How to write a BreakOut Game: Part 5: Introducing Game States

Table of Contents

Introduction to Game States

If we were to reflect back on what we have done over the last few tutorials, we have programmed in the objects and behaviours required during the actual playing state, the PLAY Game State. When our bricks have all been destroyed, we should then move into a GAMEOVER Game State.

So Game States, as the name implies, represent the state of our game and our objects can behave differently during different Game States.

For the BreakOut game for example, we can have the following Game States

- STARTUP
- WAIT TO START
- PLAY
- PAUSE
- GAMEOVER
- EXIT

The PLAY Game State

Since, without realising it, we have always been in the PLAY GameState, lets take a look at how we would implement that in our revised Game Frame Work which can now handle the different Game States.

Firstly our Main Loop will change slightly to include the GameState handling via the UpdateGameState() function call plus the Loop is now an infinite loop as the Ending of the program is handled in the UpdateGameState() function:-

```
' ----------------------MAIN LOOP-------------------------
  Repeat
    Cls
      UpdateGameState()
      For Local o:TGameObject=EachIn GameObjectList
          o.DrawSelf()
          o.UpdateSelf()
      Next
    Flip
  Forever

End
```

Our UpdateGameState function initially will look like this, ie just handling one GameState which is the PLAY GameState:-

```
Function UpdateGameState()

    Select GameState

    Case PLAY
          If KeyDown(KEY_ESCAPE) Or AppTerminate()
             End
          EndIf
```

```
        EndSelect

    EndFunction
```

We also need to define one new variable, the GameState integer variable and also a constant called PLAY. We do this at the top of our program as follows:-

```
    ' ----------------SETUP GAME CONDITIONS--------------------
    Const PLAYER_LAYER:Int=1
    Const PLAY:Int=1
    Global GameState:Int=PLAY
```

Download the full source here. Our program should behave exactly like before.

Adding the PAUSE GameState

A useful GameState to have is the PAUSE GameState where users can pause the game and go away and do something else like go to the bathroom.

Previously, whenever the ESC key was pressed or the Close Button was clicked, the program would abruptly end. We can be more polite and ask users to confirm when they press ESCAPE rather than just end the program abruptly as we've been doing so far.

In order to do this, we need to display some text (instructions) onto the screen. The framework that we have demands that everything that needs to be displayed on the screen must be descended from TGameObject. So to display text we need a TText object declared like so:-

```
    Type TText Extends TGameObject

      Field Text:String
      Field Font:TImageFont

      Function Create:TText(Text:String, Font:TImageFont,xstart:Int,ystart:Int)
        Local B:TText=New TText
        B.Text=Text
        B.Font=Font
        B.X=XStart
        B.Y=YStart
        ListAddLast GameObjectList, B
        Return B
      End Function

      Method UpdateSelf()
        'DO NOTHING
      End Method

      Method drawself()
        SetImageFont(Font)
        SetRotation 0
        SetScale 1.0,1.0
        DrawText Text,X,Y
      End Method

    EndType
```

Our new UpdateGameState function would look like this:-

```
    Function UpdateGameState()
```

```
    Select GameState

        Case PLAY
            If KeyDown(KEY_ESCAPE) Or AppTerminate()
                TText.Create("PRESS <ENTER> To RE-START",Font,80,250)
                TText.Create("PRESS ESCAPE TO EXIT",Font,80,300)
                FlushKeys()
                GameState=PAUSE
            EndIf

        Case PAUSE

            If KeyDown(KEY_ESCAPE) Or AppTerminate()
              End
            EndIf

            If KeyDown(KEY_ENTER)
              GameState=PLAY
'-------------Remove all of the Text Objects
              For Local o:TText=EachIn GameObjectList
                  ListRemove(GameObjectList,o)
               Next
            EndIf
            FlushKeys()

    EndFunction
```

We also need to load the font to use

```
    Local FontURL:String="http::www.2dgamecreators.com/tutorials/gameprogramming
    /breakout/"
    Global Font:TImageFont=LoadImageFont(FontURL+"BorisBlackBloxx.ttf",24)
```

and declare the new constant PAUSE,

```
    Const PLAY:Int=1
    Const PAUSE:Int=2
```

and modify the UpdateSelf methods of TBall and TPaddle to not do anything if not in PLAY mode as follows:-

```
        Method UpdateSelf()

            If GameState<>PLAY Then Return
```

Download the complete source here, compile and build the program. Press the ESC key or click on the close icon and you should see something like this:-

Summary

We have now enhanced our framework a little bit to allow us to use the concept of GameStates. But as you can see the overall flow of the program has not changed by very much.

With this new ability, we can then further enhance our framework (and games built with it) to include some more GameStates that we are familiar with such as STARTUP, GAMEOVER etc.

The next section of this tutorial series will introduce the concept of Game States. So press on (Click here)