CoreLibrary 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.

	DeleteResources()	Deletes all media except sounds and virtual buttons.
	InitialiseScreen(w, h, t$, col, r)	Creates the main app window centred on screen. Size: *w* by *h*. Title: *t$*. 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#*).

SpriteLibrary Functions

This library is used to reduce the number of function calls required to set up a sprite by performing the image load (or sprite cloning), positioning, sizing and visibility setting in a single function call.

fl	CalculateSpriteAngle(x1#,y1#,x2#,y2#)	Returns sprite angle req'd moving from (x1#,y1#) to (x2#,y2#).
int	CopySprite(x#,y#,w#,h#,v,spr)	Creates cloned sprite from sprite *spr*. Top-left positioned at (*x#y#*), size *w#* by *h#*. Visibility: *v*. Returns ID assigned.
int	CopySpriteByOffset(x#,y#,w#,h#,v,spr)	Creates cloned sprite from sprite *spr*. Offset positioned at (*x#y#*), size *w#* by *h#*. Visibility: *v*. Returns ID assigned.
int	InitialiseSprite(x#,y#,w#,h#,v,img)	Creates sprite using image with ID *img*. Top-left positioned at (*x#y#*), size *w#* by *h#*. Visibility: *v*. Returns ID assigned.
int	InitialiseSpriteByOffset(x#,y#,w#,h#,v,img)	Creates sprite using image with ID *img*. Offset positioned at (*x#y#*), size *w#* by *h#*. Visibility: *v*. Returns ID assigned.
	MoveSpriteWithWrap(id,x#,y#)	Move sprite id x# units in x direction, y# in y direction. Wraps around screen edges if required.

SpriteLineLibrary 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

TopScoresLibrary 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. Image names should end "top.png", "mid.png" and "bottom.png".

	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.
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. Also saves updated table to data file (calls *SaveTopScoreList()*).
	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".

ListLibrary 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).

DateLibrary 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.

StringLibrary Functions

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

str	FillString(ch$,n)	Returns string containing *n* copies of *ch$*
str	RandomString(sz)	Returns a *sz* length string of random uppercase letters If *sz* = -1, string length is random (1 to 50)
int	Pos(s$,ch$)	Returns position of first occurrence of *ch$* in *s$*
int	Occurs(s$,ch$)	Returns how often character *ch$* occurs in *s$*
str	Insert(s1$,s2$,p)	Returns a string created by inserting *s2$* into *s1$* at position *p*
str	Delete(s$,p,num)	Returns *s$* with *num* characters deleted starting at position *p*
str	Replace(s$,ch$,p)	Returns a string created by replacing char at *p* in *s$* with the char *ch$*

for

Hands On AppGameKit Studio

Volume 1

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.

October 2021

www.digital-skills.co.uk