Learning 2D Game Programming: Basic FrameWork Part 4 - Revised FrameWork

Table of Contents

Introduction

In the previous tutorial we added the enemy to our Game. It wasn't doing anything very much. Before we proceed further with this program I want to introduce two concepts before we move on.

This tutorial is what I really want to get to as it will get us into a situation where we would have completed a journey to a Game Framework which we can then use to create really sophisticated and fun games.

Unfortunately we have to get through this little hump before we can get there. So let us make a start and get this done and over with.

Type Inheritance

If we were to take a look at the two types that we have created; TSpaceShip and TAlienShip types. We can see that they are rather similar in that they have the X,Y location attributes, Image attribute as well as a speed attribute.

I am now going to introduce you to a very interesting and useful concept which allows types to inherit the attributes and behaviours of a parent type.

So if we create a type as below:-

```
Type TGameObject

   Field X:Int = 320
   Field Y:Int = 420
   Field Speed:Int=3
   Field Image:TImage

   Method DrawSelf()
      DrawImage Image,X,Y
   End Method

    Method UpdateState() Abstract

End Type
```

We can then create a child of this TGameObject type as follows:-

```
Type TSpaceShip Extends TGameObject

  Function Create:TSpaceShip(File:String,xstart:Int,ystart:Int)
     Local Ship:TSpaceShip=New TSpaceShip
     Ship.X=xstart
     Ship.Y=ystart
     Ship.Image=LoadImage(LoadBank(File))

    If Ship.Image=Null
       Print "Not able to load image file. Program aborting"
       End
    EndIf

     Return Ship
  End Function

  Method UpdateState()

     If KeyDown(KEY_LEFT)
        X :- Speed
     EndIf
     If KeyDown(KEY_RIGHT)
        X :+ Speed
     EndIf

     If X<0 Then X=0
     If X>620 Then X=620

  End Method

End Type
```

Similarly for the TAlienShip we do the following:-

```
Type TAlienShip

  Function Create:TAlienShip(File:String,xstart:Int,ystart:Int)
     Local Ship:TAlienShip=New TAlienShip
     Ship.X=xstart
     Ship.Y=ystart
     Ship.Image=LoadImage(LoadBank(File))

    If Ship.Image=Null
       Print "Not able to load image file. Program aborting"
       End
    EndIf

     Return Ship
  End Function

  Method UpdateState()
     X :- Speed
     If X<-ImageWidth(Image) Then X=620
  End Method

End Type
```

Notice that for both types we now no longer need to define the X,Y position fields as well as the Image and speed field as these have already been defined in the parent TGameObject Definition earlier. The children types inherited the attributes and behaviours of the parent

```
Type TGameObject

    Field X:Int = 320
    Field Y:Int = 420
    Field Speed:Int=3

    Method DrawSelf()
       DrawImage Image,X,Y
    End Method

     Method UpdateState() Abstract

End Type
```

We can create the DrawSelf() behaviour in the parent but not the UpdateState() as they are different (one is player controlled the other not).

The **Abstract** keyword after the UpdateState() Method declaration means that every child must have a UpdateState. The parent itself has an UpdateState() method which does nothing.

The BlitzMax List

Before we move on, let us introduce ourselves to the BlitzMax Lists. In later chapters we will learn more about lists but for now it's suffice to understand that just like shopping lists, the BlitzMax Lists allow us to keep items in this case, our gameobjects, in a BlitzMax List.

Look at the simple example below:-

```
Local ShoppingList:TList=CreateList()

ListAddLast ShoppingList,"Milk"
ListAddLast ShoppingList,"Eggs"
ListAddLast ShoppingList,"Butter"

For Local s:String=EachIn ShoppingList
   Print s
Next

End
```

Running the above programs yields:-

```
Building untitled5
Compiling:untitled5.bmx
flat assembler  version 1.64
3 passes, 3444 bytes.
```

```
Linking:untitled5.debug.exe
Executing:untitled5.debug.exe
Milk
Eggs
Butter

Process complete
```

In order to use a list we first have to create it using the **CreateList**() function

```
Local ShoppingList:TList=CreateList()
```

Then we add items to it using the **ListAddLast** function

```
ListAddLast ShoppingList,"Milk"
ListAddLast ShoppingList,"Eggs"
ListAddLast ShoppingList,"Butter"
```

We then iterate through the list using the **For/Next ... EachIn** function

```
For s:String=EachIn ShoppingList
   Print s
Next
```

That is all we need to know about lists for the moment.

The GameObject List

Now that we have an idea about lists, we can put our gameobjects into a list which we are calling the GameObjectList and can then simplify our Main Game Loop as follows:-

```
 ' ----------------------MAIN LOOP-------------------------
Repeat
      Cls
      For o:TGameObject=EachIn GameObjectList
            o.DrawSelf()
            o.UpdateState()
       Next
       Flip
Until KeyDown(KEY_ESCAPE) Or AppTerminate()
End
```

If you look at the similarity of the **For/Next...EachIn** loop to the one above, we can see that but adding our GameObjects to the GameObjectList we have a much cleaner Main Loop.

I'm sure you can appreciate that our Main Loop would have been messy if we have more than 3 gameobjects (i.e. 3 DrawSelf and UpdateState methods for each object)

The Revised Game Framework

Based on the new concepts above, our new revised Game Framework now looks like this (the text in

grey are the same as the previous framework):-

```
' -----------------SETUP GAME CONDITIONS--------------------
Global GameObjectList:TList=CreateList()
Graphics 640,480,0

Local URL:String="http::www.2dgamecreators.com/tutorials
/gameprogramming/basic/"
Local Player:TSpaceShip =
TSpaceShip.Create(URL+"/blobship_1-1.png",320,420)
Local Alien:TAlienShip =
TAlienShip.Create(URL+"/cartoonufo_1-1.png",320,0)

 ' --------------------MAIN LOOP-------------------------
Repeat
        Cls
        For o:TGameObject=EachIn GameObjectList
              o.DrawSelf()
              o.UpdateState()
         Next
         Flip
Until KeyDown(KEY_ESCAPE) Or AppTerminate()
End

' ---------------------TYPES, ATTRIBUTES AND BEHAVIOURS-------------------------
Type TGameObject

    Field X:Int = 320
    Field Y:Int = 420
    Field Speed:Int=3
    Field Image:TImage

    Method DrawSelf()
       DrawImage Image,X,Y
    End Method

     Method UpdateState() Abstract

End Type

 Type TSpaceShip Extends TGameObject

    Function Create:TSpaceShip(File:String,xstart:Int,ystart:Int)
        Local Ship:TSpaceShip=New TSpaceShip
        Ship.X=xstart
        Ship.Y=ystart
        Ship.Image=LoadImage(LoadBank(File))

       If Ship.Image=Null
          Print "Not able to load image file. Program aborting"
          End
       EndIf

        ListAddLast GameObjectList, Ship
        Return Ship
    End Function
```

```
    Method UpdateState()

        If KeyDown(KEY_LEFT)
            X :- Speed
        EndIf
        If KeyDown(KEY_RIGHT)
            X :+ Speed
        EndIf

        If X<0 Then X=0
        If X>620 Then X=620

    End Method

End Type


Type TAlienShip Extends TGameObject

    Function Create:TAlienShip(File:String,xstart:Int,ystart:Int)
        Local Ship:TAlienShip=New TAlienShip
        Ship.X=xstart
        Ship.Y=ystart
        Ship.Image=LoadImage(LoadBank(File))

       If Ship.Image=Null
          Print "Not able to load image file. Program aborting"
          End
       EndIf

        ListAddLast GameObjectList, Ship
        Return Ship
    End Function

    Method UpdateState()
        X :- Speed
        If X<-ImageWidth(Image) Then X=620
     End Method

 End Type
-.
```

Running the above code should yield exactly the same behaviour as before. We now have a Game Framework from which our prize winning games can be built from.

Summary

This Framework will form the basis of our 2D game structure. Next we will add some missile shooting capability to our player

Blitzmax commands introduced in this tutorial

Some more new keywords and functions here:-

Keyword **Extends**

Specify user defined Type supertype.

Function **CreateList**:TList()

Create a linked list.

Function **ListAddLast**:TLink(list:TList,value:Object)

Add an object to a linked list.

Keyword **EachIn**

Iterate through an array or collection/list.

Keyword **For**

Marks the start of a loop that uses an iterator to execute a section of code repeatedly.

Keyword **Next**

End a For block.

Back to the index, Previous Tutorial, Next Tutorial