Core Library Functions

The core library contains functions which are core to almost every app in the books. They are used to create the main app window, show and hide any splash screen, and show the win/lose screen and final score at the end of a game.

	InitialiseScreen(x#,y#, w, h, t$, col, r)	Creates the main app window. Size: *w* by *h at (x#,y#)*. Title: *t$*. Background colour: *col*, Orientations allowed: *r*
	ShowSplashScreen(f$)	Fills screen with image f$.
	HideSplashScreen(s)	Hides splash screen image after *s* secs or mouse press.
	HandleEndGame(wf$,w,sf$,s$,x#,y#)	When *wf$* <> "" shows whole-screen frame *w* of 2-frame *wf$* image. When *sf$* <> "" shows whole-screen image *sf$* Shows score *s$* centred on (*x#,y#*).

SpriteLine Library Functions

The sprite line library contains function to draw lines and basic outlines (rectangle, circle and polygon) using sprites.

Line

int	DrawSpriteLine(x1#,y1#,x2#,y2#,th#,col,op)	Creates a line between (x1#,y1#) and (x2#,y2#). Thick: th#. Colour: col. Opacity: op. Returns ID of line.
	RedrawSpriteLine(id,x1#,y1#,x2#,y2#,th#,col,op)	Redraws existing line, *id*, with new values.
	DeleteSpriteLine(id)	Deletes line *id*.

Box

int	DrawSpriteBox(x1#,y1#,x2#,y2#,th#,col,op)	Creates box outline. Top-Left:(x1#,y1#). Bottom-right: (x2#,y2#). Thick: th#. Colour: col. Opacity: op. Returns ID of box.
	RedrawSpriteBox(id,x1#,y1#,x2#,y2#,th#,col,op)	Redraws existing box, *id*, with new values.
	DeleteSpriteBox(id)	Deletes box *id*.

Circle

int	DrawSpriteCircle(x#,y#,rad#,th#,col,op)	Creates circle outline. Centre:(*x#*,*y#*). Radius:*rad#*. Thick: *th#*. Colour: *col*. Opacity:*op*. Returns ID of circle.
	RedrawSpriteCircle(id,x1#,y1#,x2#,y2#,th#,col,op)	Redraws existing circle, *id*, with new values.
	DeleteSpriteCircle(id)	Deletes circle *id*.

Polygon

int	DrawSpritePolygon(pnts#[],th#,col,op)	Creates polygon outline. Coords:pnts#[] (x,y,x,y, etc.). Thickness: th#. Colour: col. Opacity: op. Returns ID of polygon.
	RedrawSpriteBox(id,pnts#[],th#,col,op)	Redraws existing polygon, *id*, with new values.
	DeleteSpriteBox(id, num)	Deletes polygon *id* containing *num* edges.

Bezier Curve

int	CreateBCurve((sx#,sy#,ex#,ey#,cx#,cy#)	Creates a Bezier curve and returns its ID. Start (sx#,sy#) End (ex#,ey#). Control (cx#,cy#) Defaults black, 0.25 thick, 20 segments.
	SetBCurveControl(id,cx#,cy#)	Moves control point of B curve id to (cx#,cy#).
	SetBCurveStart(id,sx#,sy#)	Moves start point of B curve id to (sx#,sy#).
	SetBCurveEnd(id,ex#,ey#)	Moves end point of B curve id to (ex#,ey#)
	SetBCurveColour(id,col)	Sets colour of B curve id to col.
	SetBCurveThickness(id, th)	Sets thickness of B curve id’s lines to th.
	SetBCurveSegments(id, num)	Sets number of lines used to draw B curve id to num
	DrawBCurve(id)	Draws B curve id

Top Scores Library Functions

The top scores library handles the creation, loading, saving, updating and the displaying of a game’s top scores table (which holds the name and score for each top score). The table can show standard points-based scores or time-based scores (m:ss or h:mm:ss). The player’s name is listed along with their high score. Generally, numeric scores will be in descending order (highest score first) and time-based score will be in ascending order (shortest time first). The table itself is constructed from three separate graphics: top, bottom and middle, with the middle graph being repeated to allow for a varying number of rows in the table.

	AddTopScore(n$, v)	Adds *n$* and *v* as a new table entry at correct position.
	DeleteTopScoreTable()	Deletes the sprites, images and text used in table. Also saves data contents to file.
str	FormatScore(v, f)	Returns string with score v in format *f* (0,1,2).
str	GetTopScoreName()	Allows user to enter name and returns value entered.
	HandleTopScores(b$,v,f,o)	A top-level function which initialises and displays the top scores table, as well as deleting it when complete. The table is displayed over background image *b$*. The table’s scores are shown in format *f* in order *o*. If it is a high score, *v* is added to the table and a name input with this latest value being added before the table is displayed.
	InitialiseTopScoresTable((r,f,o,df$)	Initialise the top scores table to have *r* rows, using score format *f* (0:num 1:m:ss, 2: h:mm:ss) and in order *o* (-1: descending, 1: ascending). Data stored in file *df$*.
int	IsNewTopScore(v)	Returns 1 if *v* is a new high score; else zero.
	LoadTopScoresList()	Loads the top scores data from data file.
	SaveTopScoresList()	Saves the table data to data file.
	ShowTopScoresTable(x#,y#,w#,f$)	Table top-left at (*x,y*), width: *w*. Images used:*f$*+”top.png”, *f$*+”mid.png”,*f$*+”bottom.png”.

GUI Library Functions

The GUI library allows the creation of some basic GUI elements such as buttons, checkboxes, radio buttons, dialog boxes, popup menus and frames. It also has an option to create a number pad for numeric data entry.

GUIButton

A button displays a three-vertical-frame image sprite (or creates a simple default one) and overlayed text (may be blank). Frame one displays by default, frame two when the pointer is over the button, frame three when the mouse button is pressed. Only frames one and three will be seen on a touch device.

int	CreateGUIButton(x#,y#,w#,h#, g$, t$)	Creates button (dim *w#* x *h#*) at (*x#*,*y#*) ,img *g$*, txt *t$*. Returns id of button.
	DeleteGUIButton(id)	Deletes button *id.*
int	HandleGUIButton(id)	Returns 1 if *id* pressed. Makes button reacts to user.
int	SetGUIButtonDepth(id, ly)	Places button on depth *ly*. Returns 1 if okay.
int	SetGUIButtonPosition(id, x#, y#)	Sets button position to (*x#*,*y#*). Returns 1 if okay.
int	SetGUIButtonSize(id, w#, h#)	Sets button *id* to size to *w#* by *h#*. Returns 1 if okay.

GUIDialogBox

A dialog box consists of an image sprite and a single button by default (more can be added). Pressing a dialog box button causes the dialog box to be deleted and the pressed button’s number is returned (1,2,3, etc.)

int	CreateGUIDialogBox(x#, y#, w#, h#, g$, t$, bg$, bt$)	Creates a dialog box *w#* x *h#*, at (*x#*,*y#*),box framed by image *g$* and title *t$*. Button images *bg$* (\| separated) showing *bt$* (| separated). Returns id of dialog box.
int	HandleGUIDialogBox()	Returns no. of button (not id) pressed. Deletes dialog box.
int	SetGUIDialogBoxButtonPosition(n, x#, y#)	Repositions button *n* to (*x#*,*y#*). Returns 1 if okay.
int	SetGUIDialBoxButtonSize(n, w#,h#)	Resizes button n to *w#* by *h#*. Returns 1 if okay.

GUICheckbox

A checkbox consists of a two-vertical frame image sprite and associated text. Clicking on the image or text will cause the checkbox to flip to its alternate setting (checked/unchecked).

int	CreateGUICheckbox(x#, y#, g$, t$)	Positions checkbox at (*x#*,*y#*). Shows image *g$* and text *t$*. Returns id assigned.
	DeleteCheckbox(id)	Deletes checkbox *id*.
int	GetGUICheckboxState(id)	Returns checkbox *id*’s current frame (1/2).
int	HandleGUICheckbox(id)	Returns frame shown by checkbox *id* (1/2). Makes checkbox reacts to user clicks.
	SetGUICheckboxTextColor(id, col)	Changes checkbox *id*’s text colour to *col*.
	SetGUICheckboxPosition(id, x#, y#)	Places checkbox *id* at (*x#*,*y#*).
	SetGUICheckboxTextSize(id, sz#)	Changes checkbox *id*’s text size to *sz#*.

GUIRadioButton

A radio button consists of a two-vertical frame image sprite and associated text. Radio buttons are associated with a group number. Clicking on the image or text will cause an unselected radio button to become selected and all other radio buttons in that group to be unselected.

int	CreateGUIRadioButton(x#,y#,g$,t$,gp)	Positions radio button at (x#, y#). Shows image *g$* & text *t$*. Belongs to group *gp*. Returns id assigned.
	DeleteGUIRadioButtonGroup(gp)	Deletes all buttons in group *gp*.
int	GetGUIRadioButtonSelectedInGroup(gp)	Returns no. of selected button in group (1,2,3 etc.).
int	HandleGUIRadioButtonGroup(gp)	Selects/deselects when clicked. Returns current frame (1/2)
	SetGUIRadioButtonTextColor(id,col)	Sets text colour of button *id* to *col*.
int	SetGUIRadioButtonDepth(gp, ly)	Sets depth of all buttons in group *gp* to *ly*. Returns 1 if okay.
	SetGUIRadioButtonPosition(id, x#, y#)	Places button *id* at (*x#*,*y#*).
	SetGUIRadioButtonTextSize(id,sz#)	Sets text size of button *id* to *sz#*.

GUIFrame

A frame is an area to which other elements can be added. The frame may be filled with a background image and have a title. Elements positioned within a frame have positions relative to the top-left corner of the frame. Added elements are given an index number starting at 1. Moving a frame automatically moves the elements within the frame. Deleting a frame also deletes all the elements it contains.

int	CreateGUIFrame(x#,y#,w#,h#,g$)	Creates frame (dim *w#* x *h#*) at (*x#*,*y#*) filled with image *g$*. Returns frame id.
int	AddButtonToGUIFrame(frm,x#,y#,w#,h#,g$,t$)	Creates a button in frame *frm at (x#,y#)*. size:*(w#xh#);* image:*g$;* text:*t$*. Returns button’s frame index.
int	AddCheckboxToGUIFrame(frm,x#,y#,g$,t$)	Creates a checkbox in frame *frm at (x#,y#)*. image:*g$;* text:*t$*. Returns checkbox’s frame index.
int	AddEditboxToGUIFrame(frm, x#, y#, w#, h#)	Creates an edit box in frame *frm* at (*x#*,*y#*). Size: *w#* x *h#*. Returns edit box’s frame index.
int	AddRadioButtonToGUIFrame(frm,x#,y#,g$,t$,gp)	Creates a radio button in frame *frm at (x#,y#)*. image:*g$;* text:*t$*. In group *gp.* Returns radio button’s frame index.
int	AddSpriteToGUIFrame(frm,x#,y#,w#,h#,g$)	Creates a sprite in frame *frm*, dim: *w#* x *h#* ,at (*x#*,*y#*). Returns created sprite’s frame index.
int	AddTextToGUIFrame(frm,x#,y#,sz#,t$,col)	Creates a text in frame *frm at (x#,y#)*. size:*sz#;* colour:*col* text:*t$*. Returns text’s frame index.
int	GetGUIFrameElementDetails(frm,idx)	Returns details of element *idx* in *frm*. (element type *100000 + true id)
int	DeleteGUIFrame(frm)	Deletes frame *frm*. Returns 1 if okay.
int	HandleGUIFrame(frm)	Returns frame index of any frame element clicked by user.
int	SetGUIFrameDepth(frm,ly)	Places frame on depth ly. Returns 1 if okay.
int	SetGUIFramePosition(frm,x#,y#)	Positions frame *frm* at (*x#*,*y#*). Returns 1 if okay.

GUIPopUpMenu

The popup menu is created from a combination of a frame and column of buttons. Assumes only one popup menu can exist at any moment in time.

CreateGUIPopUpMenu(x#,y#,w#,h#,fg$,bg$,ops$)	Creates a popup menu (dim *w#* x *h#*) at (*x#,y#*) ; frame image:*fg$*, btn image: *bg$*. Menu options: *ops$* (| separated).
DeleteGUIPopUpMenu()	Deletes the menu.
int HandleGUIPopUpMenu()	Returns the number of the option selected (1,2,3,etc.).

GUIColorPicker

The ColorPicker widget allows the user to select any colour shown on the ColorPicker sprite. Only a single ColorPicker widget can exist at any one time. The last colour picked is saved and its value can be accessed.

CreateGUIColorPicker(x#,y#,w#,h#,fg$)	Creates a sprite showing image *fg$* at (*x#,y#*) with dimensions *w#* by *h#*.
DeleteGUIColorPicker()	Deletes existing ColorPicker.
int GetGUIColorPickerBlue()	Returns the blue element of the last selected colour.
int GetGUIColorPickerDepth()	Returns ColorPicker's current depth layer.
int GetGUIColorPickerExists()	Returns 1 if the ColorPicker exists, else returns zero.
int GetGUIColorPickerGreen()	Returns the green element of the last selected colour.
int GetGUIColorPickerRed()	Returns the red element of the last selected colour.
int HandleGUIColorPicker()	Returns the colour selected as a single integer. -1 if none.
SetGUIColorPickerDepth(d)	Draws the ColorPickers sprite on layer *d*.

GUINumberPad

The number pad is created using a frame with 12 buttons (0..9, backspace and enter). The background image used should contain a rectangular display area where the value entered can be displayed. There is an option to have the number pad delete automatically after a value has been entered. Assumes only one number pad can exist at any moment in time.

CreateGUINumberPad(x#,y#,w#,h#,fg$,bg$,del)	Creates a number pad (dim *w#* x *h#*) at (*x#,y#*) ; frame image:*fg$*, btn image: *bg$*. Delete after: *del* (1 = delete)
int HandleGUINumberpad()	Accepts key presses. Displays value entered. When Enter pressed, delete number pad or reset its display to zero. Returns value entered.
DeleteGUINumberPad()	Deletes the number pad.
MoveGUINumberPadText(x#,y#)	Moves pad's display to (*x#,y#*) within pad.
ResizeGUINumberPadText(sz#)	Resizes display text to *sz#*.

GUIStopwatch

The Stopwatch widget allows the elapsed time to be shown in minutes and seconds (internally recorded in milliseconds). The visuals are created by four images and a text label. Although in theory the programmer can create their own images this may prove difficult because of the hard-wired code for positioning and sizing the various elements when the watch is created. It is best to use the default images supplied. The user can control the starting and stopping of the stopwatch as well as resetting its time to zero by pressing the displayed watch buttons. This ability can be disabled.

int CreateGUIStopwatch(x#,y#,w#,h#,f$)	Positions stopwatch at (*x#,y#*); dimensions (w# by h#) Constructed from image file f$ and related named files. Returns ID assigned to watch
DeletegGUIStopwatch(id)	Deletes watch id and its sprites, images and label
int GetGUIStopwatchControls(id)	Returns 1 = buttons enabled, 0 = buttons disabled
int GetGUIStopwatchExists(id)	Returns 1 if watch with ID id exists, else 0 returned
int GetGUIStopwatchState(id)	Returns 1 if watch running; 0 if stopped
int GetGUIStopwatchTime(id)	Returns the time recorded in watch (msecs)
int HandleGUIStopwatch	Returns user button press(-1 stop, 1:start, 2:reset, 0: none)
ResetGUIStopwatch(id)	Sets time back to zero and stops the watch
SetGUIStopwatchControls(id,f)	f = 1 : enable controls f = 0: disable controls
SetGUIStopwatchDepth(id, d)	Sets watch face depth to d (buttons d+1; text d - 1)
SetGUIStopwatchPosition(id,x#,y#)	Positions top-left of watch face at (x#,y#)
SetGUIStopwatchSize(id,w#,h#)	Resizes watch face to(w# by h#) (best make one value -1)
StartGUIStopwatch(id)	Starts watch id running
StopGUIStopwatch(id)	Stops watch id (displayed time unchanged)
UpdateGUIStopwatch(id)	Updates display of watch and time stored (call each frame)

List Library Functions

A List is a data structure designed to contain integers which represent the IDs of memblocks. The format of the memblocks themselves needs to be defined within each new app as does the access to the fields within those memblocks.
The list operations are designed to manipulate the integer values within the core List data structure. Parameters marked by an asterisk (*ref* parameters) are modified by the function.

When using a List which references memblocks, start by defining *DataType* giving the fields that need to be stored in the memblock. This is your record structure.
Write a *RecordToMemblock()* function which takes a *DataType* parameter and stores its contents in a memblock and returns the ID of that memblock (created by the function). It is this ID that should be stored in the List structure. Write other functions as required. See AliceList example in book.

CreateList(*list, sz, fx)	Creates an empty list (*list*) containing *sz* elements. May be of a fixed size (*fx* = 1) or may expand as required (*fx* = 0).
AddToList(*list, v)	Adds *v* to end of *list*.
DeleteFromList(*list, p)	Deletes value at position *p* in *list*.
DeleteList(*list)	Deletes the contents of *list*.
int FindInList(list, v)	Returns the position of *v* in *list* (-1 if not found).
int GetFromList(list, p)	Returns the value at position *p* in *list* (-1 if invalid *p*).
InsertInList(*list, v, p)	Inserts *v* at position *p* in *list* (*p* starts at 1).
int IsEmptyList(list)	Returns 1 if list empty, else zero.
int IsFullList(list)	Returns 1 if list full, else zero.
int LengthOfList(list)	Returns the number of entries in *list*.
str ToStringList(list)	Returns a string contain every value in *list* (| separated).

Date Library Functions

These are a collection of functions which may be used when manipulating dates.

int CalcDayOfWeek(d,m,y)	Returns the day of the week *d/m/y* falls on (0=Sunday).
int DateToJDN(d,m,y)	Returns number of days between *d/m/y* and 1/1/4713BC.
str JDNToDate(jdn)	Returns date equivalent of *jdn* as string in format dd/mm/yyyy.
int DaysBetween(d1,m1,y1,d2,m2,y2)	Returns the number of days between *d1/m1/y1* and *d2/m2/y2*.
str AddDays(d,m,y,dys)	Returns a string giving date of *d/m/y* + *dys* days.

For

Hands On AppGameKit Studio

Volumes 1 and 2

User-Defined Library Functions

The functions listed here are created by various book activities and become a library of user-defined routines for use in other projects.

May 2020

www.digital-skills.co.uk