How to write a BreakOut Game: Part 2: The TBall Object

Table of Contents

The TBall Object

Having the Game Framework already established allows us to concentrate on adding one object at a time to our game.

The first object we are going to add is the TBall object whose main characteristic is to ability to bounce off the walls of the graphic screen.

The declaration of the the TBall Object looks like this:-

```
Type TBall Extends TGameObject

   Function Create:TBall(Image:TImage,xstart:Int,ystart:Int)
      Local B:TBall=New TBall
      CreateObject(B,Image,xstart,ystart)
      Return B
   End Function

   Method UpdateSelf()

      X :+ XSpeed  'Ball moves by its speed in the X direction
      Y :+ YSpeed  'Ball moves by its speed in the Y direction

      If x>Width Or x<0 Then 'Collision with left or right boundary
        XSpeed=-XSpeed         'simply reverses direction
      EndIf
      If Y>Height Or Y<0 Then  'Collision with left or right boundary
        YSpeed=-YSpeed          'simply reverses direction
      EndIf

  End Method

End Type
```

Notice the UpdateSelf Method where two things happens:-

- The TBall Object moves by its XSpeed and YSpeed rate each loop
- The position of the TBall object is checked against the boundary of the graphic screen and the XSpeed and YSpeed is changed accordingly to simulate the bounce

Our complete program (for the moment) then are as follows:-

```
Strict
' ----------------SETUP GAME CONDITIONS--------------------

Global GameObjectList:TList=CreateList()
Global Width:Int=640
Global Height:Int=480
Global URL:String=BlitzMaxPath()+"/samples/Breakout/media/"
```

```
Graphics Width,Height
AutoMidHandle True
HideMouse()

TBall.Create(LoadImage(URL+"ball.png"),Width/2,400)

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

End
'============================================================

' ---------------THE MOTHER OF ALL OBJECT, TYPE GAMEOBJECT-------------
Type TGameObject

    Field X:Int
    Field Y:Int
    Field XSpeed:Float=3
    Field YSpeed:Float=-3
    Field Image:TImage
    Field XScale:Float=1.0
    Field YScale:Float=1.0

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

     Method UpdateSelf() Abstract

End Type

' ----------------------DEFINE REQUIRED GAME OBJECTS-------------------------
Type TBall Extends TGameObject

    Function Create:TBall(Image:TImage,xstart:Int,ystart:Int)
        Local B:TBall=New TBall
        CreateObject(B,Image,xstart,ystart)
        Return B
    End Function

    Method UpdateSelf()

        X :+ XSpeed  'Ball moves by its speed in the X direction
        Y :+ YSpeed  'Ball moves by its speed in the Y direction

        If x>Width Or x<0 Then 'Collision with left or right boundary
           XSpeed=-XSpeed          'simply reverses direction
        EndIf
        If Y>Height Or Y<0 Then  'Collision with left or right boundary
           YSpeed=-YSpeed           'simply reverses direction
        EndIf

   End Method
```

```
End Type


'------------------------------HELPER FUNCTIONS-------------------------
Function CreateObject(Obj:TGameObject, Image:TImage,xstart:Int,ystart:Int,Scale:Float=1.0)

        Obj.X=xstart
        Obj.Y=ystart
        Obj.XScale=Scale
        Obj.YScale=Scale
        Obj.Image=Image

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

        ListAddLast GameObjectList, Obj

End Function
```

Running the above code will give us a bouncing ball. Note that we are using the ball image that came with the BlitzMax sample BreakOut program.

Adding Rotational Behaviour and Scale to the TBall Object

To enhance our ball a little bit lets make it bigger (increase the size by twice) and also allow it to rotate.

To add the Rotation behaviour, we need to do 3 things

- Add the Rotation Attribute to TGameObject
- Modify the DrawSelf method to set the rotation prior to drawing the image
- And lastly, to update the rotation attribute in the TBall UpdateSelf Method.

```
' ----------------THE MOTHER OF ALL OBJECT, TYPE GAMEOBJECT-------------
Type TGameObject
```

```
    Field X:Int
    Field Y:Int
    Field XSpeed:Float=3
    Field YSpeed:Float=-3
    Field Image:TImage
    Field XScale:Float=1.0
    Field YScale:Float=1.0
    Field Rotation:int=0

    Method DrawSelf()
        SetScale XScale, YScale
        SetRotation Rotation
        DrawImage Image,X,Y
    End Method

     Method UpdateSelf() Abstract

End Type
```

We modify the TBall UpdateSelf Method by adding 10 rotations per frame and resetting it back to zero once it has completed a full turn (360 degrees) as follows:-

```
' ---------------------DEFINE REQUIRED GAME OBJECTS-------------------------
Type TBall Extends TGameObject

    Function Create:TBall(Image:TImage,xstart:Int,ystart:Int)
        Local B:TBall=New TBall
        CreateObject(B,Image,xstart,ystart,2.0)
        Return B
    End Function

    Method UpdateSelf()

        X :+ XSpeed
        Y :+ YSpeed

        If x>Width Or x<0 Then
          XSpeed=-XSpeed
        EndIf
        If Y>Height Or Y<0 Then
          YSpeed=-YSpeed
        EndIf

        Rotation :+ 10
        If Rotation >= 360 Then Rotation=0

   End Method

End Type
```

Note that to increase the size of the TBall object is fairly straightforward as we already have the XScale, YScale attribute in TGameObject from before and so we simply need to create the object with a scale of 2.0 instead of 1.0. Study the helper function CreateObject to understand better how this works

```
Type TBall Extends TGameObject

    Function Create:TBall(Image:TImage,xstart:Int,ystart:Int)
        Local B:TBall=New TBall
```

```
        CreateObject(B,Image,xstart,ystart,2.0)
        Return B
    End Function
```

Download the source here and then run the program. You should see a bigger ball spinning around the graphic screen.

Summary

So this again is very straightforward due to the strength of the Blitzmax object construct. With a good framework in place, creating a game is just a matter of adding the desired game objects into the program

Next we will add the player controlled paddle.

Return to Index, Goto Next Tutorial