

## How to write a BreakOut Game: Part 3: The TPaddle Object

(c) Assari 2006

### Table of Contents

1. [The TPaddle Object](#)
2. [Adding Collision between Paddle and Ball](#)
3. [Summary](#)

## The TPaddle Object

The next thing we are going to add is the player controlled paddle.  
The declaration of the the TPaddle Object looks like this:-

Type TPaddle Extends TGameObject

Function Create:TPaddle(Image:TImage,xstart:Int,ystart:Int)

Local B:TPaddle=New TPaddle

CreateObject(B,Image,xstart,ystart)

B.XSpeed :\* 1.5 'need a slightly faster speed than the balls

B.YSpeed :\* 1.5 'otherwise we cannot catch them

Return B

End Function

Method UpdateSelf()

'-----Move X direction depending on which key was pressed

If KeyDown(KEY\_LEFT) X :- XSpeed

If KeyDown(KEY\_RIGHT) X :+ XSpeed

Y = height-60 'Y remains the same

'-----Make sure paddle cannot move beyond the gameplay area

If X<ImageWidth(Image)/2 X=ImageWidth(Image)/2

If X>(Width-ImageWidth(Image)/2) X=(Width-ImageWidth(Image)/2)

End Method

End Type

Just like the TBall object the UpdateSelf Method has 2 main behaviours:-

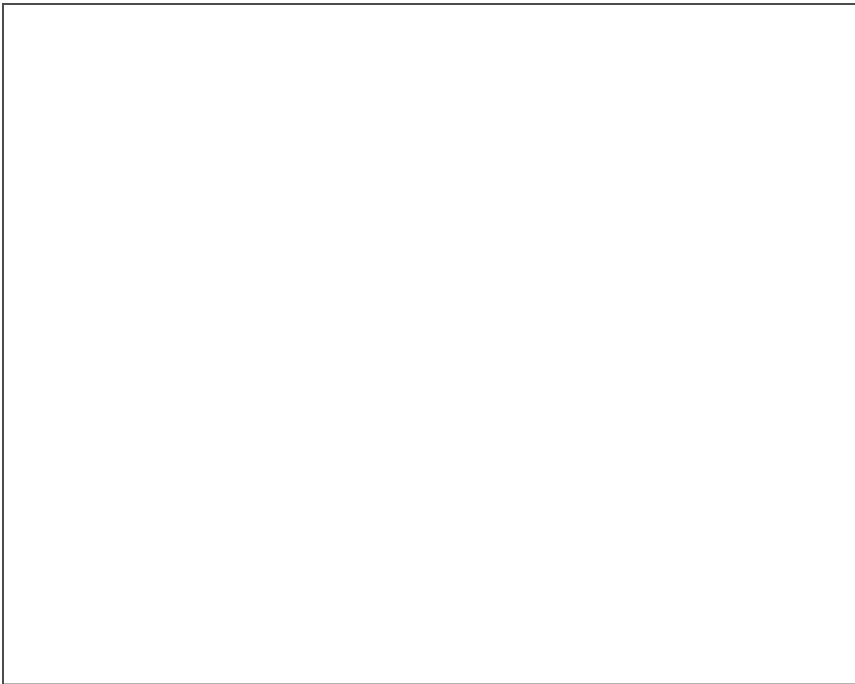
- The TPaddle Object moves in the direction of user key input (left or right cursor)
- The position of the TPaddle object is checked against the left & right edge to ensure it is always within the play area.

We also need to create an instance of our player:-

TBall.Create(LoadImage(URL+"ball.png"),Width/2,400) 'Creates our  
Bouncing Ball

TPaddle.Create(LoadImage(URL+"paddle.png"),Width/2,0)

Running the code ([link](#)) will give us a paddle which can be controlled by pressing the left and right cursor keys. As we have not implemented any collision, the ball is simply passing through the paddle for now.



## Adding Collision between Paddle and Ball

We are going to use the built-in Max2D collision function **CollideImage()** to detect the collision between the ball and the paddle. If you want to learn more about how to use collisions in BlitzMax, head over to my Collision Tutorials ([link here](#)). To recap, the collision detection requires a 3 step process

- First reset the collision layer to clear previous entries
- Write the image to the collision layer
- Read the Image from the collision later to check for collision

The reset and write operations is done within the TPaddle UpdateSelf Method as follows:-

Type TPaddle Extends TGameObject

```
Function Create:TPaddle(Image:TImage,xstart:Int,ystart:Int)
    Local B:TPaddle=New TPaddle
    CreateObject(B,Image,xstart,ystart)
    B.XSpeed :* 1.5 'need a slightly faster speed than the balls
    B.YSpeed :* 1.5 'otherwise we cannot catch them
    Return B
End Function
```

Method UpdateSelf()

```
'-----Move X direction depending on which key was pressed
    If KeyDown(KEY_LEFT) X :- XSpeed
    If KeyDown(KEY_RIGHT) X :+ XSpeed
    Y = height-60 'Y remains the same

'-----Make sure paddle cannot move beyond the gameplay area
    If X<ImageWidth(Image)/2 X=ImageWidth(Image)/2
    If X>(Width-ImageWidth(Image)/2) X=(Width-ImageWidth(Image)/2)

'-----Reset Collision Layer and Write image to collision layer
    ResetCollisions(PLAYER_LAYER)
    CollideImage(Image,X,Y,0,0,PLAYER_LAYER,Self)
```

End Method

End Type

We also need to make sure to define the PLAYER\_LAYER as a constant at the start of the program:-

```
Strict
'-----SETUP GAME CONDITIONS-----
Const PLAYER_LAYER:Int=1
```

And finally we need to perform the collision checking and response (bouncing off the paddle) in the TBall UpdateSelf Method:-

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()

```
    If GameState<>PLAY Return
```

```
    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
```

```
    CheckCollision()
```

End Method

```
Method CheckCollision()
'-----Set the correct scale and rotation before collision checking
SetScale XScale, YScale
SetRotation Rotation
If CollideImage(Image,X,Y,0,PLAYER_LAYER,0)
    YSpeed=-YSpeed
EndIf

End Method
```

End Type

Download the source [here](#) and then build and run the program. You should now be able to bounce the ball off the paddle.

## Summary

In this tutorial, we've added the second object, the player controlled paddle. A simple collision check produces

the necessary bounce behaviour off the paddle.

Next we will be adding the bricks to complete the first phase of this game.

Goto [Previous](#) Tutorial, [Next](#) Tutorial