?f=rss" /></HEAD>
<BODY bgColor=#808080><CENTER>
<TABLE border=0 cellPadding=20 cellSpacing=2 height=100% width=85%>
<TR><TD bgColor=white vAlign=top>}
footer: {</TD></TR></TABLE></CENTER></BODY></HTML>}
unless exists? %index.html [
    write %index.html {<html><head><title></title>
        <META HTTP-EQUIV="REFRESH" CONTENT="0; URL=./index.cgi"></head>
        <body bgcolor="#FFFFFF"></body></html>}
]
if submitted/2 = "downloadreader" [
    print trim/lines {
        <center><a href="./index.cgi?f=home">Home</a></center><br>
        (Open the REBOL console and "do" this page URL)<br><br><pre>
    }
    print decompress #{
789CC5544B8BDB3010BEFB570C82425C70B4CE52167C6B69F7B4A5901E430EB2
3DD9A8B525571AC7D996FEF74AF2234EB2DBC7A9821069E69BC7F78DA5F58777
9F1E6043922ACC80ADC3F15E9BB686358A120DDB46912D0CA2CAC03E59C29A1F
2476BCB7253B5120B7F23B4202E9CDCDD1FD22D28D2C6C06856E9E60B3052805
895C589C4C51D436CEE80CA5460B9B08DCBA080BB68BC84A8BD2E1929DAC100C
7E6BD15238F04080FB6D88F38B3D78F447B4563CA2CD183006AF0CE6BADA797E
4B3A52C0BA138A62DFD79F2A0E4DF9259A065539F4073B692CF58780E81BA594
FBC8A1CF1E1A1C76AF3BE70D7B9172C2230DA01F3F4F0091464E142F2C14A808
4D10162AF1A45B1A5A092A2F7ADDE33E5F61B4B543FD0C7CEEA492AEBBC542D8
A411D2C0C2E856959CF418C953780DCBD59B18D278B2ADE2D8CD6F757BBCBB8D
830223D99308F5A863DF3C639367D46F334040891A81648D5B58080297E8EB49
562754A10856F12CF74CE4B18C9BEE172DD50568D60828EC2AA966FFBE2A7363
0FA52FDD576958F277CB65FC5DAA6D74BD3B1FF3C868728F033F0509373CE144
84CE88E6CFC3BB7B7978219D416A8D0ADB9C14B0F7BA53E1E6ACD17F5CD39560
33753B23099FBF5A561C9EBD5F41C5CFCE392574F170EF202FDFB57139B225EC
899A8CF319ACD035CFF36599CFC439A7F0D6147B79C0F23F9048445FFB5FC88C
21D78CFCEB0A23AD733EFDDBE8DECD68FB0BE01DA7BF9F050000
    }
    print {</pre>}
    print footer
    quit
]
if submitted/2 = "api" [
    print {<pre>}
    print decompress #{
789CB5503D0BC23010DDFB2B8E0E3A15F74217C14E82E01A3AC4F62A912629F9
4044FADF4DD3D8D6A238883784BCBB07EFE3B8DB1EF6400C330DA6101F3DCCA5
B21C722B4AC3A4D0711145B4AA14EAD62147ABDD0548DDB312AB1A187E46B6AC
14969F50858DD5A804E5182047ADE9190B2010811B7E4B5AA94D0A0A2F920920
7EDBCFBD0EDAD94CB79BCEAB9956D6BDD19F517B032367743423045B232760CF
28FCAB90569BD26A23394CA91556B64420EB3EC4334C110D6509BC7EEA69D18A
B7FD5B434EACFB967859DE6B6D7FEDE30149C2C94C64020000
    }
    print {</pre>}
    print footer
    quit
]
if submitted/2 = "source" [
    print trim/lines {
```

```
        <center><a href="./index.cgi?f=home">Home</a></center><br>
        (Open the REBOL console and "do" this page URL)<br><br><pre>
    }
    prin {REBOL []  editor decompress }
    print compress read %index.cgi
    print </pre>
    print footer
    quit
]
write/append %archive.db ""
write/append %bb.db ""
bbs: load %bb.db
displaylength: 49
captchacheck: does [
    if submitted/10 <> (trim/all submitted/12) [
        print {<strong>Incorrect Captcha Text</strong><br><br>
            Click the [BACK] button in your browser to try again.
            <br><br><a href="./index.cgi?f=home">Home</a><br>
        }
        print footer
        quit
    ]
]
random/seed now/time  password: copy []  wrds: first system/words
foreach ch mold pick wrds (random length? wrds) [append password ch]
if submitted/2 = "addnew" [
    if (submitted/4 = "") or (submitted/6 = "") or (submitted/8 = "") [
        print {
            <strong>Incomplete submission</strong><br><br>
            Click the [BACK] button in your browser to try again.
            <br><br><a href="./index.cgi?f=home">Home</a><br>
        }
        print footer
        quit
    ]
    captchacheck
    make-dir %./history/
    save rejoin [
        %./history/ now/year "_" now/month "_" now/day "_"
        (replace/all form now/time ":" "_") ".db"
    ] bbs
    entry: copy []
    append entry submitted/6  ; topic
    submitted-message: replace/all submitted/8 {REBOL []}
        {R E B O L []}
    submitted-message: replace/all submitted-message {REBOL[]}
        {R E B O L []}
    append entry submitted-message  ; message
    append entry submitted/4  ; name
    append entry form (now + 3:00)
    append/only tail bbs entry
    if (length? bbs) > displaylength [
        write/append %archive.db mold bbs/1
        remove head bbs
    ]
    reverse bbs
    foreach topic (copy bbs) [
       foreach item topic [
           if find item {<i>STICKY:} [
               move/to (find/only bbs topic) 1
           ]
       ]
    ]
    reverse bbs
```

```
    save %bb.db bbs
    print {<strong>New Topic Added</strong>}
    print footer
    wait :00:02
    print {<META HTTP-EQUIV="REFRESH"
        CONTENT="0; URL=./index.cgi?f=added">}
    quit
]
if submitted/2 = "printtopic" [
    if submitted/6 = "archive" [bbs: load %archive.db]
    current-topic: copy pick bbs (to-integer submitted/4)
    print rejoin [
        {<center><a href="./index.cgi?f=home">Home</a>  
        <a href="./index.cgi?f=printarchive">Archive</a></center><br>
        <center><table border=0 cellPadding=0 cellSpacing=0 width=90%>
        <tr><td><hr><br><font size=5>} (current-topic/1) {</font><br><br>}
    ]
    foreach [message name timestamp] (at current-topic 2) [
        replace/all message newline {  <br>  }
        message2: copy message
        append message { }
        replace/all message2 {    } {    }
        parse/all message [any [thru "http://" copy link to { } (
            replace message2 (rejoin [{http://} link]
        ) (rejoin [
            { <a href="} {http://} link {" target=_blank>http://}
            link {</a> }
        ])] to end]
        print rejoin [
            message2 {<br><br><font size=1>posted by:   }
            name {       }
            timestamp {</font><br><br><hr><br>}
        ]
    ]
    if submitted/6 = "new" [
        print rejoin [
            {<FORM method="post" ACTION="./index.cgi">
                <input type=hidden name="addresponse" value="addresponse">
                <input type=hidden name="topicnumber" value="} submitted/4
                {">
                Name: <br>
                <input type=text size="60" name="username"><br><br>
                Message: <br>
                <textarea name=message rows=5 cols=50></textarea><br><br>
                Type this quoted captcha text below: "<strong>} password
                {</strong>"<br><br>
                <input type=text size="60" name="pass"><br><br>
                <input type="hidden" name="password" value="} password {">
                <input type="submit" name="submit"
                    value="submit response">
            </FORM></td></tr>
            <tr><td align=right><a href="./index.cgi?f=home">Home</a></td>
            </tr></table></center>}
        ]
    ]
    print footer
    quit
]
if submitted/2 = "search" [
    print {<center><a href="./index.cgi?f=home">Home</a></center><br><hr>
        <br>}
    search-all: does [
        foreach topic bbs [
            foreach [message name timestamp] (at topic 2) [
```

```
                if any [
                    (find message submitted/4) (find name submitted/4)
                ] [
                    replace/all message newline {  <br>  }
                    message2: copy message
                    append message { }
                    replace/all message2 {    } {    }
                    parse/all message [any [
                        thru "http://" copy link to { } (
                            replace message2 (rejoin [{http://} link]) (
                                rejoin [
                                    { <a href="} {http://} link
                                    {" target=_blank>http://} link {</a> }
                                ]
                            )
                        )
                    ] to end]
                    print rejoin [
                        {<font size=5>} archive-note topic/1
                        {</font><br><br>}
                        message2 {<br><br><font size=1>posted by:   }
                        name {       }
                        timestamp {</font><br><br><hr><br>}
                    ]
                ]
            ]
        ]
    ]
    archive-note: ""  search-all
    bbs: load %archive.db  archive-note: {(Archive) }  search-all
    print footer
    quit
]
if submitted/2 = "addresponse" [
    if (submitted/6 = "") or (submitted/8 = "") [
        print {
            <strong>Incomplete submission.</strong><br><br>
            Click the [BACK] button in your browser to try again.
            <br><br><a href="./index.cgi?f=home">Home</a><br>
        }
        print footer
        quit
    ]
    captchacheck
    save rejoin [
        %./history/ now/year "_" now/month "_" now/day "_"
        (replace/all form now/time ":" "_") ".db"
    ] bbs
    topicnumber: to-integer submitted/4    ; topic number
    submitted-message: replace/all submitted/8 {REBOL [}
        {R E B O L [}
    submitted-message: replace/all submitted-message {REBOL[}
        {R E B O L [}
    append bbs/:topicnumber submitted-message    ; message
    append bbs/:topicnumber submitted/6    ; name
    append bbs/:topicnumber form (now + 3:00)
    move/to (at bbs topicnumber) (length? bbs)  ; sort by most recent
    responded-topic: (first last bbs)
    reverse bbs  ; move sticky messages to top
    foreach topic (copy bbs) [
       foreach item topic [
            if find item {<i>STICKY:} [
                move/to (find/only bbs topic) 1
            ]
```

```
        ]
    ]
    reverse bbs
    save %bb.db bbs
    print rejoin [
        {<strong>Response added to "} responded-topic {"</strong>}
    ]
    print footer
    wait :00:03
    print {<META HTTP-EQUIV="REFRESH"
        CONTENT="0; URL=./index.cgi?f=added">}
    quit
]
either submitted/2 = "printarchive" [
    archiveflag: "archive"
    bbs: load %archive.db
    head-text: "Archive"
] [
    archiveflag: "new"
    head-text: "REBOL Forum"
]
print rejoin [{
    <center><font size=6>} head-text {</font><br>
    <FORM method="post" ACTION="./index.cgi">
        <input type=hidden name="search" value="search">
        <input type=text size="50" name="searchtext">
        <input type="submit" name="submit" value="search">
    </FORM>
    <table border=1 cellPadding=5 cellSpacing=0 width=90%>
}]
counter: 1
reverse bbs
foreach bb bbs [
    print rejoin [
        {<tr><td width=65%>  
            <a href="./index.cgi?f=printtopic&topicnumber=}
        ((length? bbs) + 1 - counter)
        {&archiveflag=} archiveflag {">} bb/1
        {</a></td><td width=5%>} ((length? bb) - 1 / 3)
        {</td><td width=30%><font size=1>} (last bb)
        {, } pick bb ((length? bb) - 1) {</font></td></tr>}
    ]
    counter: counter + 1
    if counter > displaylength [break]
]
message-count: 0
try [foreach record bbs [
    message-count: message-count + ((length? record) - 1 / 3)
]]
either submitted/2 <> "printarchive" [
    print rejoin [
        {<tr><td>   <a href="./index.cgi?f=printarchive">
            ARCHIVED MESSAGES</a></td>}
        {<td colspan=2 align=center><strong>} message-count
        { active messages </strong></td></tr>}
    ]
] [
    print rejoin [
        {<td colspan=2>   <strong>} message-count
        { archived messages</strong></td>}
        {<td align=center>   
            <a href="./index.cgi?f=home">Home</a></td></tr>}
    ]
]
```

```
print rejoin [{
    </table><br>
    <FORM method="post" ACTION="./index.cgi">
        <table border=0 cellPadding=0 cellSpacing=4 width=50%>
        <input type=hidden name="addnew" value="addnew">
        <tr><td width=10%>Name:</td>
        <td><input type=text size="65" name="username"></td></tr>
        <tr><td width=10%>New Topic:</td>
        <td><input type=text size="65" name="subject"></td></tr>
        <tr><td width=10%>Message:</td>
        <td><textarea name=message rows=5 cols=50></textarea></td></tr>
        <tr><td width=10%>Captcha:</td>
        <td>Type this quoted captcha text below: "<strong>} password
            {</strong>"<br>
        <input type=text size="65" name="pass"><br>
        <input type="hidden" name="password" value="} password
             {"></td></tr>
        <tr><td></td><td>
        <input type="submit" name="submit" value="submit new topic">
        </td></tr></table>
    </FORM>
    <table border=1 cellPadding=5 cellSpacing=0 width=88%>
            <tr><td align=center><font size=2>
        <a href="./index.cgi?f=home"><strong>Home</strong></a>  
        <a href="./index.cgi?f=rss">RSS Feed</a>  
        <a href="./index.cgi?f=downloadreader">Reader</a>  
        <a href="./index.cgi?f=source">Source</a>  
        <a href="http://synapse-ehr.com/forums/forumdisplay.php?3-Rebol"
            target=_blank>Graham's Forum</a>  
        <a href="http://www.rebol.org/aga-groups-index.r?world=r3wp"
            target=_blank>AltME</a>  
        <a href="http://mail.rebol.net/cgi-bin/mail-list.r" target=_blank>
            ML</a>  
        <a href="http://www.rebol.net/cgi-bin/r3blog.r" target=_blank>
            R3 Blog</a>  
        <a href="http://translate.google.com/translate?hl=en&sl=fr&
            u=http://www.digicamsoft.com/cgi-bin/rebelBB.cgi&
            ei=te4HTNCBGsKBlAfdnIyZDg&sa=X&oi=translate&ct=result&
            resnum=1&ved=0CCsQ7gEwAA&prev=/search%3Fq%3Drebelbb%26hl%3
            Den%26client%3Dopera%26hs%3DF5Z%26rls%3Den" target=_blank>
            RebelBB</a>  
        <a href="http://www.rebol.com/community.html" target=_blank>
            Community</a>  
        <a href="http://re-bol.com" target=_blank>Tutorial</a>
    </font></td></tr></table></center>
}]
print footer
quit


#!./rebol276 -cs
REBOL [title: "CGI Sitebuilder"]
print "content-type: text/html^/"
print [<HTML><HEAD><TITLE>"Sitebuilder"</TITLE></HEAD><BODY>]
read-cgi: func [/local data buffer][
    switch system/options/cgi/request-method [
        "POST" [
            data: make string! 1020
            buffer: make string! 16380
            while [positive? read-io system/ports/input buffer 16380][
                append data buffer
                clear buffer
            ]
        ]
```

```
        "GET" [data: system/options/cgi/query-string]
    ]
    data
]
submitted: decode-cgi submitted-bin: read-cgi
; if no data has been submitted, request user/pass:
if ((submitted/2 = none) or (submitted/4 = none)) [
    print [<strong>"W A R N I N G  -  "]
    print ["Private Server, Login Required:"</strong><BR><BR>]
    print [<FORM METHOD="post" ACTION="./sitebuilder.cgi">]
    print [" Username: " <input type=text size="50" name="name"><BR><BR>]
    print [" Password: " <input type=text size="50" name="pass"><BR><BR>]
    print [<INPUT TYPE="SUBMIT" NAME="Submit" VALUE="submit">]
    print [</FORM>]
    print {</BODY></HTML>} quit
]
; check user/pass every time - end program if incorrect:
username: submitted/2 password: submitted/4
either ((username = "username") and (password = "password")) or (
    (find submitted/2 {Content-Disposition: form-data;}) <> none) [
    ; if user/pass is ok, go on
][
    print "Incorrect Username/Password."
    print {</BODY></HTML>} quit
]
if not exists? %sitemap.r [
    write %sitemap.r {%Home []}
    write %Home {}
]
if not exists? %upload.cgi [
    write/binary/allow %upload.cgi to-binary decompress 64#{
eJyFV21v2zYQ/m7A/+GqoYUDTFHaAV2h2A7aNVsHpEixZhgGwxsoibbZSqJKUs2M
LP99d3yRZNnFBFgv5N3xubvnjvR3T84TxTNZvvjxJcS5nk5+u35zewOr9XTSKFEb
eMhlbXhtYrNveAqG/2OSnanKv5LHILKav7t7f7Ocv7t+/XY5v/v17uZ6GX0Uhmet
KAuuonniBueJE3lz+/bPJa4wnSjOijjfihQ2bZ3DKillzkoomGGQtZsNV+vVdAJ4
6Xth8h3ovTa8SmRjhKx1gqqI/0vLtYkrbnayAC9PV/Th9uNdNByhi4ynULHPHLRB
B7ZP4PnFi4tDIbf4WOzlD69Gcvc7UXJYNVILI77yK7AeCRmANlIZnYi6aY236ays
R6DoYk3D62Lo/LFMXnKmTs6u+8/Ba/TLNUXA+XwieBg6tY+dg17NP0hlOrFZKngu
Cx5XbWlEw5SJN1JVsbPp8uZUmnhIljCGsTGxs0YDPsWl0AZyDAo6zEtRxRnf+jcK
Qi3rOFf2UW78F75UouIxQUBkYVWylEIum72lLQ0JUjGwkWldXMFGoMFDbBB1ziSd
MxgnTJ9pVe3QWd9t0IP9Uw5mRQqfpKghiuPIidGCiWGiJAejjFAwtV9EPrDBXa+H
ESDfhpMYAT9pvc6KgMQFInVUkFVT8grRQL5jSnOD6Is2RzqiWkiji+D/a9T8vhQ1
H6rR2qS4Ctn4F/qUeLkuISg4oPQMPY8LoRtJkbMhS0mT07jrJJFtJU3JRB3BWa/a
s+Gyt44lSHc7h41lQP0oZMSuRlUoazTuEhFAAKv3PYn6aNOF1YAeRodtrte3bDlS
H5aY/b4Mbp7C6Gy5+X7UyAHfvZHeYdsJjk3NYMfqosRi7CQ7JwPagOSsq2f3MtY8
LF5fwN7YaKyn+0Gd90O+qk3VQM0qDl9Z2dp73HgUp9dJoSEiDkbg4XIRPQbMloto
EHceGSNh43upCphpXnKsrYFWRFLRgEdcmB122xmx7oqAYf871tpg+x5r+gujBbNR
CwndY8Dds/H+Yl1Pj/O9/oag3WJk9gmhPRmL0BUg+h6EnhzLONaGHjXOV7i6SvSx
caHpsukKnRPQUbsb0t26crDXHMXcpj11QbOdlOAfWlhRB6q3PJF1ibWF4XRatkdJ
TR1pthGKdLu5M5jZkJ0hwvXYnN883bbS2xiyhlCcsNF50FqVb5bTIYOJtQkry+EG
h9paIvn72nR7wqBhdIsRUL+9dvophPPQdIK32JHCUgl36Dan5vKtnfjU5n6w47mM
9nvxdHIJjZIZznRLYQci30u212B9wyU0yA1gWkG3GTYjwwvolve0JVN/KGxUlqvU
2DRXX5EIrcZjhVWWSmxFjR0isPl7W1+4S4vN3kpg9FVKsPAjJqkUSobJRDjCYDTN
jlqAXcD10w510uykkUmwPJ3cE5Ykw/WUte21xgpdhfqT7jzHL66WcwY7xTeL6DzR
/Sn2HPWvyP6CkNLLs4ZpvaAb0euZi8/CPaLlG5Z/plAMDsLzhC3nmVo+9ksaliG0
e1GY3eLVxVPI0BJXi+cHMgjJFF7o+cXFU2skoO0FHbHmSBVZb5e/f7i5ff0Wfrp9
/+Hm+u56nvhxq4u/Q+mfMUIa8lYptIoViUcPsxMaj1AEPD2hjev6SSS9Mpa68PSc
RvEVj+o26E6kOyC6/wuK27PN6mEQ6Ucn/xAB7vRbbhZ/ZyWrPy/9eBRiF63XtmxC
cAhPYgq62TeKJz7tXwz8v0F/TBDpl1aY/wDPpuhm7AwAAA==
} [read write execute read write execute read write execute]
    if error? try [call {chmod 755 ./upload.cgi}] [
        print {
            <center><table border="1" width=80% cellpadding="10"><tr><td>
            <strong>./upload.cgi</strong> has been created, but there was
```

```
            apparently a problem setting permissions for it.  Please be
            sure that upload.cgi is chmod to 755.<br><br><center>
            <a href="./sitebuilder.cgi?name=username&pass=password
                &submit=submit">
            Continue</a></center></td></tr></table></center></BODY></HTML>
        } quit
    ]
]
; if only user/pass has been entered, print main start page :
if submitted/6 = "submit" [
    ; write/append %sitemap.r ""  ; make sure it exists
    print rejoin [
        "<center>Path: " what-dir
        {<br><table border="1" width=80% cellpadding="10"><tr><td>}
    ]
    print rejoin [
        {<br>
        <FORM ACTION=
        "./upload.cgi" METHOD="post" ENCTYPE="multipart/form-data">
        Upload File: <INPUT TYPE="file" size="50" NAME="photo">
        <INPUT TYPE="submit" NAME="Submit" VALUE="Upload">
            <a href="./sitebuilder.cgi?name=username&pass=password
            &subroutine=listfiles">Files</a>
        </FORM>
        <FORM method="post" ACTION="./sitebuilder.cgi">
        <INPUT TYPE=hidden NAME=username VALUE="} submitted/2 {">
        <INPUT TYPE=hidden NAME=password VALUE="} submitted/4 {">
        <INPUT TYPE=hidden NAME=subroutine VALUE="edit">
        Create New Page:
        <INPUT TYPE=text size="50" name="file" value="">
        <INPUT TYPE="SUBMIT" NAME="Submit" VALUE="Submit">
        </FORM>}
    ]
    pages: sort read %.
    ; dont-show-suffixs: [
        %.html %.jpg %.gif %.png %.bmp %.rip %.exe %.pdf %.cgi %.php
        %.zip %.txt %.tpl %.r %.tgz
    ]
    ; remove-each page pages [find dont-show-suffixs (suffix? page)]
    remove-each page pages [find to-string page "/"]
    dont-show-files: [%rebol276 %sitemap %.ftpquota]
    remove-each page pages [find dont-show-files page]
    print "<hr><br>Edit Existing Pages:<br><br>"
    foreach page pages [
        if ((suffix? page) = none) [
            print rejoin [
                {<a href="./sitebuilder.cgi?name=username&pass=password
                    &subroutine=edit&file=}
                to-string page {">} to-string page {</a>    
                } ; <br>}
            ]
        ]
    ]
    print {<br><br><hr>}
    print {<a href="./sitebuilder.cgi?name=username&pass=password
        &subroutine=cleanedit&file=sitemap.r">Edit Site Map</a>       }
    print {<a href="./sitebuilder.cgi?name=username&pass=password
        &subroutine=buildsite">Build Site</a>       }
    print rejoin [
        {<a href="./} (to-string first load %sitemap.r)
        {.html" target=_blank>View Home Page</a>       }
    ]
    print {<a href="./sitebuilder.cgi?name=username&pass=password
        &subroutine=console">Console</a>       }
```

```
    print {<a href="./sitebuilder.cgi?name=username&pass=password
        &subroutine=instructions">Instructions</a>}
    print {<br></td></tr></table></center></BODY></HTML>} quit
]
; if constructed edit link has been submitted:
if submitted/6 = "edit" [
    write/append to-file rejoin [what-dir submitted/8] ""
    if not exists? %./openwysiwyg/scripts/wysiwyg.js [
        write/binary
            %./openwysiwyg.rip read/binary
            http://re-bol.com/openwysiwyg.rip
        print {
            <center><table border="1" width=80% cellpadding="10"><tr><td>
            <center><strong>INITIAL SETUP: PLEASE RELOAD THIS PAGE AFTER
                FILES HAVE BEEN UNPACKED...</strong>
            </center></td></tr></table></center></BODY></HTML>
        }
        do %openwysiwyg.rip
        print {
            <center><table border="1" width=80% cellpadding="10"><tr><td>
            <center><strong>FILES HAVE BEEN UNPACKED: PLEASE RELOAD THIS
                PAGE NOW</strong>
            </center></td></tr></table></center></BODY></HTML>
        }
    ]
    ; backup (before changes are made):
    cur-time: to-string replace/all to-string now/time ":" "-"
    document_text: read to-file rejoin [what-dir submitted/8]
    make-dir %edit_history
    write to-file rejoin [
        what-dir "edit_history/"
        to-string (second split-path to-file submitted/8)
        "--" now/date "_" cur-time ".txt"
    ] document_text

    ; note the POST method in the HTML form:

    prin rejoin [
        {<script type="text/javascript"
            src="openwysiwyg/scripts/wysiwyg.js"></script>
        <script type="text/javascript">
            var full = new WYSIWYG.Settings();
            full.ImagesDir = "openwysiwyg/images/";
            full.PopupsDir = "openwysiwyg/popups/";
            full.CSSFile = "openwysiwyg/styles/wysiwyg.css";
            full.Width = "85%";
            full.Height = "250px";
            WYSIWYG.attach('all', full);
        </script>}
        {<center><strong>Be sure to SUBMIT when done:</strong><BR><BR>
        <FORM method="post" ACTION="./sitebuilder.cgi">
        <INPUT TYPE=hidden NAME=username VALUE="} submitted/2 {">
        <INPUT TYPE=hidden NAME=password VALUE="} submitted/4 {">
        <INPUT TYPE=hidden NAME=subroutine VALUE="save">
        <INPUT TYPE=hidden NAME=path VALUE="} submitted/8 {">
        <textarea id="textarea1" name="test1" cols="100" rows="15"
            name="contents">}
        replace/all document_text "</textarea>" "<\/textarea>"
        {</textarea>
        <a href="./sitebuilder.cgi?name=username&pass=password
            &subroutine=listfiles-popup" target=_blank><FONT size=1>Files
            </FONT></a><BR><BR>
        <INPUT TYPE="SUBMIT" NAME="Submit" VALUE="Submit">
        </FORM></center></BODY></HTML>}
```

```
    ]
    print {</BODY></HTML>} quit
]
; non-wysiwyg edit:
if submitted/6 = "cleanedit" [
    write/append to-file rejoin [what-dir submitted/8] ""
    ; backup (before changes are made):
    cur-time: to-streaaaeading replace/all to-string now/time ":" "-"
    document_text: read to-file rejoin [what-dir submitted/8]
    make-dir %edit_history
    write to-file rejoin [
        what-dir "edit_history/"
        to-string (second split-path to-file submitted/8)
        "--" now/date "_" cur-time ".txt"
    ] document_text

    ; note the POST method in the HTML form:

    prin rejoin [
        {<center><strong>Be sure to SUBMIT when done:</strong><BR><BR>
        <FORM method="post" ACTION="./sitebuilder.cgi">
        <INPUT TYPE=hidden NAME=username VALUE="} submitted/2 {">
        <INPUT TYPE=hidden NAME=password VALUE="} submitted/4 {">
        <INPUT TYPE=hidden NAME=subroutine VALUE="save">
        <INPUT TYPE=hidden NAME=path VALUE="} submitted/8 {">
        <textarea id="textarea12" name="test2" cols="100" rows="15"
            name="contents">}
        replace/all document_text "</textarea>" "<\/textarea>"
        {</textarea><br>
        <a href="./sitebuilder.cgi?name=username&pass=password
            &subroutine=listfiles-popup" target=_blank><FONT size=1>Files
            </FONT></a><BR><BR>
        <INPUT TYPE="SUBMIT" NAME="Submit" VALUE="Submit">
        </FORM></center></BODY></HTML>}
    ]
    print {</BODY></HTML>} quit
]
; if edited file text has been submitted:
if submitted/6 = "save" [
    ; save newly edited document:
    write
        (to-file rejoin [what-dir submitted/8])
        (replace/all submitted/10 "<\/textarea>" "</textarea>")
    either (submitted/8 <> "sitemap.r") and (
        submitted/8 <> (to-string first load %sitemap.r)
    ) [
        print {<center><strong>Document Saved</strong><br><br>}
        recurse-sitemap: func [page] [
            append sitemap-pages page/1
            if not (page/2 = []) [
                foreach block page/2 [recurse-sitemap block]
            ]
        ]
        sitemap-pages: copy []
        recurse-sitemap load %sitemap.r
        prin {<table border="1" width=80% cellpadding="10"><tr><td>
            <center>Now ADD this page as a SUB-PAGE of another in
            your site map:<br><br>
        }
        foreach page sitemap-pages [
            prin rejoin [
                {<a href="./sitebuilder.cgi?name=username&pass=password
                    &subroutine=addsitemap&newpage=}
                submitted/8 {&existingpage=} page {">} page {</a>       }
```

```
            ]
        ]
        print {
            <br><br>If you've ALREADY added this page to your site map,
                or if you do not want it in your site map
            <a href="./sitebuilder.cgi?name=username&pass=password
                &submit=submit"><strong>click here</strong></a>
            </center><br></td></tr></table></center>
        }
    ] [
        print {<html><head><META HTTP-EQUIV="REFRESH"
            CONTENT="0; URL=./sitebuilder.cgi?name=username&pass=password
            &submit=submit"></head>}
    ]
    print {</BODY></HTML>} quit
]
; If page has been added to site map via link:
if submitted/6 = "addsitemap" [
    recurse-add-sitemap: func [page] [
        if page/1 = (to-file submitted/10) [
            new-block: copy []
            append new-block (to-file submitted/8)
            append/only new-block []
            insert/only page/2 new-block
        ]
        if not (page/2 = []) [
            foreach block page/2 [recurse-add-sitemap block]
        ]
    ]
    recurse-add-sitemap new-site-map: load %sitemap.r
    save %sitemap.r new-site-map
    print {<html><head><META HTTP-EQUIV="REFRESH"
        CONTENT="0; URL=./sitebuilder.cgi?name=username&pass=password
        &submit=submit"></head>}
]
; If file upload has been submitted:
if ((find submitted/2 {Content-Disposition: form-data;}) <> none) [
    decode-multipart-form-data: func [
        p-content-type
        p-post-data
        /local list ct bd delim-beg delim-end non-cr non-lf non-crlf
            mime-part
    ] [
        list: copy []
        if not found? find p-content-type "multipart/form-data" [
            return list
        ]
        ct: copy p-content-type
        bd: join "--" copy find/tail ct "boundary="
        delim-beg: join bd crlf
        delim-end: join crlf bd
        non-cr:     complement charset reduce [ cr ]
        non-lf:     complement charset reduce [ newline ]
        non-crlf:   [ non-cr | cr non-lf ]
        mime-part:  [
            ( ct-dispo: content: none ct-type: "text/plain" )
            delim-beg ; mime-part start delimiter
            "content-disposition: " copy ct-dispo any non-crlf crlf
            opt [ "content-type: " copy ct-type any non-crlf crlf ]
            crlf ; content delimiter
            copy content
            to delim-end crlf ; mime-part end delimiter
            ( handle-mime-part ct-dispo ct-type content )
        ]
```

```
        handle-mime-part: func [
            p-ct-dispo
            p-ct-type
            p-content
            /local tmp name value val-p
        ] [
            p-ct-dispo: parse p-ct-dispo {;="}
            name: to-set-word (select p-ct-dispo "name")
            either (none? tmp: select p-ct-dispo "filename")
                   and (found? find p-ct-type "text/plain") [
                value: content
            ] [
                value: make object! [
                    filename: copy tmp
                    type: copy p-ct-type
                    content: either none? p-content [none][copy p-content]
                ]
            ]
            either val-p: find list name
                [change/only next val-p compose [
                    (first next val-p) (value)]
                ]
                [append list compose [(to-set-word name) (value)]]
        ]
        use [ct-dispo ct-type content] [
            parse/all p-post-data [some mime-part "--" crlf]
        ]
        list
    ]
    cgi-object: construct decode-multipart-form-data
        system/options/cgi/content-type copy submitted-bin
    ; probe cgi-object ; displays all parts of the submitted object
    ; Write file to server using the original filename, notify the user:
    the-file: last split-path to-file copy cgi-object/photo/filename
    write/binary the-file cgi-object/photo/content
    print {<center><a href="./sitebuilder.cgi?name=username&pass=password
        &submit=submit">Back to Sitebuilder</a><br>}
    print {<table width=80% border=1>}
    print {<tr><td width=100%><br><center>}
    print {
        <strong>UPLOAD COMPLETE</strong><br><br>
        <strong>Files currently in this folder:</strong><br><br>
    }
    folder: sort read %.
    foreach file folder [
        print [rejoin [
            {<a href="./} file {" target=_blank>} file "</a><br>"
        ]]
    ]
    print {<br></td></tr></table></BODY></HTML>}
    quit
]
; List existing files:
if submitted/6 = "listfiles" [
    print {<center><a href="./sitebuilder.cgi?name=username&pass=password
        &submit=submit">Back to Sitebuilder</a><br>}
    print {<table width=60% border=1 cellpadding="10">}
    print {<tr><td width=100%><br>}
    folder: sort read %.
    foreach file folder [
        print [rejoin [
            {    <a href="./sitebuilder.cgi?name=username
                &pass=password&subroutine=cleanedit&file=}
            file {">(edit)</a>    }
```

```
        ]]
        print [rejoin [
            {<a href="./} file {" target=_blank>} file {</a><br>}
        ]]
    ]
    print {<br></td></tr></table></center></BODY></HTML>}
    quit
]
if submitted/6 = "listfiles-popup" [
    print {<center><table width=80% border=1>}
    print {<tr><td width=100%><br><center>}
    folder: sort read %.
    foreach file folder [
        print [rejoin [{<a href="./} file {">} file "</a><br>"]]
    ]
    print {<br></center></td></tr></table></center></BODY></HTML>}
    quit
]
; Run REBOL console (for file and OS operations):
if submitted/6 = "console" [
    if not exists? %rebol276 [
        print "<center>REBOL version 276 required!</center><br>"
    ]
    print {<center><a href="./sitebuilder.cgi?name=username&pass=password
        &submit=submit">Back to Sitebuilder</a></center>}
    entry-form: [
        print {
            <CENTER><FORM METHOD="post" ACTION="./sitebuilder.cgi">
            <INPUT TYPE=hidden NAME=username VALUE="username">
            <INPUT TYPE=hidden NAME=password VALUE="password">
            <INPUT TYPE=hidden NAME=subroutine VALUE="console">
            <INPUT TYPE=hidden NAME=submit_confirm
                VALUE="command-submitted">
            <TEXTAREA COLS="100" ROWS="10" NAME="contents"></TEXTAREA>
            <BR><BR><INPUT TYPE="SUBMIT" NAME="Submit" VALUE="Submit">
            </FORM></CENTER></BODY></HTML>
        }
    ]
    if submitted/8 = "command-submitted" [
        write %commands.txt join "REBOL[]^/" submitted/10
        ; The "call" function requires REBOL version 2.76:
        call/output/error
            "./rebol276 -qs commands.txt"
            %conso.txt %conse.txt
        do entry-form
        print rejoin [
            {<CENTER>Output: <BR><BR>}
            {<TABLE WIDTH=80% BORDER="1" CELLPADDING="10"><TR><TD><PRE>}
            read %conso.txt
            {</PRE></TD></TR></TABLE><BR><BR>}
            {Errors: <BR><BR>}
            read %conse.txt
            {</CENTER>}
        ]
        quit
    ]
    do entry-form
]
; Build site:
if submitted/6 = "buildsite" [
    if not exists? %menu.tpl [
        write %menu.tpl decompress #{
        789CB556DB4EDB40107DE72B0623A456AA63C7691E00AFA5247649A484A46129
        E2A9F265B15D8C9DAE9D00ADFA41FDCBCEAEED90847015355292DD999D3D73E6
```

```
    CC1873D71EF7E8C5C4813E1D0D6172D61D0E7AA0A89A76DEEA699A4DEDD2F0B9
    A103E56E9AC7459CA56EA269CE8962ED98C268997DA7635B261DD0A163EDE063
    EEAA2AA02BF3E6711230FEBD888B8481AA5A20CD5AE96A8E1CDA81A828662AFB
    398F17A497A5054B0B95DECD18F8E5822805BB2DB4A8B84E8EC08F5C9EB382DC
    C46990DDE46AD3681B02860CD4A774A23A5FCF06DF88327143A63A18802BD01B
    9F50E7841265CA16CC4D641A1FC09E7357E4428C4FF78991A6F151C43BA51743
    07042FD5F57E9EE3BE7B18650BC6E137824B327E085E326747203CD480F95919
    F110E629269DC429DA3CD7BF0A79863B87B0F7E5C0B11D1BFE2003F206BC4993
    DCED98DDB17D015ED81371C99E2E1FDCA69D2E027193384C89CF443E2B1189A2
    809771BC8BE8E0B32499B84110A72131E4EA74E6FAE52A62711821934D5DDF57
    E0260E8A882807ED7D912A15378BEFA9F8B037C35788121120E4EC0E161D09A6
    C8664B78EF85482EDE00A93A6EB4CAD3DB3055676E2214658DB0B986B0A9BF1C
    220868880C6F9BAC15072D5F506DA53C48374110D807BF1869A1E5944EC727C7
    F8C38588B34BA2A088D96D43285BB1FAD9353335D712CAA8DC34114A7C4F7017
    2FC30F418826132CB79EA0C79397574934756B13A96CC41A49435BC5620E46C7
    95B3E7E64C08199748 85D0AA3385C1A873ECAC143ACA9134863F72EE8B581173
    D1D0F8310B3116E654F6FC7A16756D9F97D35A3EFF5B5CD551437F58DAD5CA96
    3A929535ACBAB02396CE97E5ABAAB72D6DC1C6EB1AED0194AE909F24ABCC0E7B
    D9326BE3E6F0C5FA5DE562F89A5AEDB2D45085AE1ECB5321B792A1DA074C8F57
    4E0F51BFB87A7B97F279DFFA9945F02C8BF7055DF273CF4211BCB81ACB38CBF1
    29E0949AA83AFCD102CCDC22DAE0FF125F6E2B00DE21918DBE5A0F87AA457A53
    D25AE7F63513BCD97E6B976D657855DD9B7C556FFE923231E56A716E6AF3BEB3
    1EB73CF71AD90B5AE2AF4EA7BD6DA6D76DF25498F295FD449847E647D3EA65B3
    3B2E4EC15F3074FD006814E770CE3C3845AF06749204A43D07CE72C6172C686C
    1D2F2FA747125A954893FFC3EDFC039C3D1A760A0A0000
    }
]
if not exists? %nomenu.tpl [
    write %nomenu.tpl decompress #{
    789CAD556D6FDA3010FECEAFB8A6AAB4490B09B47C288D2351921624681975 57
    F5D3E4246E92CD4B9863A0DDB41FB47FB973121874636DA7FA83E397E3B9E79E
    3B1FCE9E77D9A7B7131F06743C82C9F5E968D807C3B4AC9BC3BE6579D4AB2E8E
    9A3650C9B22255699E316159FE85E1361C7DE93A03BFE7B90E1DD291EF420387
    B3679A80B63C98A722E2F2A34A95E0609AF5B555D9369CB14F7B90283533F9D7
    79BA20FD3C533C53267D987108AB0D3114BF5756A2BE88130813260BAEC832CD
    A27C5998AD76A76DAC8006944E4CFFFDF5F00331262CE6A68F00D280FEE505F5
    2F2831A67CC19928037903DE5C321D0D69BFFB1D1A691DBDD57857F476E48356
    A6761F16059EB36E922FB884EF484EE4B20B8198F313D01666C4C3BC42ECC23C
    C3A8459AE15DC0C2CFB1CCF1A40BFB67C7BEE77BF00325283DA027AB54AFE19C
    5E7AB710C47D8D4BF6ED72E031ED9D221126D2382321D7F16C2012C3802097E8
    8BD8107221262C8AD22C26ED7277356361B54B781A27A864CBB60F0C58A6914A
    8871DC39D0A152ED597FA77AF21EC3D78C840688257F8045AF24A3F2D99ADE6B
    312A37DB9476330A041EAF705AB6EB4CB66472CB4A6390487E478CA68505C3EF
    9BBA8A0CD7198ECF6BE380155C270AB7C846E7C29FC270DC3BF73702490AE4CD
    7151C890180967 78DCFC348B11C96295236B826BEAE969BA53CCA7A55BE5FFAE
    1CAF2B253C2BBD355A1B05AD8574ACF5424565B42F016AD9EB7AD38CD6A08F5B
    C48CA94477880D6F28E74BBDEDA6FDCFC4E06B463D3372B82DE64B0ABCD5F9CF
    B4FC3DC28D52F943AABA31566AD52295E1E154E2E257D3DEA8C8DD374F3DFCBA
    15ADC2E9A0EDA387E69C617FAD1A225926C813C97EE3A4E5F6F3D983D4BF829F
    D0B6ED63A0495AC00D0FE00AAD9A003D21A0342840F282CB058F9A8EA5E1DCED
    07F5FCC050885A5AABFC736AFC02A1651F4AE3060000
    }
]
homepage: to-string first load %sitemap.r
current-path: rejoin [
    {<a href="./} homepage {.html">} homepage {</a>}
]
begin-recurse: true
recurse: func [page current-path][
    either begin-recurse = true [
        print-path: (to-string page/1)
    ] [
        print-path: rejoin [current-path { : } (to-string page/1)]
    ]
```

```
    begin-recurse: false
    either (page/2 = []) [
        constructed: replace (read %nomenu.tpl)
            {<!-- sitebuilder_content -->} (read to-file page/1)
        constructed: replace constructed
            {<!-- sitebuilder_title -->} (to-string page/1)
        constructed: replace constructed
            {<!-- sitebuilder_path -->} print-path
    ] [
        constructed: replace (read %menu.tpl)
            {<!-- sitebuilder_content -->}(read to-file page/1)
        link-list: copy {}
        foreach item page/2 [
            link-list: rejoin [
                link-list
                {<TR><TD style="border: solid" }
                {onmouseOver="this.bgColor='#FFFFFF'"; }
                {onmouseOut="this.bgColor='#D3D3D3'";> }
                {<CENTER>
                <FONT face="Arial, Verdana, MS Sans Serif" size=1>}
                {<A HREF="./} (to-string item/1) {.html">}
                (to-string item/1) {</A>}
                {</FONT></CENTER></TD></TR>}
                newline
            ]
        ]
        constructed: replace constructed
            {<!-- sitebuilder_links -->} link-list
        constructed: replace constructed
            {<!-- sitebuilder_title -->} (to-string page/1)
        constructed: replace constructed
            {<!-- sitebuilder_path -->} print-path
    ]
    write (to-file join page/1 ".html") constructed
    print page/1 print { ... DONE<br>}
    if not (page/2 = []) [
        if (to-string page/1) <> homepage [
            current-path: rejoin [
                current-path
                { : <a href="./} (to-string page/1) {.html">}
                (to-string page/1) {</a>}
            ]
        ]
        foreach block page/2 [recurse block current-path]
    ]
]
print {<center><table border="1" width=80% cellpadding="10"><tr><td>}
recurse mymap: load %sitemap.r current-path
print {</td></tr></table><br><a href="./sitebuilder.cgi?name=username
    &pass=password&submit=submit">Back to Sitebuilder</a></center>}
if not exists? %index.html [
    write %index.html rejoin [{
        <html>
        <head>
        <title></title>
        <META HTTP-EQUIV="REFRESH" CONTENT="0; URL=./}
            (to-string mymap/1) {.html">
        </head>
        <body bgcolor="#FFFFFF"><div id="divId">
        </div>
        </body>
        </html>
    }]
]
```

```
    quit
]
;  Print instructions:
if submitted/6 = "instructions" [
    print {<pre>}
    print instructions: {
    REBOL WEB SITE BUILDER:

    This script enables you to easily create, edit, and arrange HTML pages
    on your web site.  The first step is to create and/or upload page
    content.  The built-in WSYIWYG HTML editor allows you to layout pages
    visually, without having to write any code.  It works just like a word
    processor, except it runs directly in your browser, right on your web
    site.  You can adjust fonts, colors, and all essential formatting/
    layout options.  You can add tables, images, links, and other
    elements, all without writing any code.  Of course, if you prefer to
    write your own HTML code or copy/paste from other sources, you can
    switch instantly between visual and code view, for complete control
    and instant preview.  The built-in file upload allows you to upload
    any HTML files, scripts, images, or binary files of any sort, from any
    computer.  The template system automatically builds menu links to
    other pages, using a simple and quick site map layout that you
    specify, and the generated pages are all wrapped in templates that you
    can upload or create/edit directly online (2 generic templates are
    included to get you started).  Because this whole system runs in your
    browser, you can add pages, upload files, and edit site content
    instantly from any location, using any OS, without installing any
    software.


    CREATING AND EDITING PAGES:

    To create a new page for your web site, simply type in a name for the
    page and click the "Create New Page" link.  The visual editor will
    open, and you can begin editing content.  You can create new pages
    from scratch or copy/paste content directly into the visual view.
    Page names should NEVER CONTAIN SPACES (use underscores instead), and
    should not have any file extensions.  It's suggested that title case
    be used for page names (every important word capitalized).

    To upload images, scripts, or any other content that you've created on
    your local computer, simply click the "Choose" button and then the
    "Upload" button.

    You can edit any text or code on a page, whether it was created using
    the online editor, or uploaded, by simply clicking the file name in
    "Edit Existing Pages".  To add an image to a page, simply click the
    image icon and type in the file name of any image that you've
    uploaded.  Adding, editing, and previewing scripts is as simple as
    clicking the HTML/Text button, and using the built-in preview button.
    Centering and aligning content, changing font sizes, styles and
    colors, creating bulleted lists, and all typical operations function
    just like they do in most word processors.  Just select items and
    click on the icons to adjust your layout.  It's all very easy and
    intuitive, even for absolute beginners.


    THE SITE MAP:

    When you are done editing/uploading content pages, you will be asked
    if you want to add them as SUB-PAGES of other pages on your site.  A
    site map is automatically generated, and published pages contain
    automatically generated menus which enable users to easily navigate
    around your site.  The site map can be edited to easily arrange page
```

links on your site, based on the simple sub-page layout. Any page content added to the site map is also automatically framed in nicely designed templates, to give your entire site a consistent look and feel.

If you want to edit the order of pages in your site map, or add/remove pages from your site, click the "Edit Site Map" link. Starting with the home page, every entry in your site map is simply a BLOCK containing 2 items:

SOURCE FILE NAME:

This is a file name containing page content that you've created, which you want to appear in an .html page of the same name on your site. Content file names should be listed exactly as they were named when creating or uploading them, as they appear in the edit list. In the site map, all source file names MUST BE PRECEDED BY A PERCENT SYMBOL ("%").

SUB-PAGE LINKS:

Each page entry in your site map must be followed by a pair of square brackets. These brackets contain a block of links to other pages on the site, to appear in a link menu on the current page. The home page can contain as many sub-pages (menu links) as you want, and any sub-pages can contain as many sub-page links as you want, and so on, for as many levels deep as you want.

Your site map must have one and only one "home" page. It can be any file name you've created - typically "Home" (a %Home file is automatically created when this script is first run). This script automatically creates an index.html page that forwards to your home page, if no index.html exists. It's recommended that you keep your home page file named "Home".

Here's an example of how your site map would look if you only wanted one page to appear on your web site, labeled "Home.html":

```
%Home []
```

The file name (%Home above) contains the name of a source file to be processed (a content file that you've previously uploaded or created with the built-in editor). The block following it (empty above) contains the names of any SUB-PAGES that will be processed and automatically linked to it (none in the case above).

Below is an example of how the site map would look if you wanted a site made up of a home page and two sub-pages. Home.html, Page_One.html and Page_Two.html would all be created from the source files listed, and a menu bar would be automatically generated and placed on Home.html, linking to the 2 other pages. Neither Page_One.html nor Page_Two.html would contain any menu bars with links, because they don't contain any sub-pages:

```
%Home [
; your home page (index.html forwards to it)
    [%Page_One []]
    ; Page_One.html appears in the menu bar of Home.html
    [%Page_Two []]
    ; Page_Two.html appears in the menu bar of Home.html
]
```

The next example site map below contains a home page with 5 sub pages, the 3rd of which contains 2 sub pages, and the 2nd of that contains 3

sub pages. In the generated .html pages, link menus are only placed on pages which have sub-pages (i.e., only Home.html, Page_Three.html and Page_Three_B.html below would contain link menus):

```
%Home [
; your home page
    [%Page_One []]
    ; Page_One.html appears in the menu bar of Home.html
    [%Page_Two []]
    ; Page_Two.html appears in the menu bar of Home.html
    [%Page_Three [
    ; Page_Three.html appears in the menu bar of Home.html
        [%Page_Three_A []]
        ; Page_Three_A.html appears in menu bar of Page_Three_A.html
        [%Page_Three_B [
        ; Page_Three_B.html appears in menu bar of Page_Three_B.html
            [%Page_3_B_1 []]
            ; Page_3_B_1.html appears in menu bar of Page_Three_B.html
            [%Page_3_B_2 []]
            ; Page_3_B_2.html appears in menu bar of Page_Three_B.html
            [%Page_3_B_3 []]
            ; Page_3_B_3.html appears in menu bar of Page_Three_B.html
        ]]
    ]]
    [%Page_Four []]
    ; Page_Four.html appears in the menu bar of Home.html
    [%Page_Five []]
    ; Page_Five.html appears in the menu bar of Home.html
]
```

The key to understanding the site map is that any source file names followed by a link block will contain an auto-generated menu of links to those sub-pages in the created .html file. Pages without link blocks do not contain any sub-page links. They are simply wrapped in a template. Of course, you can manually link to any page that you've created, if you don't want any auto-generated link menus or template design to appear on your site. You can use this script to simply upload content, or to create/edit HTML/script files. If that's the case, you don't need to create a site map.

Once you've finished creating content files, and have arranged them into a site map, simply click the "Build Site" link. You can then view the generated web site by clicking the "View Home Page" link.

OTHER FEATURES:

If you need to perform any file or OS operations, click the "Console" link. You can run operating system commands using the following format (replace "dir" with any OS command):

```
    call {dir}
```

You can also use the console to run any REBOL functions/scripts (for any sort of batch file operations, text searches, to download file/directories from other FTP sites, etc.). This adds enormous power to the system:

```
    rename %oldfile.txt %newfile.txt
    delete %unwanted_file.txt
    foreach file (read ftp://u:p@site.com/) [
        write file read (join http://site.com/ file)
    ]
```

(You can perform almost any non-interactive operation possible

```
    in the REBOL console)

During use, backups are automatically created of any file which is
edited using the built-in editor (saved in the %./edit_history
subfolder), so you can always easily fix mistakes or revert to
previous versions of a page or site map.  It's all extremely SIMPLE
and QUICK to implement and use.  New users can learn the system in a
matter of minutes (the syntax pattern for editing the site map is the
only thing that requires any thought whatsoever, and that's only
necessary if you want to make _changes_ to the site layout).


INSTALLATION:

To install, just copy this script and an appropriate REBOL interpreter
to your web server, (version 2.76+ is required for console
operations), set permissions and the shebang line of this script, then
start adding/editing pages to your site.


TEMPLATE FILES (for advanced users):

Two generic page templates are built into this script, but ANY HTML
template can be added and used on your site.  Template files are
simply HTML files that act as a "frame" for new content that you
create with this script.  They can be edited to radically change the
look, feel, and design of destination .html files generated by this
script.

Templates are extremely simple to create.  They can be created/edited
directly online using the built-in editor, or uploaded and edited
later using this script.  IMPORTANT:  Code files such as templates
should be edited using the plain text editor (with no visual WYSIWYG),
available by clicking the "Files" link, next to the "Upload" button on
the main page of this script.

NOTE:  The built-in templates insert a header image at the top of
every page (%header.jpg by default).  If you want to use the built-in
templates, you can simply upload a header image to appear at the top
of every page in your site.  Just create your own image, save it as
"header.jpg" and use the built-in upload facility to upload it to your
web site.  That's all you need to do to create a minimally unique
design for different sites.  If you do this, try to keep the
header.jpg image download size small.  You can reduce the .jpg quality
and number of colors in your image editing software.  The shape of a
header image should be like a banner - avoid letting it get too wide
or too tall, or it will take up too much screen real estate on your
site (500x100 pixels is a good ball park size for the built in
templates).

Templates contain 4 short lines of code that indicate where the source
file text/code should be placed on your destination pages, and where
the link menu should be placed on pages with sub-page link blocks.
You can use existing HTML pages to create templates or create
completely new designs for every web site.  To make them work with
this script, simply insert the codes below where you want the content
to appear on generated destination pages:

sitebuilder_title
; Page title in head tag  * By default same as the source file name *
sitebuilder_links
; Link menu(s) generated by this script (as defined in your site map)
sitebuilder_path
; Links through the hierarchy of sub-pages, back to your home page
```

```
    sitebuilder_content
    ; All of the data contained in the source file of each content page

    There are two main types of templates:  those with menu bars, and
    those without.  The built-in template %menu.tpl displays a menu of
    links on the left side of the page (each with a text rollover effect).
    The %menu.tpl file is used for any source pages that have ONE OR MORE
    sub-page(s) in the link block.  The built-in %nomenu.tpl template is
    used for pages with EMPTY link blocks.  You can edit the built-in
    template files, or create new HTML templates from scratch.  It
    literally takes just a few seconds to create template files from
    existing HTML pages.  Examine the built in templates to see how it
    works - it's very straightforward.  Simply name your templates
    menu.tpl and nomenu.tpl, then upload them to the folder on your server
    which contains this script.
    }
    print {<pre>}
    quit
]
quit


REBOL [title: "Rebface (SDK) GUI Web Cam Example"]
do %gfx-colors.r
do %gfx-funcs.r
do %view-funcs.r
do %view-vid.r
do %view-edit.r
do %view-feel.r
do %view-images.r
do %view-styles.r
do %view-request.r
do %view.r
;WM_CAP_START: 0x400
;WM_START: to-integer #{00000400}
WM_CAP_START: 1024
WM_CAP_UNICODE_START: WM_CAP_START + 100
WM_CAP_PAL_SAVEA: WM_CAP_START + 81
WM_CAP_PAL_SAVEW: WM_CAP_UNICODE_START + 81
WM_CAP_UNICODE_END: WM_CAP_PAL_SAVEW
WM_CAP_ABORT: WM_CAP_START + 69
WM_CAP_DLG_VIDEOCOMPRESSION: WM_CAP_START + 46
WM_CAP_DLG_VIDEODISPLAY: WM_CAP_START + 43
WM_CAP_DLG_VIDEOFORMAT: WM_CAP_START + 41
WM_CAP_DLG_VIDEOSOURCE: WM_CAP_START + 42
WM_CAP_DRIVER_CONNECT: WM_CAP_START + 10
WM_CAP_DRIVER_DISCONNECT: WM_CAP_START + 11
WM_CAP_DRIVER_GET_CAPS: WM_CAP_START + 14
WM_CAP_DRIVER_GET_NAMEA: WM_CAP_START + 12
WM_CAP_DRIVER_GET_NAMEW: WM_CAP_UNICODE_START + 12
WM_CAP_DRIVER_GET_VERSIONA: WM_CAP_START + 13
WM_CAP_DRIVER_GET_VERSIONW: WM_CAP_UNICODE_START + 13
WM_CAP_EDIT_COPY: WM_CAP_START + 30
WM_CAP_END: WM_CAP_UNICODE_END
WM_CAP_FILE_ALLOCATE: WM_CAP_START + 22
WM_CAP_FILE_GET_CAPTURE_FILEA: WM_CAP_START + 21
WM_CAP_FILE_GET_CAPTURE_FILEW: WM_CAP_UNICODE_START + 21
WM_CAP_FILE_SAVEASA: WM_CAP_START + 23
WM_CAP_FILE_SAVEASW: WM_CAP_UNICODE_START + 23
WM_CAP_FILE_SAVEDIBA: WM_CAP_START + 25
WM_CAP_FILE_SAVEDIBW: WM_CAP_UNICODE_START + 25
WM_CAP_FILE_SET_CAPTURE_FILEA: WM_CAP_START + 20
WM_CAP_FILE_SET_CAPTURE_FILEW: WM_CAP_UNICODE_START + 20
WM_CAP_FILE_SET_INFOCHUNK: WM_CAP_START + 24
```

```
WM_CAP_GET_AUDIOFORMAT: WM_CAP_START + 36
WM_CAP_GET_CAPSTREAMPTR: WM_CAP_START + 1
WM_CAP_GET_MCI_DEVICEA: WM_CAP_START + 67
WM_CAP_GET_MCI_DEVICEW: WM_CAP_UNICODE_START + 67
WM_CAP_GET_SEQUENCE_SETUP: WM_CAP_START + 65
WM_CAP_GET_STATUS: WM_CAP_START + 54
WM_CAP_GET_USER_DATA: WM_CAP_START + 8
WM_CAP_GET_VIDEOFORMAT: WM_CAP_START + 44
WM_CAP_GRAB_FRAME: WM_CAP_START + 60
WM_CAP_GRAB_FRAME_NOSTOP: WM_CAP_START + 61
WM_CAP_PAL_AUTOCREATE: WM_CAP_START + 83
WM_CAP_PAL_MANUALCREATE: WM_CAP_START + 84
WM_CAP_PAL_OPENA: WM_CAP_START + 80
WM_CAP_PAL_OPENW: WM_CAP_UNICODE_START + 80
WM_CAP_PAL_PASTE: WM_CAP_START + 82
WM_CAP_SEQUENCE: WM_CAP_START + 62
WM_CAP_SEQUENCE_NOFILE: WM_CAP_START + 63
WM_CAP_SET_AUDIOFORMAT: WM_CAP_START + 35
WM_CAP_SET_CALLBACK_CAPCONTROL: WM_CAP_START + 85
WM_CAP_SET_CALLBACK_ERRORA: WM_CAP_START + 2
WM_CAP_SET_CALLBACK_ERRORW: WM_CAP_UNICODE_START + 2
WM_CAP_SET_CALLBACK_FRAME: WM_CAP_START + 5
WM_CAP_SET_CALLBACK_STATUSA: WM_CAP_START + 3
WM_CAP_SET_CALLBACK_STATUSW: WM_CAP_UNICODE_START + 3
WM_CAP_SET_CALLBACK_VIDEOSTREAM: WM_CAP_START + 6
WM_CAP_SET_CALLBACK_WAVESTREAM: WM_CAP_START + 7
WM_CAP_SET_CALLBACK_YIELD: WM_CAP_START + 4
WM_CAP_SET_MCI_DEVICEA: WM_CAP_START + 66
WM_CAP_SET_MCI_DEVICEW: WM_CAP_UNICODE_START + 66
WM_CAP_SET_OVERLAY: WM_CAP_START + 51
WM_CAP_SET_PREVIEW: WM_CAP_START + 50
WM_CAP_SET_PREVIEWRATE: WM_CAP_START + 52
WM_CAP_SET_SCALE: WM_CAP_START + 53
WM_CAP_SET_SCROLL: WM_CAP_START + 55
WM_CAP_SET_SEQUENCE_SETUP: WM_CAP_START + 64
WM_CAP_SET_USER_DATA: WM_CAP_START + 9
WM_CAP_SET_VIDEOFORMAT: WM_CAP_START + 45
WM_CAP_SINGLE_FRAME: WM_CAP_START + 72
WM_CAP_SINGLE_FRAME_CLOSE: WM_CAP_START + 71
WM_CAP_SINGLE_FRAME_OPEN: WM_CAP_START + 70
WM_CAP_STOP: WM_CAP_START + 68
avicap32.dll: load/library %avicap32.dll
user32.dll: load/library %user32.dll
get-focus: make routine! [return: [int]] user32.dll "GetFocus"
hwnd-hide-console: get-focus
hide-window: make routine! [
    hwnd [int]
    a [int]
    return: [int]
] user32.dll "ShowWindow"
hide-window hwnd-hide-console 0
view/new center-face layout/tight [
    image 320x240
    across
    btn "Take Snapshot" [
        sendmessage cap-result WM_CAP_GRAB_FRAME_NOSTOP 0 0
        sendmessage-file cap-result WM_CAP_FILE_SAVEDIBA 0 "scrshot.bmp"
    ]
    btn "Exit" [
        sendmessage cap-result WM_CAP_END 0 0
        sendmessage cap-result WM_CAP_DRIVER_DISCONNECT 0 0
        free user32.dll
        quit
    ]
```

```
]
set-caption: make routine! [
    hwnd [int]
    a [string!]
    return: [int]
] user32.dll "SetWindowTextA"
hwnd-set-title: get-focus
set-caption hwnd-set-title "Web Camera"
find-window-by-class: make routine! [
    ClassName   [string!]
    WindowName  [integer!]
    return:     [integer!]
] user32.dll "FindWindowA"
hwnd: find-window-by-class "REBOLWind" 0
cap: make routine! [
    cap [string!]
    child-val1 [integer!]
    val2 [integer!]
    val3 [integer!]
    width [integer!]
    height [integer!]
    handle [integer!]
    val4 [integer!]
    return: [integer!]
] avicap32.dll "capCreateCaptureWindowA"
sendmessage: make routine! [
    hWnd [integer!]
    val1 [integer!]
    val2 [integer!]
    val3 [integer!]
    return: [integer!]
] user32.dll "SendMessageA"
sendmessage-file: make routine! [
    hWnd [integer!]
    val1 [integer!]
    val2 [integer!]
    val3 [string!]
    return: [integer!]
] user32.dll  "SendMessageA"
cap-result: cap "cap" 1342177280 0 0 320 240 hwnd 0
; 1342177280 in the line above is BitOR(WS_CHILD,WS_VISIBLE)
sendmessage cap-result WM_CAP_DRIVER_CONNECT 0 0
sendmessage cap-result WM_CAP_SET_SCALE 1 0
sendmessage cap-result WM_CAP_SET_OVERLAY 1 0
sendmessage cap-result WM_CAP_SET_PREVIEW 1 0
sendmessage cap-result WM_CAP_SET_PREVIEWRATE 1 0
do-events
```

FUNDAMENTALS:

REBOL has hundreds of built-in function words that perform common tasks. As in other languages, function words are typically followed by passed parameters. Unlike other languages, passed parameters are placed immediately after the function word and are not necessarily enclosed in parenthesis. To accomplish a desired goal, functions are arranged in succession, one after another. The value(s) returned by one function are often used as the argument(s) input to another function. Line terminators are not required at any point, and all expressions are evaluated in left to right order, then vertically down through the code. Empty white space (spaces, tabs, newlines, etc.) can be inserted as desired to make code more readable. Text after a semicolon and before a new line is treated as a comment. You can complete significant work by simply knowing the predefined functions in the language, and organizing them into a useful order.

REBOL contains a rich set of conditional and looping structures, which can be used to manage program flow and data processing activities. If, switch, while, for, foreach, and other typical structures are supported.

Because many common types of data values are automatically recognized and handled natively by REBOL, calculating, looping, and making conditional decisions based upon data content is straightforward and natural to perform, without any external modules or toolkits. Numbers, text strings, money values, times, tuples, URLs, binary representations of images, sounds, etc. are all automatically handled. REBOL can increment, compare, and perform proper computations on most common types of data (i.e., the interpreter automatically knows that 5:32am + 00:35:15 = 6:07:15am, and it can automatically apply visual effects to raw binary image data, etc.). Network resources and Internet protocols (http documents, ftp directories, email accounts, dns services, etc.) can also be accessed natively, just as easily as local files. Data of any type can be written to and read from virtually any connected device or resource (i.e., "write %file.txt data" works just as easily as "write ftp://user:pass@website.com data", using the same common syntax). The percent symbol ("%") and the syntax "%(/drive)/path/path/.../file.ext" are used cross-platform to refer to local file values on any operating system.

Any data or code can be assigned a word label. The colon character (":") is used to assign word labels to constants, variable values, evaluated expressions, functions, and data/action blocks of any type. Once assigned, variable words can be used to represent all of the data and/or actions contained in the given expression, block, etc. Just put a colon at the end of a word, and thereafter it represents all the following actions and/or data. That forms a significant part of the REBOL language structure, and is the basis for it's flexible natural language dialecting abilities.

Multiple pieces of data are stored in "blocks", which are delineated by starting and ending brackets ("[]"). Blocks can contain data of any type: groups of text strings, arrays of binary data, collections of actions (functions), other enclosed blocks, etc. Data items contained in blocks are separated by white space. Blocks can be automatically treated as lists of data, called "series", and manipulated using built-in functions that enable searching, sorting, ordering, and otherwise organizing the blocked data. Data and function words contained in blocks can be evaluated (their actions performed and their data values assigned) using the "do" word. New function words can also be defined using the "does" and "func" words. "Does" forces a block to be evaluated every time its word label is encountered. The "func" word creates an executable block in the same way as "does", but additionally allows you to pass your own specified parameters to the newly defined function word. You can "do" a module of code contained in a text file, as long as it contains the minimum header "rebol[]". Blocks are also used to delineate most of the syntactic

structures in REBOL (i.e., in conditional evaluations, function definitions, etc.).

The syntax "view layout [block]" is used to create basic GUI layouts. You can add graphic widgets to the layout simply by adding widget identifier words to the enclosed block: "button", "field", "text-list", etc. Color, position, spacing, and other facet words can be added after each widget identifier. Action blocks added immediately after any widget will perform the enclosed functions whenever the widget is activated (i.e., when the widget is clicked with a mouse, when the enter key pressed, etc.). Path refinements can be used to refer to items in the GUI layout (i.e., "face/offset" refers to the position of the selected widget face). Those simple guidelines can be used to create useful GUIs for data input and output, in a way that's native (doesn't require any external toolkits) and much easier than any other language.

See http://re-bol.com for a complete REBOL tutorial with line-by-line explanations of all the code in these examples.

```
REBOL [title: "Quick Manual"]
print "This will take a minute..."  wait 2
echo %words.txt what echo off   ; "echo" saves console activity to a file
echo %help.txt
foreach line read/lines %words.txt [
    word: first to-block line
    print "_____________________________________________________________^/"
    print rejoin ["word:  " uppercase to-string word]  print ""
    do compose [help (to-word word)]
]
echo off
x: read %help.txt
write %help.txt "VID STYLES (GUI WIDGETS):^/^/"
foreach i extract svv/vid-styles 2 [write/append %help.txt join i newline]
write/append %help.txt "^/^/LAYOUT WORDS:^/^/"
foreach i svv/vid-words [write/append %help.txt join i newline]
b: copy []
foreach i svv/facet-words [
    if (not function? :i) [append b join to-string i "^/"]
]
write/append %help.txt rejoin [
    "^/^/STYLE FACETS (ATTRIBUTES):^/^/" b "^/^/SPECIAL STYLE FACETS:^/^/"
]
y: copy ""
foreach i (extract svv/vid-styles 2) [
    z: select svv/vid-styles i
    ; additional facets are held in a "words" block:
    if z/words [
        append y join i ": "
        foreach q z/words [if not (function? :q) [append y join q " "]]
        append y newline
    ]
]
write/append %help.txt rejoin [
    y "^/^/CORE FUNCTIONS:^/^/" at x 4
]
editor %help.txt
```