The Wayback Machine - http://web.archive.org/web/20070820042413/http://www.2dgamecreator...
How to write a BreakOut Game: Part 6: A new revised Framework with Game States
(c) Assari 2006

Table of Contents

Introduction

In the previous tutorial we introduced 2 GameStates, PLAY and PAUSE. In this tutorial we will add in three more GameStates

- The START GameState
- The WAIT GameState
- The GAMEOVER GameState

The START and WAIT Game States

Instead of going immediately to the PLAY GameState at the very start of a game, most games actually have a startup stage where we can normally see a splashscreen, cut-scene videos and an option screen.

For our game's START GameState, we will display some sort of introductory screen and then present an option where the user can either start the game or exit the game.

The code for this, looks like this:-

```
Case START
   CreateBricks()
   TText.Create("BREAKOUT",Font72,80,50)
   TText.Create("PRESS <ENTER> To START",Font,110,250)
   TText.Create("PRESS <ESCAPE> TO EXIT",Font,110,300)
   GameState=WAIT
```

Notice that at the end of the START block, we change our state to the WAIT GameState which actually does the waiting. A more verbose GameState for this would be WAITFORUSERTODECIDE :)

Before we get too ahead of ourselves, the alert amongst you would have noticed that the BREAKOUT text is using a diferent font. This we declare at the beginning of our framework as follows:-

```
Local FontURL:String="http::www.2dgamecreators.com/tutorials/gameprogramming
/breakout/"
Global Font:TImageFont=LoadImageFont(FontURL+"BorisBlackBloxx.ttf",24)
Global Font72:TImageFont=LoadImageFont(FontURL+"BorisBlackBloxx.ttf",72)
```

We also need to declare the new constants

```
' ----------------SETUP GAME CONDITIONS--------------------
Const PLAYER_LAYER:Int=1
Const PLAY:Int=1
Const PAUSE:Int=2
Const START:Int=3
Const WAIT:Int=4
```

Finally we need to declare the WAIT case block as follows:-

```
Case WAIT  'To Start
```

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

        If KeyDown(KEY_ENTER)
'-------------Remove all Objects
          ClearList GameObjectList
'--------------Now Create the game objects that we need
           TBall.Create(LoadImage(URL+"ball.png"),Width/2,400)
           TPaddle.Create(LoadImage(URL+"paddle.png"),Width/2,0)
           CreateBricks()
          GameState=PLAY
        EndIf
        FlushKeys()
```

Notice again that the GameState changes just as we exit the WAIT block.
Download the full source here. We should see the following:-

Adding the GAMEOVER GameState

The GameOver GameState is actually very simple. The GAMEOVER Block will look like this:-:-

```
    Case GAMEOVER
      TText.Create("GAMEOVER",Font72,60,50)
      TText.Create("PRESS <ENTER> TO RE-START",Font,80,250)
      TText.Create("PRESS <ESCAPE> TO EXIT",Font,110,300)
      FlushKeys()
      GameState=WAIT
```

All we are doing is displaying the GAMEOVER notice and then asking the user whether he/she wants to play another game or not. Then we go into a WAIT GameState to wait on his/her input.

The difficult bit is how do we decide that we have reached the GAMEOVER state from the PLAY state. For our game, GAMEOVER is when all the bricks are destroyed.

There are several ways of doing this. I have decided for clarity and to also show a diferent inheritance style to create a TScore Object inherited from the TText Object as follows:-

```
Type TScore Extends TText
```

```
    Field score:Int

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

    Method UpdateSelf()

       If GameState<>PLAY Then Return    'Do not continue if not in PLAY state

       score=0
       For Local B:TBricks=EachIn GameObjectList
          score :+ 1
       Next

       If score<90 Then GameState=GAMEOVER
    End Method

    Method drawself()
       SetImageFont(Font)
       SetRotation 0
       SetScale 0.75,0.75
       SetColor 255,0,0
       DrawText Text+Score,X,Y
    End Method

EndType
```

As you can see, the TScore object counts the number of bricks in the GameObjectList (see the UpdateSelf Method) and then displays this number on the graphic screen. Notice how I use the **SetScale** function to display a smaller font then the standard 36 point font that we were using before.

Now, doing the counting of the bricks every frame is not very efficient, a far better way is to either count (or reduce the score) at each collision but I will leave that as an exercise for you to do. (hint:use a global variable)

Note that I change from PLAY to GAMEOVER at score<90 rather than wait for every bricks to be destroyed (takes too long to wait). Change this to zero in a real game :)

```
    If score<90 Then GameState=GAMEOVER
```

Another thing to note is that I have used the SetColor function to display the text in red. Having done that, I had to upgrade all the other DrawSelf method to ensure that the SetColor for them is back to white. It is important that each object does not assume that state of the Color, Rotation and Scale of the graphic display. If you've noticed I have reset these values to what the current object need ie SetRotation to zero, scale to 0.75 etc

```
    Method drawself()
       SetImageFont(Font)
       SetRotation 0
       SetScale 0.75,0.75
       SetColor 255,0,0
       DrawText Text+Score,X,Y
```

```
        End Method
```

Before this program can be completed, we now need to create an instance of our TScore Object just before we start the game, in the WAIT block:-

```
            Case WAIT  'To Start

               If KeyDown(KEY_ESCAPE) Or AppTerminate()
                  End
               EndIf

               If KeyDown(KEY_ENTER)

'-------------Remove all Objects
                 ClearList GameObjectList
'--------------Now Create the game objects that we need
                  TBall.Create(LoadImage(URL+"ball.png"),Width/2,400)
                  TPaddle.Create(LoadImage(URL+"paddle.png"),Width/2,0)
                  TScore.CREATE("SCORE:",Font,10,5)
                  CreateBricks()
                 GameState=PLAY
               EndIf
               FlushKeys()
```

OK. We actually are done. Download this source code here and build and run the program. We should be able to see the score at the top let corner of the screen

Summary

There are other GameStates such as NEXTLEVEL, OPTIONS etc but we will leave that for later game tutorials (or you can experiment for yourself). Please feel free to experiment with these and other gamestates.

What I hope I have been able to do is to show you a possible game framework where the main elements of a game are there and to show that once a framework such as this has been developed, with the power of BlitzMax Object functionality, creating a game simply means adding the required gameobjects - their attributes and behaviours and how they relate to one another

Good Luck with your programming and don't forget to have fun :)

Back to Index.