GUI: A Stopwatch

In this Supplement:

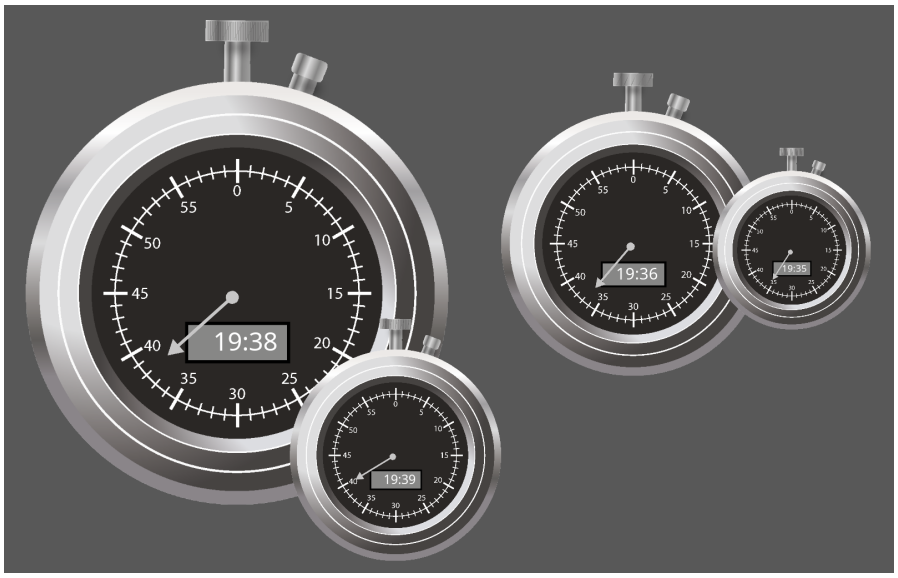
- ❑ Designing a Stopwatch Widget
- ❑ Coding a Stopwatch Widget
- ❑ Testing a Stopwatch Widget
- ❑ Omissions and Mistakes

Creating a Stopwatch Widget

Introduction

In Chapter 4 of **Hands On AppGameKit Studio Volume 2** we cover the creation of graphical user interface controls (or widgets). It covers the coding of various widgets such as Buttons, Checkboxes, Radio buttons, Frames and Popup menus.

This supplement adds to that collection by creating a **Stopwatch** widget, examples of which can be seen in the screen shot below.



The stopwatch displays a rotating “seconds” hand and a digital time showing the elapsed time in minutes and seconds.

The larger of the two buttons is used to start and stop the timer while the smaller button resets the timer to zero.

Identifying the Required Operations

There are several operations we’ll want to have available when using a stopwatch. These are briefly described below:

CreateGUIStopwatch()
: Creates a stopwatch. Initially the stopwatch is set to zero and stopped. This also specifies the position, size and images used in the visualisation of the watch.

StartGUIStopwatch()
: Sets the watch to a *running* state.

StopGUIStopwatch()
: Sets the watch to a *stopped* state.

GetGUIStopwatchState()
: Returns the state of the watch (*running* or *stopped*).

UpdateGUIStopwatch()
When the watch is in the *started* state, this operation updates the time displayed on the watch. If the watch is in the *stopped* state, this operation has no effect.

ResetGUIStopwatch()
Resets the time to zero. Sets the watch to the *stopped* state.

HandleGUIStopwatch()
Returns an indicator of how the user has interacted with the stopwatch. The following return values are possible:

```
-1   : the user has stopped the watch
 0   : there is no valid user interaction
+1   : the user has started the watch
+2   : the user has reset the watch
```

The user interacts with the stopwatch by clicking on the two buttons at the top of the watch.

GetGUIStopwatchTime()
Returns the elapsed time as recorded on the stopwatch (in milliseconds).

DeleteGUIStopwatch()
Deletes the stopwatch and its components.

SetGUIStopwatchControls()
Enables/disables the use of the *stop/start* and *reset* buttons by the user without stopping the watch.

The last option given above can be used when a stopwatch is being used to record a player's time taken to perform a task. After all, we don't want them stopping or resetting the timer!

Other operations are likely to be required because of the sprite-like nature of most GUI widgets. The ones included here are:

SetGUIStopwatchPosition()
Moves the watch to a specified position.

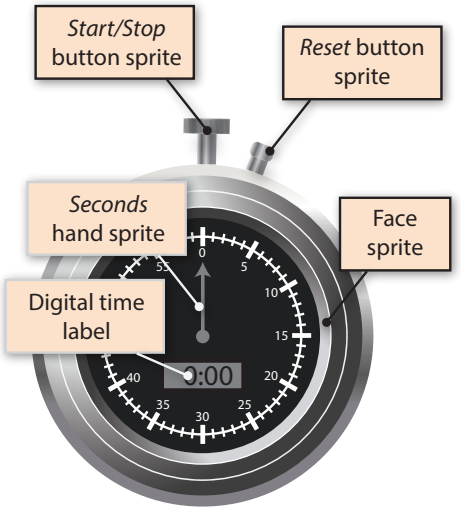
SetGUIStopwatchSize()
Sets the dimensions of the watch.

SetGUIStopwatchDepth()
Sets the layer for the watch face and the associated visual elements.

Stopwatch Visual Elements

Unlike most of the other widgets we have created, the stopwatch widget is constructed from several elements and creating your own graphics will be a more complex task.

Exactly what components make up a stopwatch are shown in the diagram below.



Implementing the Stopwatch

Data

Our first consideration in implementing our *Stopwatch* widget is to consider what information we'll need to store about the widget. In fact, most of what is required are ID's to images, sprites and the label. We'll also need to record the current state of the stopwatch (stopped or running), the elapsed time and whether the controls (the two buttons at the top of the watch) have been enabled.

Assuming those using this widget may want to create more than one stopwatch in an app, we will make use of an array element to store details of each stopwatch. As we highlighted with previous widgets, when using an array, the simplest way of deleting a watch is by making use of a *deleted* indicator rather than erasing the array entry.

So our code for the data associated with a single stopwatch becomes:

```
type StopwatchType
   faceimg        as integer  //ID of face image
   startstopimg   as integer  //ID of start/stop button image
   resetimg       as integer  //ID of reset button image
   handimg        as integer  //ID of watch hand image
   facespr        as integer  //ID of face sprite
   ssspr          as integer  //ID of start/stop button sprite
   resetspr       as integer  //ID of reset button sprite
   handspr        as integer  //ID of watch hand sprite
   timetxt        as integer  //ID of text showing elapsed time
   time           as integer  //Elapsed time since started
   state          as integer  //0: stopped, 1: running
   enabled        as integer  //Buttons enabled (1:yes, 0:no)
   deleted        as integer  //0: exists, 1:deleted
endtype
```

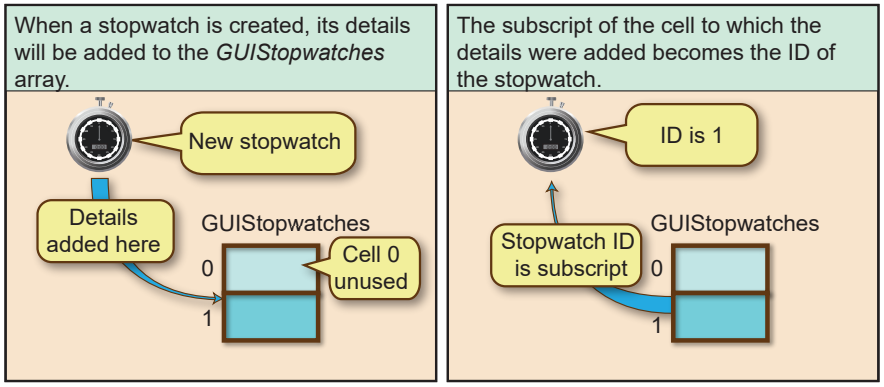
And, allowing for more than one stopwatch, we'll also need an array of our *StopwatchType* structure where we can store details of every watch:

```
global GUIStopwatches as StopwatchType[0]
```

Notice that the array has been created with only a single element (cell zero – which

we won't be using). We'll grow the array each time a stopwatch is added. And although this is not an efficient way to increase the size of an array, it has the advantage of keeping things simple.

Like our previous widgets, each stopwatch will be assigned an ID. In this case, the ID will be the subscript of the cell in which a stopwatch's details are stored. The figure below demonstrates the idea.



Functions

CreateGUIStopwatch()

Mini-spec

FUNCTION NAME	:	CreateGUIStopwatch
PARAMETERS		
In	:	x : real y : real w : real h : real image : string
Out	:	id : integer
PRE-CONDITION	:	*image* file and associated image files exist
DESCRIPTION	:	Creates a stopwatch widget using *image* as the basis of the face of the stopwatch. Other images used (for the buttons and seconds hand) are assumed to have names based on *image*. *start/stop* button : adds "ss" to *image* name *reset* button : adds "reset" to *image* name *seconds* hand: adds "hand" to *image* name The seconds hand points zero. The time in digital form reads 0:00 The two buttons are enabled. The watch is stopped. The face sprite is placed at (*x*.*y*) with dimensions *w* by *h*. Other sprites and the text are sized and positioned appropriately. The face is on layer 10, the buttons on 11, hand on 9 and the text on layer 10.

Code:

```
//*** Creates stopwatch. Uses images facefile+facefileshad+
↳facefiless+facefilereset+facefilehand ***
//*** The time elapsed is set to zero and the watch is stopped,
↳buttons are enabled              ***
function CreateGUIStopwatch(x as float, y as float, w as float,
↳h as float, facefile as string)
   //*** Add cell to stopwatch list ***
   GUIStopwatches.length = GUIStopwatches.length+1
   //*** Last position in array is new button's ID ***
   id = GUIStopwatches.length
   //*** Load images ***
   GUIStopwatches[id].faceimg = LoadImage(facefile)
   GUIStopwatches[id].startstopimg = LoadImage(Left(facefile,
   ↳Len(facefile)-4)+"ss"+Right(facefile,4))
   GUIStopwatches[id].resetimg = LoadImage(Left(facefile,
   ↳Len(facefile)-4)+"reset"+Right(facefile,4))
   GUIStopwatches[id].handimg = LoadImage(Left(facefile,
   ↳Len(facefile)-4)+"hand"+Right(facefile,4))
   //*** Create, size and position face sprite ***
   GUIStopwatches[id].facespr = CreateSprite(GUIStopwatches[id].
   ↳faceimg)
   faceid = GUIStopwatches[id].facespr
   SetSpriteSize(faceid,w,h)
   SetSpritePosition(faceid,x,y)
   //*** Create, size, position and layer start/stop button ***
   GUIStopwatches[id].ssspr = CreateSprite(GUIStopwatches[id].
   ↳startstopimg)
   ssid = GUIStopwatches[id].ssspr
   SetSpriteSize(ssid,w*0.15,-1)
   SetSpritePositionByOffset(ssid,x+GetSpriteWidth(faceid)/2.0,y-
   ↳GetSpriteHeight(ssid)/2.1)
   SetSpriteDepth(ssid,GetSpriteDepth(faceid)+1)
   //*** Create, size, position and layer reset button ***
   GUIStopwatches[id].resetspr = CreateSprite(GUIStopwatches[id].
   ↳resetimg)
   resetid = GUIStopwatches[id].resetspr
   SetSpriteSize(resetid,w*0.13,-1)
   SetSpritePositionByOffset(resetid,x+GetSpriteWidth(faceid)/1.5,y)
   SetSpriteDepth(resetid,GetSpriteDepth(faceid)+1)
   //*** Create, align, size, position and layer time label ***
   GUIStopwatches[id].timetxt = CreateText("0:00")
   txtid = GUIStopwatches[id].timetxt
   SetTextAlignment(txtid,2)
   SetTextSize(txtid,GetSpriteHeight(faceid)*0.08)
   SetTextPosition(txtid,x+GetSpriteWidth(faceid)/1.68, y+
   ↳GetSpriteHeight(faceid)/1.747)
   SetTextDepth(txtid,GetSpriteDepth(faceid))
   //*** Create, size, position and layer seconds hand ***
   GUIStopwatches[id].handspr = CreateSprite(GUIStopwatches[id].
   ↳handimg)
   handid = GUIStopwatches[id].handspr
   SetSpriteSize(handid,-1,GetSpriteHeight(faceid)*0.22)
   SetSpiteOffset(handid,GetSpriteWidth(handid)/2,
   ↳GetSpriteHeight(handid))
   SetSpritePositionByOffset(handid,GetSpriteX(faceid) +
   ↳GetSpriteWidth(faceid)/2.0,
   GetSpriteY(faceid)+GetSpriteHeight(faceid)/2.0)
   SetSpriteDepth(handid, GetSpriteDepth(faceid)-1)
```

```
   //*** Stopwatch not deleted ***
   GUIStopwatches[id].deleted = 0
   //*** Stopwatch buttons can be used ***
   GUIStopwatches[id].enabled = 1
   //*** Stopwatch stopped ***
   GUIStopwatches[id].state = 0
endfunction id
```

Note that the offset of the *seconds* hand sprite has been move to near the centre bottom of the sprite since this is the point about which we'll want the hand to rotate.

The image files required are:
swatch.png
swatchss.png
swatchrest.png
swatchhand.png

Activity 1

Start a new project called *StopwatchTest* and add the code given above for the Stopwatch widget's data and its *CreateGUIStopwatch()* function.

In *CreateInitialLayout()*, call *CreateGUIStopwatch()* and check that a stopwatch appears. Remember to copy the necessary image files to the project's *media* folder.

StartGUIStopwatch()

Mini-spec:

FUNCTION NAME	:	StartGUIStopwatch
PARAMETERS		
In	:	id : integer
Out	:	None
PRE-CONDITION	:	*id* is a valid stopwatch ID.
DESCRIPTION	:	Starts the watch running. If the watch has been stopped previously, it starts running at the time it was stopped. If the watch is already running, this operation has no effect.

Code:

```
function StartGUIStopwatch(id as integer)
   //*** If the id is invalid, exit ***
   if id < 1 or id > GUIStopwatches.length
      exitfunction
   endif
   //*** If stopwatch deleted, exit ***
   if GUIStopwatches[id].deleted = 1
      exitfunction
   endif
   //*** Set to running state ***
   GUIStopwatches[id].state = 1
endfunction
```

StopGUIStopwatch()

Mini-spec:

FUNCTION NAME	:	StopGUIStopwatch
PARAMETERS		
In	:	id : integer
Out	:	None
PRE-CONDITION	:	*id* is a valid stopwatch ID.
DESCRIPTION	:	Stops the watch running. If the watch is running , it stops. If the watch is already stopped, this operation has no effect.

Code:

```
function StopGUIStopwatch(id as integer)
   //*** If the id is invalid, exit ***
   if id < 1 or id > GUIStopwatches.length
      exitfunction
   endif
   //*** If stopwatch deleted, exit ***
   if GUIStopwatches[id].deleted = 1
      exitfunction
   endif
   //*** Set to stopped state ***
   GUIStopwatches[id].state = 0
endfunction
```

GetGUIStopwatchState()

Mini-spec:

FUNCTION NAME	:	GetGUIStopwatchState
PARAMETERS		
In	:	id : integer
Out	:	result : integer
PRE-CONDITION	:	*id* is a valid stopwatch ID.
DESCRIPTION	:	Sets *result* to 1 if the watch is running; 0 if it is stopped. Returns -1 if *id* is invalid.

Code:

```
function GetGUIStopwatchState(id as integer)
   //*** If the id is invalid, exit  -1***
   if id < 1 or id > GUIStopwatches.length
      exitfunction -1
   endif
   //*** If stopwatch deleted, exit -1 ***
   if GUIStopwatches[id].deleted = 1
      exitfunction -1
   endif
   //*** Set result to watch current state ***
   result = GUIStopwatches[id].state
endfunction result
```

UpdateGUIStopwatch()

StartGUIStopwatch() and *StopGUIStopwatch()* only have an effect on the state field of a stopwatch, but the visual effect of starting and stopping the watch and of time passing is performed by the *UpdateGUIStopwatch()* function.

Mini-spec:

FUNCTION NAME	:	StopGUIStopwatch
PARAMETERS		
In	:	id integer
Out	:	None
PRE-CONDITION	:	*id* is a valid stopwatch ID.
DESCRIPTION	:	If the watch is running, the elapsed time is saved and the *seconds* hand is moved to give an indication of the time elapsed. The seconds hand makes a full rotation of the face every 60 seconds. In addition the digital label shows the elapsed time in minutes and seconds.

Code:

```
function UpdateGUIStopwatch(id as integer)
   //*** If the id is invalid, exit ***
   if id < 1 or id > GUIStopwatches.length
      exitfunction
   endif
   //*** If stopwatch deleted, exit ***
   if GUIStopwatches[id].deleted = 1
      exitfunction
   endif//*** If watch stopped, exit ***
   if GUIStopwatches[id].state = 0
      exitfunction
   endif
   //*** Record the time passed ***
   GUIStopwatches[id].time = GetMilliseconds()
   msecs = GUIStopwatches[id].time
   //*** Set angle of hand to show seconds into current minute ***
   SetSpriteAngle(GUIStopwatches[id].handspr,msecs*6.0/1000.0)
   //*** Update the digital time on label ***
   SetTextString(GUIStopwatches[id].timetxt,
   ↳ToMinutesSeconds(msecs))
endfunction
```

Notice that the string for the digital time label is formatted using a helper function named *ToMinutesSeconds()*. This is coded as

```
//*** Creates formatted string in the form (mm)m:ss ***
function ToMinutesSeconds(msecs as integer)
   totalsecs = msecs /1000
   mins = totalsecs / 60
   secs = Mod(totalsecs,60)
   result as string
   result = Str(mins)+":"+Right("00"+Str(secs),2)
endfunction result
```

Activity 2

In *StopwatchTest*, add the code for functions *StartGUIStopwatch()*, *StopGUIStopwatch()*, *GetGUIStopwatchState()*, *UpdateGUIStopwatch()* and *ToMinutesSeconds()*.

In the main program, add a call to *StartGUIStopwatch()* (in *CreateInitialLayout()*) and *UpdateGUIStopwatch()* (in *HandleOther()*).

Does the stopwatch display the time correctly?

Activity 3

In *StopwatchTest*, to check that *StopGUIStopwatch()* operates correctly, make a call to that function after 5 seconds has passed (place the call in *HandleOther()*).

Does the watch stop after five seconds as expected?

Activity 4

In *StopwatchTest*, modify the code so that the stopwatch only begins running after five seconds have passed.

Does the watch behave as expected?

Activity 4 highlights a problem with our watch: it displays the time the app has been running rather than the time the stopwatch has been running.

In an attempt to overcome this problem, we might switch to using `Timer()` rather than `GetMilliseconds()` to get the elapsed time. Using `Timer()` we have the option of using `ResetTimer()` to reset the value returned by `Timer()` to zero.

This approach requires us to adjust the code in *StartGUIStopwatch()* to

```
//*** Starts stopwatch running ***
function StartGUIStopwatch(id as integer)
   //*** If the id is invalid, exit ***
   if id < 1 or id > GUIStopwatches.length
      exitfunction
   endif
   //*** If stopwatch deleted, exit ***
   if GUIStopwatches[id].deleted = 1
      exitfunction
   endif
   //*** Set to running state ***
   GUIStopwatches[id].state = 1
   //*** Set elapsed time to zero ***
   ResetTimer()
endfunction
```

Since `Timer()` returns seconds (and fractions of a second) rather than milliseconds, the minimum change required to *UpdateGUIStopwatch()* is that the line

```
GUIStopwatches[id].time = GetMilliseconds()
```

to

```
GUIStopwatches[id].time = Timer()*1000
```

Activity 5

In *StopwatchTest*, make the modifications to *StartGUIStopwatch()* and *UpdateGUIStopwatch()* as suggested above.

Does the watch now behave as expected?

Extend the `if` statement in *HandleOther()* by adding

```
and GetGUIStopwatchState(g.id) = 0
```

How does this affect the program?

For the moment we'll make the (mistaken) assumption that we've fixed the elapsed time problem and continue implementing the other operations of the stopwatch.

ResetGUIStopwatch()

Mini-spec:

FUNCTION NAME	:	ResetGUIStopwatch
PARAMETERS		
In	:	id : integer
Out	:	None
PRE-CONDITION	:	*id* is a valid stopwatch ID.
DESCRIPTION	:	The watch's time is reset to zero and the watch stopped. The seconds hand returns to the 0 position and the digital time is set to 0:)).

Code:

```
//*** Resets the time to zero and places ***
//*** the watch in stopped mode          ***
function ResetGUIStopwatch(id as integer)
   //*** If the id is invalid, exit ***
   if id < 1 or id > GUIStopwatches.length
      exitfunction
   endif
   //*** If stopwatch deleted, exit ***
   if GUIStopwatches[id].deleted = 1
      exitfunction
   endif
   //*** Stopwatch stopped ***
   GUIStopwatches[id].state = 0
   //*** No time has been recorded ***
   GUIStopwatches[id].time = 0
   //*** Reset second hand and digital time ***
   SetSpriteAngle(GUIStopwatches[id].handspr,0)
   SetTextString(GUIStopwatches[id].timetxt,"0:00")
endfunction
```

Activity 6

Add *ResetGUIStopwatch()* to *StopwatchTest* then modify the code so that *CreateInitialLayout()* starts the watch and *HandleOther()* resets it after five seconds.
Does the watch behave as expected?

HandleGUIStopwatch()

This is the function which allows the user to interact with the stopwatch (as long as the watch controls aren't disabled). As such it needs to manipulate the position of the buttons on the watch to give a visual indication that the watch has reacted to the user's input. Pressing the buttons will cause functions *StartGUIStopwatch()*, *StopGUIStopwatch()* or *ResetGUIStopwatch()* to be executed as appropriate. The routine returns a value indicating what action, if any, the user has taken.

Mini-spec:

FUNCTION NAME : HandleGUIStopwatch
PARAMETERS
In : id : integer
Out : result : integer
PRE-CONDITION : *id* is a valid stopwatch ID AND watch is enabled.
DESCRIPTION : If the user presses on the *start/stop* button, the button will be seen to depress and then return to its original position. This will cause the watch to switch from its current state (*running* or *stopped*) to the opposite state. The time displayed will not be affected if the watch is stopped. The time displayed will advance from its current setting if the watch is started.

If the user presses the *reset* button, the button will be seen to depress and then return to its original position. This will cause the watch to reset the *seconds* hand to the zero position and the digital time to 0:00. The watch will be set to the *stopped* state.

result will be set as follows:

-1	if the watch is stopped by pressing the *start/stop* button.
0	if *id* is invalid, the controls are disabled or the user has not interacted with the watch.
1	the watch was started by pressing the *start/stop* button.
2	the watch was reset by pressing the *reset* button.

Code:

```
//*** Handles user interaction with watch ***
//*** Allows start/stop and reset         ***
//*** Returns a value indicating action   ***
function HandleGUIStopwatch(id as integer)
   //*** If the id is invalid, exit zero ***
   if id < 1 or id > GUIStopwatches.length
      exitfunction 0
   endif
   //*** If stopwatch deleted, exit zero ***
   if GUIStopwatches[id].deleted = 1
      exitfunction 0
   endif
   result = 0
   //*** If watch disabled, exit zero ***
   if GUIStopwatches[id].enabled = 0
      exitfunction 0
   endif
   result = 0
   //*** If pointer pressed ***
   if GetPointerPressed()
      hit = GetSpriteHit(GetPointerX(), GetPointerY())
      //*** If start/stop button pressed ***
      if hit = GUIStopwatches[id].ssspr
         //*** Show start/stop button moving down and back ***
         SetSpriteY(hit, GetSpriteY(hit)+GetSpriteHeight(hit)/2.0)
         Sync()
         Sleep(100)
         SetSpriteY(hit, GetSpriteY(hit)-GetSpriteHeight(hit)/2.0)
         //*** If watch stopped, start it ***
         if GUIStopwatches[id].state = 0
            StartGUIStopwatch(id)
            GUIStopwatches[id].state = 1
            result = 1
         else //If watch running, stop it
            StopGUIStopwatch(id)
            result = -1
         endif
      //*** If reset pressed ***
      elseif hit = GUIStopwatches[id].resetspr
         //*** Show reset button moving down and back ***
         SetSpritePosition(hit,GetSpriteX(hit)-GetSpriteWidth(hit)/
         ↳5.0,GetSpriteY(hit)+GetSpriteHeight(hit)/4.0)
         Sync()
         Sleep(100)
         SetSpritePosition(hit,GetSpriteX(hit)+GetSpriteWidth(hit)/
         ↳5.0,GetSpriteY(hit)-GetSpriteHeight(hit)/4.0)
         ResetGUIStopwatch(id)
         result = 2
      endif
   endif
endfunction result
```

Activity 7

Add *HandleGUIStopwatch()* to *StopwatchTest* then call that function from *HandleUserInput()*. Remove the call to *ResetGUIStopwatch()* in *HandleOther()*.
Does the watch stop, start and reset as expected?

Activity 7 highlights the sort of problems that occur in real world programming. Books on programming generally present perfect programs that appear to have been developed flawlessly by the author on the first attempt. This can give those starting out in programming the idea that, because they are forever making mistakes, they aren't going to be too great at programming. But, of course, the truth is we all make mistakes all of the time when it comes to programming.

In the case of our stopwatch our error is one of omission and limited understanding of the consequences of the code we've created. We've forgotten to take into account the fact that, since *StartGUIStopwatch()* calls *ResetTimer()* and since the stopwatch uses the value returned by *Timer()* alone to determine the elapsed time, the time must restart at zero each time it enters *running* mode.

When a mistake is discovered, the further on we are in our coding the more complex the changes required will be to fix the problem. In this case, we need to rethink how we calculate the time displayed on the watch.

Let's say we stop the watch when it is reading 7 seconds and then restart it 10 seconds later. At this point we will want the watch to start running again from the 7 seconds mark. This means our calculation should be

time on watch = time on watch when last stopped + time elapsed since watch was restarted

In order to perform that calculation we are going to have to record the watch's time when it is stopped. And to do that, we're going to have to add a new field to *StopwatchType*:

```
type StopwatchType
   faceimg     as integer  //ID of face image
   startstopimg as integer    //ID for start/stop button image
   resetimg    as integer  //ID for reset button image
   handimg     as integer  //ID of watch hand image
   facespr     as integer  //ID of face sprite
   ssspr       as integer  //ID of start/stop button sprite
   resetspr    as integer  //ID of reset button sprite
   handspr     as integer  //ID of watch hand sprite
   timetxt     as integer  //ID of text showing elapsed time
   time        as integer  //Elapsed time since started (msecs)
   stoptime    as integer  //Time on watch when stopped
   state       as integer  //0: stopped, 1: running
   enabled     as integer  //Buttons enabled (1:yes, 0:no)
   deleted     as integer  //(0: exists, 1:deleted)
endtype
```

The value in this new field will have to be set when *StopGUIStopwatch()* is called:

```
//*** Stops the stopwatch running ***
function StopGUIStopwatch(id as integer)
   //*** If the id is invalid, exit ***
   if id < 1 or id > GUIStopwatches.length
      exitfunction
   endif
   //*** If stopwatch deleted, exit ***
   if GUIStopwatches[id].deleted = 1
      exitfunction
   endif
   //*** Set to stopped state ***
   GUIStopwatches[id].state = 0
   //*** Save current time on watch ***
   GUIStopwatches[id].stoptime = GUIStopwatches[id].time
endfunction
```

In *UpdateGUIStopwatch()* we'll need to use our new formula to calculate the time shown on the watch:

```
//*** Update the stopwatch visuals, moving   ***
//*** seconds hand and changing digital time ***
function UpdateGUIStopwatch(id as integer)
   //*** If the id is invalid, exit ***
   if id < 1 or id > GUIStopwatches.length
      exitfunction
   endif
   //*** If stopwatch deleted, exit ***
   if GUIStopwatches[id].deleted = 1
      exitfunction
   endif
   //*** If watch stopped, exit ***
   if GUIStopwatches[id].state = 0
      exitfunction
   endif
   //*** Record the time passed ***
   GUIStopwatches[id].time = GUIStopwatches[id].stoptime +
   ↳Timer()*1000
   msecs = GUIStopwatches[id].time
   //*** Modify angle of seconds hand to show seconds into
   ↳current minute ***
   SetSpriteAngle(GUIStopwatches[id].handspr,msecs*6.0/1000.0)
   //*** Update the digital time on label ***
   SetTextString(GUIStopwatches[id].timetxt,
   ↳ToMinutesSeconds(msecs))
endfunction
```

Finally, in *ResetGUIStopwatch()*, we'll need to reset the *stoptime* value to zero:

```
//*** Resets the time to zero and places ***
//*** the watch in stopped mode          ***
function ResetGUIStopwatch(id as integer)
   //*** If the id is invalid, exit ***
   if id < 1 or id > GUIStopwatches.length
      exitfunction
   endif
   //*** If stopwatch deleted, exit ***
   if GUIStopwatches[id].deleted = 1
      exitfunction
   endif
   //*** Stopwatch stopped ***
   GUIStopwatches[id].state = 0
   //*** No time has been recorded ***
   GUIStopwatches[id].time = 0
   GUIStopwatches[id].stoptime = 0
   //*** Reset second hand and digital time ***
   SetSpriteAngle(GUIStopwatches[id].handspr,0)
   SetTextString(GUIStopwatches[id].timetxt,"0:00")
endfunction
```

Activity 8

In *StopwatchTest* make the four changes shown above and retest the program.

Do all three operations – start, stop and reset – now perform correctly?

The next three functions are relatively minor

GetGUIStopwatchTime()

Mini-spec:

FUNCTION NAME	:	GetGUIStopwatchTime
PARAMETERS		
In	:	id : integer
Out	:	result : integer
PRE-CONDITION	:	*id* is a valid stopwatch ID.
DESCRIPTION	:	*result* is set to the time recorded internally in the stopwatch data structure. The time is given in milliseconds. If the *id* is invalid, -1 is returned.

Code:

```
//*** Returns the stopwatch's time ***
function GetGUIStopwatchTime(id as integer)
   //*** If id invalid, exit -1 ***
   if id < 1 or id > GUIStopwatches.length
      exitfundtion -1
   endif
   //*** If stopwatch deleted, exit -1 ***
   if GUIStopwatches[id].deleted = 1
      exitfunction -1
   endif
   //*** Return time ***
   result = GUIStopwatches[id].time
endfunction result
```

DeleteGUIStopwatch()

Mini-spec:

FUNCTION NAME	:	DeleteGUIStopwatch
PARAMETERS		
In	:	id : integer
Out	:	None
PRE-CONDITION	:	*id* is a valid stopwatch ID.
DESCRIPTION	:	Deletes the watch's sprites and images. Also marks the stopwatch with this ID as deleted without removing its data.

Code:

```
//*** Deletes stopwatch and resources ***
function DeleteGUIStopwatch(id as integer)
   //*** If id invalid, exit ***
   if id < 1 or id > GUIStopwatches.length
      exitfundtion
   endif
```

Code (continued):

```
   //*** If stopwatch deleted, exit ***
   if GUIStopwatches[id].deleted = 1
      exitfunction
   endif
   //*** Mark as deleted ***
   GUIStopwatches[id].deleted = 1
   //*** Delete sprites ***
   DeleteSprite(GUIStopwatches[id].facespr)
   DeleteSprite(GUIStopwatches[id].ssspr)
   DeleteSprite(GUIStopwatches[id].resetspr)
   DeleteSprite(GUIStopwatches[id].handspr)
   //*** Delete label ***
   SeleteText(GUIStopwatches[id].timetxt)
   //*** Delete images ***
   DeleteImage(GUIStopwatches[id].faceimg)
   DeleteImage(GUIStopwatches[id].startstopimg)
   DeleteImage(GUIStopwatches[id].resetimg)
   DeleteImage(GUIStopwatches[id].handimg)
endfunction
```

SetGUIStopwatchControls()

Mini-spec:

FUNCTION NAME	:	SetGUIStopwatchControls
PARAMETERS		
In	:	id : integer flag : integer
Out	:	None
PRE-CONDITION	:	*id* is a valid stopwatch ID.
DESCRIPTION	:	Enables/disables user interaction with the stopwatch. If *flag* is 0, then the *start/stop* and *reset* buttons no longer react to user clicks. All other values of *flag* enable the stopwatch controls.

Code:

```
//*** Enables/disables user interaction with stopwatch ***
function SetGUIStopwatchControls(id as integer, flag as integer)
   //*** If the id is invalid, exit zero ***
   if id < 1 or id > GUIStopwatches.length
      exitfunction
   endif
   //*** If stopwatch deleted, exit zero ***
   if GUIStopwatches[id].deleted = 1
      exitfunction
   endif
   //*** Set controls ***
   if flag = 0
      GUIStopwatches[id].enabled = 0
   else
      GUIStopwatches[id].enabled = 1
   endif
 endfunction
```

Missing Functions

Another common mistake when creating new code is to realise only at the coding stage that some useful operations have been omitted from the design.

In this case, there are two immediately obvious routines that are missing. These are *GetGUIStopwatchControls()* which would return the current setting of the *enabled* field defined in *StopwatchType* and *GetGUIStopwatchExists()* which returns 1 if a stopwatch of a given ID exists and returns zero, if the watch does not exist.

Activity 9

In *StopwatchTest*, add the functions *GetGUIStopwatchTime()*, *DeleteGUIStopwatch()*, *SetGUIStopwatchControls()*.

Create a mini-spec for *GetGUIStopwatchControls()* and implement it within the project.

Create a mini-spec for *GetGUIStopwatchExists()* and implement it within the project.

To test the routines, add a `Print` statement to the `do...loop` of the main program which displays the value returned by *GetGUIStopwatchTime()*.
In *HandleOther()*, add code to do the following:

```
IF user has control of watch AND 5 seconds or more has passed THEN
    Disable user control
ENDIF
IF 15 seconds or more has passed  AND the stopwatch exists THEN
    Delete the Stopwatch
ENDIF
```

We have tested each of the stopwatch functions we have created and they appear to be working correctly, so we'll move on to the final three functions.

SetGUIStopwatchPosition()

When we specify the position of a stopwatch, we are, in fact, specifying the position of the top-left corner of the face with the control buttons appearing above the specified *y* position. To move a stopwatch to a new location we need to move the sprites and text label that make up the watch.

Mini-spec:

```
FUNCTION NAME   :   SetGUIStopwatchPosition
PARAMETERS
    In          :   id      : integer
                    x       : real
                    y       : real
    Out         :   None
PRE-CONDITION   :   id is a valid stopwatch ID.
DESCRIPTION     :   Moves the watch with ID id to world coordinates
                    (x, y).
```

Code:

```
//*** Positions stopwatch face top-left at (x,y) ***
function SetGUIStopwatchPosition(id as integer, x as float,
↳y as float)
   //*** If id invalid, exit ***
   if id < 1 or id > GUIStopwatches.length
      exitfunction
   endif
   //*** If stopwatch deleted, exit ***
   if GUIStopwatches[id].deleted = 1
      exitfunction
   endif

   //*** Create a set of shorter names ***
   faceid = GUIStopwatches[id].facespr
   ssid = GUIStopwatches[id].ssspr
   resetid = GUIStopwatches[id].resetspr
   handid = GUIStopwatches[id].handspr
   txtid = GUIStopwatches[id].timetxt
   //*** Position face ***
   SetSpritePosition(faceid,x,y)
   //*** Position start/stop button ***
   SetSpritePositionByOffset(ssid,x+GetSpriteWidth(faceid)/2.0,y -
   ↳GetSpriteHeight(ssid)/2.1)
   //*** Position reset button ***
   SetSpritePositionByOffset(resetid,x+GetSpriteWidth(faceid)/1.5,y)
   //*** Position time label ***
   SetTextPosition(txtid,x+GetSpriteWidth(faceid)/1.68, y +
   ↳GetSpriteHeight(faceid)/1.747)
   //*** Position seconds hand ***
   SetSpritePositionByOffset(handid,GetSpriteX(faceid),
   ↳GetSpriteY(faceid) + GetSpriteHeight(faceid)/2.0)
endfunction
```

SetGUIStopwatchSize()

Size, like position, refers to the face sprite only, with the control buttons adding to the overall height (width is unaffected). When we adjust the size of the face we must also adjust the size (and position) of the other elements.

Mini-spec:

FUNCTION NAME	:	SetGUIStopwatchSize
PARAMETERS		
In	:	id : integer
		w : real
		h : real
Out	:	None
PRE-CONDITION	:	*id* is a valid stopwatch ID.
DESCRIPTION	:	Adjusts the size of the stopwatch's face (becomes *w* units wide and *h* units high). Other elements are adjusted in size and position as required.

Code:

```
//*** Resizes the specified stopwatch ***
function SetGUIStopwatchSize(id as integer, w as float, h as float)
   //*** If id invalid, exit ***
   if id < 1 or id > GUIStopwatches.length
      exitfunction
   endif
   //*** If stopwatch deleted, exit ***
   if GUIStopwatches[id].deleted = 1
      exitfunction
   endif
   //*** Create a set of shorter names ***
   faceid = GUIStopwatches[id].facespr
   ssid = GUIStopwatches[id].ssspr
   resetid = GUIStopwatches[id].resetspr
   handid = GUIStopwatches[id].handspr
   txtid = GUIStopwatches[id].timetxt
   x as float
   x = GetSpriteX(faceid)
   y as float
   y = GetSpriteY(faceid)
   //*** Sizeface sprite ***
   SetSpriteSize(faceid,w,h)
   //*** Size and reposition start/stop button ***
   SetSpriteSize(ssid,w*0.15,-1)
   SetSpritePositionByOffset(ssid,x+GetSpriteWidth(faceid)/2.0,y -
   ↳GetSpriteHeight(ssid)/2.1)
   //*** Size and reposition reset button ***
   SetSpriteSize(resetid,w*0.13,-1)
   SetSpritePositionByOffset(resetid,x+GetSpriteWidth(faceid)/1.5,y)
   //*** Size and reposition time label ***
   SetTextSize(txtid,GetSpriteHeight(faceid)*0.08)
   SetTextPosition(txtid,x+GetSpriteWidth(faceid)/1.68, y +
   ↳GetSpriteHeight(faceid)/1.747)
   //*** Size and reposition seconds hand ***
   SetSpriteSize(handid,-1,GetSpriteHeight(faceid)*0.22)
   SetSpriteOffset(handid,GetSpriteWidth(handid)/2,
   ↳GetSpriteHeight(handid))
   SetSpritePositionByOffset(handid,GetSpriteX(faceid)+
   ↳GetSpriteWidth(faceid)/2.0,GetSpriteY(faceid) +
   ↳GetSpriteHeight(faceid)/2.0)
endfunction
```

SetGUIStopwatchDepth()

Mini-spec:

FUNCTION NAME	:	SetGUIStopwatchSize
PARAMETERS		
In	:	id : integer
		d : integer
Out	:	None
PRE-CONDITION	:	*id* is a valid stopwatch ID AND 2 <= d <= 9999
DESCRIPTION	:	Places the face of the watch and the time label on layer *d*. The watch buttons are on layer *d*+1 and the *seconds* hand on layer *d*-1.

Code:

```
//*** Sets the depth of the stopwatch ***
function SetGUIStopwatchDepth(id as integer, d as integer)
   //*** If the id is invalid, exit ***
   if id < 1 or id > GUIStopwatches.length
      exitfunction
   endif
   //*** If stopwatch deleted, exit ***
   if GUIStopwatches[id].deleted = 1
      exitfunction
   endif
   //*** If depth invalid, exit ***
   if d < 2 or d > 9999
      exitfunction
   endif
   //*** Create a set of shorter names ***
   faceid = GUIStopwatches[id].facespr
   ssid = GUIStopwatches[id].ssspr
   resetid = GUIStopwatches[id].resetspr
   handid = GUIStopwatches[id].handspr
   txtid = GUIStopwatches[id].timetxt
   //*** Set the depth of each component ***
   SetSpriteDepth(faceid,d)
   SetSpriteDepth(ssid,GetSpriteDepth(faceid)+1)
   SetSpriteDepth(resetid,GetSpriteDepth(faceid)+1)
   SetTextDepth(txtid,GetSpriteDepth(faceid))
   SetSpriteDepth(handid, GetSpriteDepth(faceid)-1)
endfunction
```

Activity 10

In *StopwatchTest*, add the functions *SetGUIStopwatchPosition()*, *SetGUIStopwatchSize()*, *SetGUIStopwatchDepth()*.
To test the routines, modify *HandleOther()* to perform the following logic:

```
Update stopwatch display
IF user has control of watch AND 5 seconds or more has passed THEN
      Set watch size to a random value between 30 and 60
      Set watch position to a random position (0 to 60, 0 to 60)
      Set watch layer to 20
      Disable user control
ENDIF
IF watch not deleted AND 10 seconds or more has passed THEN
      Delete the stopwatch
ENDIF
```

Adding the *Stopwatch* Widget to the *GUILibrary.agc* File

Now that we appear to have the Stopwatch widget fully functioning, we can copy its data structure and associated functions to our existing *GUILibrary.agc* file in the *Function Library* folder thereby making it easily accessable for use in other projects.

Activity 11

Open *GUILibrary.agc* and add the code for the Stopwatch widget to the file before resaving it.

Keep on Testing

It seems like we have our stopwatch working perfectly. But our testing has been very superficial so there's a likely chance we've missed something. And we have.

Activity 12

Start a new project called *StopwatchTest2* and create two stopwatches. Run the program.

What happens when you try starting and stopping each watch independently by clicking on the watch buttons?

Our problem this time was caused by our previous decision to use *Timer()* and *ResetTimer()* when starting and updating the stopwatch. The call to *ResetTimer()* in the *StartGUIStopwatch()* function meant that the elapsed time used by every stopwatch was reset rather than just the specific watch whose *start/stop* button had been pressed.

To solve this problem, we'll start by adding a new field to the *StopwatchType* data structure. This new field will record the absolute time (from the start of the app's execution) at which the stopwatch was last started:

```
type StopwatchType
   faceimg     as integer  //ID of face image
   startstopimg as integer    //ID for start/stop button image
   resetimg    as integer  //ID for reset button image
   handimg     as integer  //ID of watch hand image
   facespr     as integer  //ID of face sprite
   ssspr       as integer  //ID of start/stop button sprite
   resetspr    as integer  //ID of reset button sprite
   handspr     as integer  //ID of watch hand sprite
   timetxt     as integer  //ID of text showing elapsed time
   time        as integer  //Elapsed time since started (msecs)
   stoptime    as integer  //Time on watch when stopped
   laststarted as integer  //App time of last start
   state       as integer  //0: stopped, 1: running
   enabled     as integer  //Buttons disabled (1:yes, 0:no)
   deleted     as integer  //(0: exists, 1:deleted)
endtype
```

This new field should be set to zero in *CreateGUIStopwatch()*:

```
//*** Stopwatch never been started ***
GUIStopwatches[id].laststarted = 0
```

Next, we'll have this new value set by *StartGUIStopwatch()* where we'll also remove the call to *ResetTimer()*:

```
//*** Starts stopwatch running ***
function StartGUIStopwatch(id as integer)
   //*** If the id is invalid, exit ***
   if id < 1 or id > GUIStopwatches.length
      exitfunction
   endif
   //*** If stopwatch deleted, exit ***
   if GUIStopwatches[id].deleted = 1
      exitfunction
   endif
   //*** Set to running state ***
   GUIStopwatches[id].state = 1
```

```
    //*** Record app time when watch started ***
    GUIStopwatches[id].laststarted = GetMilliseconds()
endfunction
```

Finally, in *UpdateGUIStopwatch()* we'll use the new field to help calculate the time showing on the watch:

```
GUIStopwatches[id].time = GUIStopwatches[id].stoptime +
↳(GetMilliseconds() - GUIStopwatches[id].laststarted)
```

Activity 13

In *GUILibrary.agc* make the changes specified above.

What happens this time when you try starting and stopping each watch independently?

Things will be Better Next Time Round

No matter how much effort we put into design before implementation, the chances are when we've spent a long time coding a data structure and its associated functions we'll come up with better ideas on how we'd modify it next time round.

In the case of the *Stopwatch* widget, if we'd written *GetGUIStopwatchExists()* earlier, we could have replaced the code

```
//*** If the id is invalid, exit ***
if id < 1 or id > GUIStopwatches.length
   exitfunction
endif
//*** If stopwatch deleted, exit ***
if GUIStopwatches[id].deleted = 1
   exitfunction
endif
```

which appears in most of the functions, with the code

```
if NOT GetGUIStopwatchExists(id)
   exitfunction
```

Also, it would have been better to allow *SetGUIStopwatchControls()* to enable/disable the watch buttons on an individual basis. That way, when using the stopwatch in a game, we could allow the player to stop the timing but not reset it.

This widget (an improved version) and others will be included in the next version of **Hands On AppGameKit Studio Volume 2** (available free to those who have previously purchased the book directly from **Digital Skills** (*www.digital-skills.co.uk*)).

Solutions

Activity 1

Code for *StopwatchTest*:

```
// Project: StopwatchTest
// Created: 20-04-21
//*** Include required library files ***
#include "../Function Library/CoreLibrary.agc"


//*************************************
//***   Stopwatch Data & Functions   ***
//*************************************
type StopwatchType
  faceimg     as integer  //ID of face image
  startstopimg  as integer  //ID for start/stop
                            ↳button image
  resetimg    as integer  //ID for reset button
                          ↳image
  handimg     as integer  //ID of watch hand image
  facespr     as integer  //ID of face sprite
  ssspr       as integer  //ID of start/stop button
                          ↳sprite
  resetspr    as integer  //ID of reset button
                          ↳sprite
  handspr     as integer  //ID of watch hand sprite
  timetxt     as integer  //ID of text showing
                          ↳elapsed time
  time        as integer  //Elapsed time since
                          ↳started (msecs)
  state       as integer  //Stopwatch state (0:
                          ↳stopped, 1: running)
  disabled    as integer  //Buttons disabled (1:yes,
                          ↳0:no)
  deleted     as integer  //(0: exists, 1:deleted)
endtype

global GUIStopwatches as StopwatchType[0]

//*** Creates stopwatch. Uses images facefile+
↳facefileshad+facefiless+facefilereset+facefilehand ***
//*** The time elapsed is set to zero and the watch
↳is stopped, buttons are enabled              ***
function CreateGUIStopwatch(x as float, y as float,
↳w as float, h as float, facefile as string)
  //*** Add cell to stopwatch list ***
  GUIStopwatches.length = GUIStopwatches.length+1
  //*** Last position in array is new button's ID
  ↳***
  id = GUIStopwatches.length
  //*** Load images ***
  GUIStopwatches[id].faceimg = LoadImage(facefile)
  GUIStopwatches[id].startstopimg = LoadImage(Left(
  ↳facefile,Len(facefile)-4)+"ss"+Right(facefile,4))
  GUIStopwatches[id].resetimg = LoadImage(Left(
  ↳facefile,Len(facefile)-4)+"reset"+Right(facefile,4))
  GUIStopwatches[id].handimg = LoadImage(Left(
  ↳facefile,Len(facefile)-4)+"hand"+Right(facefile,4))
  //*** Create, size and position face sprite ***
  GUIStopwatches[id].facespr = CreateSprite(
  ↳GUIStopwatches[id].faceimg)
  faceid = GUIStopwatches[id].facespr
  SetSpriteSize(faceid,w,h)
  SetSpritePosition(faceid,x,y)
  //*** Create, size, position and layer start/stop
  ↳button ***
  GUIStopwatches[id].ssspr = CreateSprite(
  ↳GUIStopwatches[id].startstopimg)
  ssid = GUIStopwatches[id].ssspr
  SetSpriteSize(ssid,w*0.15,-1)
  SetSpritePositionByOffset(ssid,x+GetSpriteWidth(
  ↳faceid)/2.0,y-GetSpriteHeight(ssid)/2.1)
  SetSpriteDepth(ssid,GetSpriteDepth(faceid)+1)
  //*** Create, size, position and layer reset
  ↳button ***
  GUIStopwatches[id].resetspr =
  ↳CreateSprite(GUIStopwatches[id].resetimg)
  resetid = GUIStopwatches[id].resetspr
  SetSpriteSize(resetid,w*0.13,-1)
  SetSpritePositionByOffset(resetid,x+
  ↳GetSpriteWidth(faceid)/1.5,y)
  SetSpriteDepth(resetid,GetSpriteDepth(faceid)+1)
  //*** Create, align, size, position and layer time
  ↳label ***
  GUIStopwatches[id].timetxt = CreateText("0:00")
  txtid = GUIStopwatches[id].timetxt
  SetTextAlignment(txtid,2)
  SetTextSize(txtid,GetSpriteHeight(faceid)*0.08)
  SetTextPosition(txtid,x+GetSpriteWidth(faceid)/
  ↳1.68, y+GetSpriteHeight(faceid)/1.747)
  SetTextDepth(txtid,GetSpriteDepth(faceid))
  //*** Create, size, position and layer seconds
  ↳hand ***
  GUIStopwatches[id].handspr = CreateSprite(
  ↳GUIStopwatches[id].handimg)
  handid = GUIStopwatches[id].handspr
  SetSpriteSize(handid,-1,GetSpriteHeight(
  ↳faceid)*0.22)
  SetSpriteOffset(handid,GetSpriteWidth(handid)/2,
  ↳GetSpriteHeight(handid))
  SetSpritePositionByOffset(handid,GetSpriteX(
  ↳faceid)+ GetSpriteWidth(faceid)/2.0,
  GetSpriteY(faceid)+GetSpriteHeight(faceid)/2.0)
  SetSpriteDepth(handid, GetSpriteDepth(faceid)-1)
  //*** Stopwatch not deleted ***
  GUIStopwatches[id].deleted = 0
  //*** Stopwatch not disabled ***
  GUIStopwatches[id].disabled = 0
  //*** Stopwatch stopped ***
  GUIStopwatches[id].state = 0
endfunction id



type GameType
  id   as integer  //ID of Stopwatch
endtype

global g as GameType

//*************************************
//***          Main program          ***
//*************************************
InitialiseScreen(1000,750,"StopwatchTest", 0x585858,
↳%1111)
LoadResources()
CreateInitialLayout()
do
  in = GetUserInput()
  HandleUserInput(in)
  HandleOther()
  Sync()
loop

//*************************************
//***          Functions             ***
//*************************************
function LoadResources()
endfunction

function CreateInitialLayout()
  g.id = CreateGUIStopwatch(30,30,40,-1,
  ↳"swatch.png")
endfunction

function GetUserInput()
  result = 1
endfunction result

function HandleUserInput(in as integer)
endfunction

function HandleOther()
endfunction
```

Activity 2

Modified code for *StopwatchTest*:

```
// Project: StopwatchTest
// Created: 20-04-21
//*** Include required library files ***
#include "../Function Library/CoreLibrary.agc"


//*************************************
//***   Stopwatch Data & Functions   ***
//*************************************
type StopwatchType
  faceimg     as integer  //ID of face image
  startstopimg  as integer  //ID for start/stop
                            ↳button image
  resetimg    as integer  //ID for reset button
                          ↳image
```

```
  handimg     as integer  //ID of watch hand image
  facespr     as integer  //ID of face sprite
  ssspr       as integer  //ID of start/stop button
                          ↳sprite
  resetspr    as integer  //ID of reset button
                          ↳sprite
  handspr     as integer  //ID of watch hand sprite
  timetxt     as integer  //ID of text showing
                          ↳elapsed time
  time        as integer  //Elapsed time since
                          ↳started (msecs)
  state       as integer  //Stopwatch state (0:
                          ↳stopped, 1: running)
  disabled    as integer  //Buttons disabled (1:yes,
                          ↳0:no)
  deleted     as integer  //(0: exists, 1:deleted)
endtype

global GUIStopwatches as StopwatchType[0]

//*** Creates stopwatch. Uses images facefile+
↳facefileshad+facefiless+facefilereset+facefilehand ***
//*** The time elapsed is set to zero and the watch
↳is stopped, buttons are enabled                ***
function CreateGUIStopwatch(x as float, y as float,
↳w as float, h as float, facefile as string)
  //*** Add cell to stopwatch list ***
  GUIStopwatches.length = GUIStopwatches.length+1
  //*** Last position in array is new button's ID
  ↳***
  id = GUIStopwatches.length
  //*** Load images ***
  GUIStopwatches[id].faceimg = LoadImage(facefile)
  GUIStopwatches[id].startstopimg = LoadImage(Left(
  ↳facefile,Len(facefile)-4)+"ss"+Right(facefile,4))
  GUIStopwatches[id].resetimg = LoadImage(Left(
  ↳facefile,Len(facefile)-4)+"reset"+Right(facefile,4))
  GUIStopwatches[id].handimg = LoadImage(Left(
  ↳facefile,Len(facefile)-4)+"hand"+Right(facefile,4))
  //*** Create, size and position face sprite ***
  GUIStopwatches[id].facespr = CreateSprite(
  ↳GUIStopwatches[id].faceimg)
  faceid = GUIStopwatches[id].facespr
  SetSpriteSize(faceid,w,h)
  SetSpritePosition(faceid,x,y)
  //*** Create, size, position and layer start/stop
  ↳button ***
  GUIStopwatches[id].ssspr = CreateSprite(
  ↳GUIStopwatches[id].startstopimg)
  ssid = GUIStopwatches[id].ssspr
  SetSpriteSize(ssid,w*0.15,-1)
  SetSpritePositionByOffset(ssid,x+GetSpriteWidth(
  ↳faceid)/2.0,y-GetSpriteHeight(ssid)/2.1)
  SetSpriteDepth(ssid,GetSpriteDepth(faceid)+1)
  //*** Create, size, position and layer reset
  ↳button ***
  GUIStopwatches[id].resetspr =
  ↳CreateSprite(GUIStopwatches[id].resetimg)

  resetid = GUIStopwatches[id].resetspr
  SetSpriteSize(resetid,w*0.13,-1)
  SetSpritePositionByOffset(resetid,x+
  ↳GetSpriteWidth(faceid)/1.5,y)
  SetSpriteDepth(resetid,GetSpriteDepth(faceid)+1)
  //*** Create, align, size, position and layer time
  ↳label ***
  GUIStopwatches[id].timetxt = CreateText("0:00")
  txtid = GUIStopwatches[id].timetxt
  SetTextAlignment(txtid,2)
  SetTextSize(txtid,GetSpriteHeight(faceid)*0.08)
  SetTextPosition(txtid,x+GetSpriteWidth(faceid)/
  ↳1.68, y+GetSpriteHeight(faceid)/1.747)
  SetTextDepth(txtid,GetSpriteDepth(faceid))
  //*** Create, size, position and layer seconds
  ↳hand ***
  GUIStopwatches[id].handspr = CreateSprite(
  ↳GUIStopwatches[id].handimg)

  handid = GUIStopwatches[id].handspr
  SetSpriteSize(handid,-1,GetSpriteHeight(
  ↳faceid)*0.22)
  SetSpriteOffset(handid,GetSpriteWidth(handid)/2,
  ↳GetSpriteHeight(handid))
  SetSpritePositionByOffset(handid,GetSpriteX(
  ↳faceid)+ GetSpriteWidth(faceid)/2.0,
  ↳GetSpriteY(faceid)+GetSpriteHeight(faceid)/2.0)
  SetSpriteDepth(handid, GetSpriteDepth(faceid)-1)
  //*** Stopwatch not deleted ***
  GUIStopwatches[id].deleted = 0
  //*** Stopwatch not disabled ***
  GUIStopwatches[id].disabled = 0
  //*** Stopwatch stopped ***
  GUIStopwatches[id].state = 0
endfunction id


//*** Starts stopwatch running ***
function StartGUIStopwatch(id as integer)
  //*** If the id is invalid, exit ***
  if id < 1 or id > GUIStopwatches.length
    exitfunction
  endif
  //*** If stopwatch deleted, exit ***
  if GUIStopwatches[id].deleted = 1
    exitfunction
  endif
  //*** Set to running state ***
  GUIStopwatches[id].state = 1
endfunction


//*** Stops the stopwatch running ***
function StopGUIStopwatch(id as integer)
  //*** If the id is invalid, exit ***
  if id < 1 or id > GUIStopwatches.length
    exitfunction
  endif
  //*** If stopwatch deleted, exit ***
  if GUIStopwatches[id].deleted = 1
    exitfunction
  endif
  //*** Set to stopped state ***
  GUIStopwatches[id].state = 0
endfunction

//*** Returns the current state of the ***
//*** stopwatch (running or stopped)   ***
function GetGUIStopwatchState(id as integer)
  //*** If the id is invalid, exit  -1***
  if id < 1 or id > GUIStopwatches.length
    exitfunction -1
  endif
  //*** If stopwatch deleted, exit -1 ***
  if GUIStopwatches[id].deleted = 1
    exitfunction -1
  endif
  //*** Set result to watch current state ***
  result = GUIStopwatches[id].state
endfunction result


//*** Update the stopwatch visuals, moving   ***
//*** seconds hand and changing digital time ***
function UpdateGUIStopwatch(id as integer)
  //*** If the id is invalid, exit ***
  if id < 1 or id > GUIStopwatches.length
    exitfunction
  endif
  //*** If stopwatch deleted, exit ***
  if GUIStopwatches[id].deleted = 1
    exitfunction
  endif//*** If watch stopped, exit ***
  if GUIStopwatches[id].state = 0
    exitfunction
  endif
  //*** Record the time passed ***
  GUIStopwatches[id].time = GetMilliseconds()
  msecs = GUIStopwatches[id].time
  //*** Modify angle of seconds hand to show seconds
  ↳into current minute ***
  SetSpriteAngle(GUIStopwatches[id].handspr,
  ↳msecs*6.0/1000.0)
  //*** Update the digital time on label ***
  SetTextString(GUIStopwatches[id].timetxt,
  ↳ToMinutesSeconds(msecs))
endfunction


//************************************
//***  Stopwatch Helper Functions    ***
//************************************
function ToMinutesSeconds(msecs as integer)
  totalsecs = msecs /1000
  mins = totalsecs / 60
  secs = Mod(totalsecs,60)
  result as string
  result = Str(mins)+":"+Right("0"+Str(secs),2)
endfunction result
```

```
type GameType
  id   as integer  //ID of Stopwatch
endtype

global g as GameType
//*************************************
//***          Main program           ***
//*************************************
InitialiseScreen(1000,750,"StopwatchTest", 0x585858,
↳%1111)
LoadResources()
CreateInitialLayout()
do
  in = GetUserInput()
  HandleUserInput(in)
  HandleOther()
  Sync()
loop

//*************************************
//***           Functions             ***
//*************************************
function LoadResources()
endfunction

function CreateInitialLayout()
  g.id = CreateGUIStopwatch(30,30,40,-1,
  ↳"swatch.png")
  StartGUIStopwatch(g.id)
endfunction

function GetUserInput()
  result = 1
endfunction result

function HandleUserInput(in as integer)
endfunction

function HandleOther()
  UpdateGUIStopwatch(g.id)
endfunction
```

The stopwatch should show the elapsed time correctly.

Activity 3

To check that the watch stops, *HandleOther()* is modified to:

```
function HandleOther()
  UpdateGUIStopwatch(g.id)
  if GetMilliseconds() > 5000
    StopGUIStopwatch(g.id)
  endif
endfunction
```

The watch should stop after 5 seconds.

Activity 4

To check that the watch runs as expected after a delay of five seconds, we'll start by removing the call to *StartGUIStopwatch()* in *CreateInitialLayout()*:

```
function CreateInitialLayout()
  g.id = CreateGUIStopwatch(30,30,40,-1,
  ↳"swatch.png")
endfunction
```

Next, we'll change *HandleOther()* to call *StartGUIStopwatch()* rather than *StopGUIStopwatch()*:

```
function HandleOther()
  UpdateGUIStopwatch(g.id)
  if GetMilliseconds() > 5000
    StartGUIStopwatch(g.id)
  endif
endfunction
```

Although the stopwatch does start after five seconds, it doesn't start at zero. Instead it jumps immediately to show that five seconds has passed. This is not the reaction we would want since the stopwatch is meant to show the time that has elapsed since it was placed in *running* mode.

Activity 5

Modified code for *StartGUIStopwatch()*:

```
//*** Starts stopwatch running ***
function StartGUIStopwatch(id as integer)
  //*** If the id is invalid, exit ***
  if id < 1 or id > GUIStopwatches.length
    exitfunction
  endif
  //*** If stopwatch deleted, exit ***
  if GUIStopwatches[id].deleted = 1
    exitfunction
  endif
  //*** Set to running state ***
  GUIStopwatches[id].state = 1
  //*** Set elapsed time to zero ***
  ResetTimer()
endfunction
```

Modified code for *UpdateGUIStopwatch()*:

```
/*** Update the stopwatch visuals, moving   ***
//*** seconds hand and changing digital time ***
function UpdateGUIStopwatch(id as integer)
  //*** If the id is invalid, exit ***
  if id < 1 or id > GUIStopwatches.length
    exitfunction
  endif
  //*** If stopwatch deleted, exit ***
  if GUIStopwatches[id].deleted = 1
    exitfunction
  endif//*** If watch stopped, exit ***
  if GUIStopwatches[id].state = 0
    exitfunction
  endif
  //*** Record the time passed ***
  GUIStopwatches[id].time = Timer()*1000
  msecs = GUIStopwatches[id].time
  //*** Modify angle of seconds hand to show seconds
  ↳into current minute ***
  SetSpriteAngle(GUIStopwatches[id].handspr,
  ↳msecs*6.0/1000.0)
  //*** Update the digital time on label ***
  SetTextString(GUIStopwatches[id].timetxt,
  ↳ToMinutesSeconds(msecs))
endfunction
```

Although the stopwatch now starts at zero, it keeps resetting every five seconds!

By modifying *HandleOther()* to read

```
function HandleOther()
  UpdateGUIStopwatch(g.id)
  if GetMilliseconds() > 5000
  ↳and GetGUIStopwatchState(g.id) = 0
    StartGUIStopwatch(g.id)
  endif
endfunction
```

the resetting problem is eliminated.

Activity 6

After adding the code for *ResetGUIStopwatch()*, the main program functions *CreateInitialLayout()* and *HandleOther()* program's code should be changed to:

```
function CreateInitialLayout()
  g.id = CreateGUIStopwatch(30,30,40,-1,
  ↳"swatch.png")
  StartGUIStopwatch(g.id)
endfunction


function HandleOther()
  UpdateGUIStopwatch(g.id)
  if GetMilliseconds() > 5000 and
  ↳GetGUIStopwatchState(g.id) = 1
    ResetGUIStopwatch(g.id)
  endif
endfunction
```

The *seconds* hand should now rotate as far as the 5 seconds mark before reseting itself to the zero position. The digital time should also reset to 0:00.

Activity 7

After adding the code for *HandleGUIStopwatch()*, the following changes are required in the main program's functions:

```
function HandleUserInput(in as integer)
  HandleGUIStopwatch(g.id)
endfunction


function HandleOther()
  UpdateGUIStopwatch(g.id)
endfunction
```

When we test the program, the buttons move up and down as expected. Stop and reset operations perform as expected.

But, when we restart the stopwatch it always returns to zero rather than continuing on from its previous setting.

Activity 8

After making the changes, all operations perform as expected.

Activity 9

FUNCTION NAME	:	GetGUIStopwatchControls	
PARAMETERS			
In	:	id	: integer
Out	:	result	: integer
PRE-CONDITION	:	*id* is a valid stopwatch ID.	
DESCRIPTION	:	Sets *result* to 1 if the user can control the start/stop and reset buttons. *result* is set to zero if user cannot use controls. *result* is set to -1 if id is invalid.	

Code:

```
//*** Returns 1 if user can interact ***
//*** with stopwatch buttons, else   ***
//*** zero is returned               ***
function GetGUIStopwatchControls(id as integer)
  //*** If the id is invalid, exit -1 ***
  if id < 1 or id > GUIStopwatches.length
    exitfunction -1
  endif
  //*** If stopwatch deleted, exit -1 ***
  if GUIStopwatches[id].deleted = 1
    exitfunction -1
  endif
  result = GUIStopwatches[id].enabled
endfunction result
```

FUNCTION NAME	:	GetGUIStopwatchExists	
PARAMETERS			
In	:	id	: integer
Out	:	result	: integer
PRE-CONDITION	:	None	
DESCRIPTION	:	Sets *result* to 1 if a stopwatch with an ID of *id* exists otherwise *result* is set to zero.	

Code:

```
//*** Returns 1 if watch id exists,  ***
//*** else returns zero              ***
function GetGUIStopwatchExists(id as integer)
  result = 1
  //*** If the id is invalid, result is 0 ***
  if id < 1 or id > GUIStopwatches.length
    result = 0
  endif
  //*** If stopwatch deleted, result is 0 ***
  if GUIStopwatches[id].deleted = 1
    result = 0
  endif
endfunction result
```

Modified code for

```
//**************************************
//***          Main program          ***
//**************************************
InitialiseScreen(1000,750,"StopwatchTest", 0x585858,
↳%1111)
LoadResources()
CreateInitialLayout()
do
  in = GetUserInput()
  HandleUserInput(in)
  HandleOther()
  Print(GetGUIStopwatchTime(g.id))
  Sync()
loop


function HandleOther()
  UpdateGUIStopwatch(g.id)
  if GetGUIStopwatchControls(g.id) = 1 and
  ↳GetMilliseconds() > 5000
    SetGUIStopwatchControls(g.id,0)
  endif
  if GetMilliseconds() > 15000 and
  ↳GetGUIStopwatchExists(g.id)
    DeleteGUIStopwatch(g.id)
  endif
endfunction
```

Activity 10

Modified code for *HandleOther()*:

```
function HandleOther()
  UpdateGUIStopwatch(g.id)
  if GetGUIStopwatchControls(g.id) = 1 and
  ↳GetMilliseconds() > 5000
    SetGUIStopwatchSize(g.id,Random2(30,60),-1)
    SetGUIStopwatchPosition(g.id,Random2(0,60),
    ↳Random2(0,60))
    SetGUIStopwatchDepth(g.id,20)
    SetGUIStopwatchControls(g.id,0)
  endif
  if GetGUIStopwatchExists(g.id) and
  ↳GetMilliseconds() > 10000
    DeleteGUIStopwatch(g.id)
  endif
endfunction
```

The new functions operate as expected.

Activity 11

No solution required.

Activity 12

Code for *StopwatchTest2*:

```
// Project: StopwatchTest2
// Created: 20-04-25

//*** Include required library files ***
#include "../Function Library/CoreLibrary.agc"
#include "../Function Library/GUILibrary2.agc"

type GameType
  id1    as integer
  id2    as integer
endtype

global g as GameType

//**************************************
//***          Main program          ***
//**************************************
```

```
InitialiseScreen(1000,750,"StopwatchTest2",
0x585858, %1111)
LoadResources()
CreateInitialLayout()
do
  in = GetUserInput()
  HandleUserInput(in)
  HandleOther()
  Sync()
loop

//*************************************
//***          Functions            ***
//*************************************
function LoadResources()
endfunction

function CreateInitialLayout()
  g.id1 = CreateGUIStopwatch(10,20,30,-1,
  ↳"swatch.png")
  g.id2 = CreateGUIStopwatch(60,20,30,-1,
  ↳"swatch.png")
endfunction

function GetUserInput()
  result = 1
endfunction result

function HandleUserInput(in as integer)
  HandleGUIStopwatch(g.id1)
  HandleGUIStopwatch(g.id2)
endfunction

function HandleOther()
  UpdateGUIStopwatch(g.id1)
  UpdateGUIStopwatch(g.id2)
endfunction
```

The problem is that when we start the second watch after previously starting the first, the first watch resets itself to zero.

Activity 13

This time the two stopwatches operate as expected.

The final listing of *Stopwatch* widget's code in *GUILibrary.agc*:

```
//*************************************
//***   Stopwatch Data & Functions   ***
//*************************************
type StopwatchType
  faceimg     as integer  //ID of face image
  startstopimg  as integer  //ID for start/stop
                            ↳button image
  resetimg    as integer  //ID for reset button
                          ↳image
  handimg     as integer  //ID of watch hand image
  facespr     as integer  //ID of face sprite
  ssspr       as integer  //ID of start/stop button
                          ↳sprite
  resetspr    as integer  //ID of reset button
                          ↳sprite
  handspr     as integer  //ID of watch hand sprite
  timetxt     as integer  //ID of text showing
                          ↳elapsed time
  time        as integer  //Elapsed time since
                          ↳started (msecs)
  stoptime    as integer  //Time on watch when
                          ↳stopped
  laststarted as integer  //App time of last start
  state       as integer  //0: stopped, 1: running
  enabled     as integer  //Buttons enabled (1:yes,
                          ↳0:no)
  deleted     as integer  //(0: exists, 1:deleted)
endtype

global GUIStopwatches as StopwatchType[0]


//*** Creates stopwatch. Uses images facefile+
facefileshad+facefilesss+facefilereset+facefilehand ***

//*** The time elapsed is set to zero and the watch
↳is stopped, buttons are enabled              ***
function CreateGUIStopwatch(x as float, y as float,
↳w as float, h as float, facefile as string)
  //*** Add cell to stopwatch list ***
  GUIStopwatches.length = GUIStopwatches.length+1
  //*** Last position in array is new button's ID
  ↳***
  id = GUIStopwatches.length
  //*** Load images ***
  GUIStopwatches[id].faceimg = LoadImage(facefile)
  GUIStopwatches[id].startstopimg = LoadImage(Left(
  ↳facefile,Len(facefile)-4)+"ss"+Right(facefile,4))
  GUIStopwatches[id].resetimg = LoadImage(Left(
  ↳facefile,Len(facefile)-4)+"reset"+Right(facefile,4))
  GUIStopwatches[id].handimg = LoadImage(Left(
  ↳facefile,Len(facefile)-4)+"hand"+Right(facefile,4))
  //*** Create, size and position face sprite ***
  GUIStopwatches[id].facespr = CreateSprite(
  ↳GUIStopwatches[id].faceimg)
  faceid = GUIStopwatches[id].facespr
  SetSpriteSize(faceid,w,h)
  SetSpritePosition(faceid,x,y)
  //*** Create, size, position and layer start/stop
  ↳button ***
  GUIStopwatches[id].ssspr = CreateSprite(
  ↳GUIStopwatches[id].startstopimg)
  ssid = GUIStopwatches[id].ssspr
  SetSpriteSize(ssid,w*0.15,-1)
  SetSpritePositionByOffset(ssid,x+GetSpriteWidth(
  ↳faceid)/2.0,y-GetSpriteHeight(ssid)/2.1)
  SetSpriteDepth(ssid,GetSpriteDepth(faceid)+1)
  //*** Create, size, position and layer reset
  ↳button ***
  GUIStopwatches[id].resetspr = CreateSprite(
  ↳GUIStopwatches[id].resetimg)
  resetid = GUIStopwatches[id].resetspr
  SetSpriteSize(resetid,w*0.13,-1)
  SetSpritePositionByOffset(resetid,x +
  ↳GetSpriteWidth(faceid)/1.5,y)
  SetSpriteDepth(resetid,GetSpriteDepth(faceid)+1)
  //*** Create, align, size, position and layer time
  ↳label ***
  GUIStopwatches[id].timetxt = CreateText("0:00")
  txtid = GUIStopwatches[id].timetxt
  SetTextAlignment(txtid,2)
  SetTextSize(txtid,GetSpriteHeight(faceid)*0.08)
  SetTextPosition(txtid,x+GetSpriteWidth(faceid)/
  ↳1.68, y+GetSpriteHeight(faceid)/1.747)
  SetTextDepth(txtid,GetSpriteDepth(faceid))
  //*** Create, size, position and layer seconds
  ↳hand ***
  GUIStopwatches[id].handspr = CreateSprite(
  ↳GUIStopwatches[id].handimg)
  handid = GUIStopwatches[id].handspr
  SetSpriteSize(handid,-1, GetSpriteHeight(
  ↳faceid)*0.22)
  SetSpriteOffset(handid,GetSpriteWidth(handid)/2,
  ↳GetSpriteHeight(handid))
  SetSpritePositionByOffset(handid,GetSpriteX(faceid)
  ↳+ GetSpriteWidth(faceid)/2.0,
  GetSpriteY(faceid)+GetSpriteHeight(faceid)/2.0)
  SetSpriteDepth(handid, GetSpriteDepth(faceid)-1)
  //*** Stopwatch not deleted ***
  GUIStopwatches[id].deleted = 0
  //*** Stopwatch is enabled ***
  GUIStopwatches[id].enabled = 1
  //*** Stopwatch stopped ***
  GUIStopwatches[id].state = 0
  //*** Stopwatch never been started ***
  GUIStopwatches[id].laststarted = 0
endfunction id

//*** Starts stopwatch running ***
function StartGUIStopwatch(id as integer)
  //*** If the id is invalid, exit ***
  if id < 1 or id > GUIStopwatches.length
    exitfunction
  endif
  //*** If stopwatch deleted, exit ***
  if GUIStopwatches[id].deleted = 1
    exitfunction
  endif
  //*** Set to running state ***
  GUIStopwatches[id].state = 1
  //*** Record app time when watch started ***
  GUIStopwatches[id].laststarted = GetMilliseconds()
endfunction


//*** Stops the stopwatch running ***
function StopGUIStopwatch(id as integer)
  //*** If the id is invalid, exit ***
```

```
  if id < 1 or id > GUIStopwatches.length
    exitfunction
  endif
  //*** If stopwatch deleted, exit ***
  if GUIStopwatches[id].deleted = 1
    exitfunction
  endif
  //*** Set to stopped state ***
  GUIStopwatches[id].state = 0
  //*** Save current time on watch ***
  GUIStopwatches[id].stoptime =
  ↳GUIStopwatches[id].time
endfunction


//*** Returns the current state of the ***
//*** stopwatch (running or stopped)   ***
function GetGUIStopwatchState(id as integer)
  //*** If the id is invalid, exit  -1***
  if id < 1 or id > GUIStopwatches.length
    exitfunction -1
  endif
  //*** If stopwatch deleted, exit -1 ***
  if GUIStopwatches[id].deleted = 1
    exitfunction -1
  endif
  //*** Set result to watch current state ***
  result = GUIStopwatches[id].state
endfunction result


//*** Update the stopwatch visuals, moving   ***
//*** seconds hand and changing digital time ***
function UpdateGUIStopwatch(id as integer)
  //*** If the id is invalid, exit ***
  if id < 1 or id > GUIStopwatches.length
    exitfunction
  endif
  //*** If stopwatch deleted, exit ***
  if GUIStopwatches[id].deleted = 1
    exitfunction
  endif
  //*** If watch stopped, exit ***
  if GUIStopwatches[id].state = 0
    exitfunction
  endif
  //*** Record the time passed ***
  GUIStopwatches[id].time =
  ↳GUIStopwatches[id].stoptime + (GetMilliseconds()-
  ↳GUIStopwatches[id].laststarted)
  msecs = GUIStopwatches[id].time
  //*** Modify angle of seconds hand to show seconds
  ↳into current minute ***
  SetSpriteAngle(GUIStopwatches[id].handspr,
  ↳msecs*6.0/1000.0)
  //*** Update the digital time on label ***
  SetTextString(GUIStopwatches[id].timetxt,
  ↳ToMinutesSeconds(msecs))
endfunction


//*** Resets the time to zero and places ***
//*** the watch in stopped mode         ***
function ResetGUIStopwatch(id as integer)
  //*** If the id is invalid, exit ***
  if id < 1 or id > GUIStopwatches.length
    exitfunction
  endif
  //*** If stopwatch deleted, exit ***
  if GUIStopwatches[id].deleted = 1
    exitfunction
  endif
  //*** Stopwatch stopped ***
  GUIStopwatches[id].state = 0
  //*** No time has been recorded ***
  GUIStopwatches[id].time = 0
  GUIStopwatches[id].stoptime = 0
  //*** Reset second hand and digital time ***
  SetSpriteAngle(GUIStopwatches[id].handspr,0)
  SetTextString(GUIStopwatches[id].timetxt,"0:00")
endfunction


//*** Handles user interaction with watch ***
//*** Allows start/stop and reset         ***
//*** Returns a value indicating action   ***
function HandleGUIStopwatch(id as integer)
  //*** If the id is invalid, exit zero ***
  if id < 1 or id > GUIStopwatches.length
    exitfunction 0
  endif
  //*** If stopwatch deleted, exit zero ***
  if GUIStopwatches[id].deleted = 1
    exitfunction 0
  endif
  result = 0
  //*** If watch disabled, exit zero ***
  if GUIStopwatches[id].enabled = 0
    exitfunction 0
  endif
  result = 0
  //*** If pointer pressed ***
  if GetPointerPressed()
    hit = GetSpriteHit(GetPointerX(), GetPointerY())
    //*** If start/stop button pressed ***
    if hit = GUIStopwatches[id].ssspr
      //*** Show start/stop button moving down and
      ↳back ***
      SetSpriteY(hit, GetSpriteY(hit) +
      ↳GetSpriteHeight(hit)/2.0)
      Sync()
      Sleep(100)
      SetSpriteY(hit, GetSpriteY(hit) -
      ↳GetSpriteHeight(hit)/2.0)
      //*** If watch stopped, start it ***
      if GUIStopwatches[id].state = 0
        StartGUIStopwatch(id)
        GUIStopwatches[id].state = 1
        result = 1
      else //If watch running, stop it
        StopGUIStopwatch(id)
        result = -1
      endif
    //*** If reset pressed ***
    elseif hit = GUIStopwatches[id].resetspr
      //*** Show reset button moving down and back
      ↳***
      SetSpritePosition(hit,GetSpriteX(hit) -
      ↳GetSpriteWidth(hit)/5.0,GetSpriteY(hit) +
      ↳GetSpriteHeight(hit)/4.0)
      Sync()
      Sleep(100)
      SetSpritePosition(hit,GetSpriteX(hit) +
      ↳GetSpriteWidth(hit) / 5.0,GetSpriteY(hit) -
      ↳GetSpriteHeight(hit)/4.0)
      ResetGUIStopwatch(id)
      result = 2
    endif
  endif
endfunction result


//*** Disables user interaction with stopwatch ***
function SetGUIStopwatchControls(id as integer,
↳flag as integer)
  //*** If the id is invalid, exit ***
  if id < 1 or id > GUIStopwatches.length
    exitfunction
  endif
  //*** If stopwatch deleted, exit ***
  if GUIStopwatches[id].deleted = 1
    exitfunction
  endif
  //*** Disable controls ***
  if flag = 0
    GUIStopwatches[id].enabled = 0
  else
    GUIStopwatches[id].enabled = 1
  endif
endfunction


//*** Returns 1 if user can interact ***
//*** with stopwatch buttons, else   ***
//*** zero is returned               ***
function GetGUIStopwatchControls(id as integer)
  //*** If the id is invalid, exit -1 ***
  if id < 1 or id > GUIStopwatches.length
    exitfunction -1
  endif
  //*** If stopwatch deleted, exit -1 ***
  if GUIStopwatches[id].deleted = 1
    exitfunction -1
  endif
  result = GUIStopwatches[id].enabled
endfunction result


//*** Returns the stopwatch's time ***
function GetGUIStopwatchTime(id as integer)
```

```
  //*** If id invalid, exit -1 ***
  if id < 1 or id > GUIStopwatches.length
    exitfunction -1
  endif
  //*** If stopwatch deleted, exit -1 ***
  if GUIStopwatches[id].deleted = 1
    exitfunction -1
  endif
  //*** Return time ***
  result = GUIStopwatches[id].time
endfunction result


//*** Returns 1 if watch id exists,  ***
//*** else returns zero              ***
function GetGUIStopwatchExists(id as integer)
  result = 1
  //*** If the id is invalid, result is 0 ***
  if id < 1 or id > GUIStopwatches.length
    result = 0
  endif
  //*** If stopwatch deleted, result is 0 ***
  if GUIStopwatches[id].deleted = 1
    result = 0
  endif
endfunction result


//*** Deletes stopwatch and resources ***
function DeleteGUIStopwatch(id as integer)
  //*** If id invalid, exit ***
  if id < 1 or id > GUIStopwatches.length
    exitfunction
  endif
  //*** If stopwatch deleted, exit ***
  if GUIStopwatches[id].deleted = 1
    exitfunction
  endif
  //*** Mark as deleted ***
  GUIStopwatches[id].deleted = 1
  //*** Delete sprites ***
  DeleteSprite(GUIStopwatches[id].facespr)
  DeleteSprite(GUIStopwatches[id].ssspr)
  DeleteSprite(GUIStopwatches[id].resetspr)
  DeleteSprite(GUIStopwatches[id].handspr)
  //*** Delete label ***
  DeleteText(GUIStopwatches[id].timetxt)
  //*** Delete images ***
  DeleteImage(GUIStopwatches[id].faceimg)
  DeleteImage(GUIStopwatches[id].startstopimg)
  DeleteImage(GUIStopwatches[id].resetimg)
  DeleteImage(GUIStopwatches[id].handimg)
endfunction


//*** Positions stopwatch face top-left at (x,y) ***
function SetGUIStopwatchPosition(id as integer,
↳x as float, y as float)
  //*** If id invalid, exit ***
  if id < 1 or id > GUIStopwatches.length
    exitfunction
  endif
  //*** If stopwatch deleted, exit ***
  if GUIStopwatches[id].deleted = 1
    exitfunction
  endif
  //*** Create a set of shorter names ***
  faceid = GUIStopwatches[id].facespr
  ssid = GUIStopwatches[id].ssspr
  resetid = GUIStopwatches[id].resetspr
  handid = GUIStopwatches[id].handspr
  txtid = GUIStopwatches[id].timetxt
  //*** Position face ***
  SetSpritePosition(faceid,x,y)
  //*** Position start/stop button ***
  SetSpritePositionByOffset(ssid, x +
  ↳GetSpriteWidth(faceid) / 2.0, y -
  ↳GetSpriteHeight(ssid) / 2.1)
  //*** Position reset button ***
  SetSpritePositionByOffset(resetid,x +
  ↳GetSpriteWidth(faceid) /1.5, y)
  //*** Position time label ***
  SetTextPosition(txtid,x+GetSpriteWidth(faceid) /
  ↳1.68, y + GetSpriteHeight(faceid) / 1.747)
  //*** Position seconds hand ***
  SetSpritePositionByOffset(handid,GetSpriteX(faceid)
  ↳+GetSpriteWidth(faceid)/2.0,GetSpriteY(faceid) +
  ↳GetSpriteHeight(faceid) / 2.0)
endfunction


//*** Resizes the specified stopwatch ***
function SetGUIStopwatchSize(id as integer, w as
float, h as float)
  //*** If id invalid, exit ***
  if id < 1 or id > GUIStopwatches.length
    exitfunction
  endif
  //*** If stopwatch deleted, exit ***
  if GUIStopwatches[id].deleted = 1
    exitfunction
  endif
  //*** Create a set of shorter names ***
  faceid = GUIStopwatches[id].facespr
  ssid = GUIStopwatches[id].ssspr
  resetid = GUIStopwatches[id].resetspr
  handid = GUIStopwatches[id].handspr
  txtid = GUIStopwatches[id].timetxt
  x as float
  x = GetSpriteX(faceid)
  y as float
  y = GetSpriteY(faceid)
  //*** Sizeface sprite ***
  SetSpriteSize(faceid,w,h)
  //*** Size and reposition start/stop button ***
  SetSpriteSize(ssid, w * 0.15, -1)
  SetSpritePositionByOffset(ssid, x +
  ↳GetSpriteWidth(faceid) / 2.0, y -
  ↳GetSpriteHeight(ssid) / 2.1)
  //*** Size and reposition reset button ***
  SetSpriteSize(resetid, w * 0.13,-1)
  SetSpritePositionByOffset(resetid,x +
  ↳GetSpriteWidth(faceid) / 1.5, y)
  //*** Size and reposition time label ***
  SetTextSize(txtid,GetSpriteHeight(faceid)*0.08)
  SetTextPosition(txtid,x + GetSpriteWidth(faceid) /
  ↳1.68, y + GetSpriteHeight(faceid) / 1.747)
  //*** Size and reposition seconds hand ***
  SetSpriteSize(handid, -1, GetSpriteHeight(faceid)
  ↳* 0.22)
  SetSpriteOffset(handid,GetSpriteWidth(handid) / 2,
  ↳GetSpriteHeight(handid))
  SetSpritePositionByOffset(handid,GetSpriteX(faceid)
  ↳+ GetSpriteWidth(faceid) / 2.0,GetSpriteY(faceid)
  ↳ + GetSpriteHeight(faceid) / 2.0)
endfunction


//*** Sets the depth of the stopwatch ***
function SetGUIStopwatchDepth(id as integer,
↳d as integer)
  //*** If the id is invalid, exit ***
  if id < 1 or id > GUIStopwatches.length
    exitfunction
  endif
  //*** If stopwatch deleted, exit ***
  if GUIStopwatches[id].deleted = 1
    exitfunction
  endif
  //*** If depth invalid, exit ***
  if d < 2 or d > 9999
    exitfunction
  endif
  //*** Create a set of shorter names ***
  faceid = GUIStopwatches[id].facespr
  ssid = GUIStopwatches[id].ssspr
  resetid = GUIStopwatches[id].resetspr
  handid = GUIStopwatches[id].handspr
  txtid = GUIStopwatches[id].timetxt
  //*** Set the depth of each component ***
  SetSpriteDepth(faceid,d)
  SetSpriteDepth(ssid,GetSpriteDepth(faceid)+1)
  SetSpriteDepth(resetid,GetSpriteDepth(faceid)+1)
  SetTextDepth(txtid,GetSpriteDepth(faceid))
  SetSpriteDepth(handid, GetSpriteDepth(faceid)-1)
endfunction


//**************************************
//***  Stopwatch Helper Functions    ***
//**************************************
//*** Returns a formatted time string (m:ss)***
function ToMinutesSeconds(msecs as integer)
  totalsecs = msecs /1000
  mins = totalsecs / 60
  secs = Mod(totalsecs,60)
  result as string
  result = Str(mins)+":"+Right("0"+Str(secs),2)
endfunction result
```