Learning 2D Game Programming: Max 2D Collision Part 1

Part 1, Part 2, Part 3

Table of Contents

Introduction

Having a game where nothing happens when different moving objects simple passes over each other is no fun. Even a simple action game such as Pong requires that some form of collision mechanism be present.

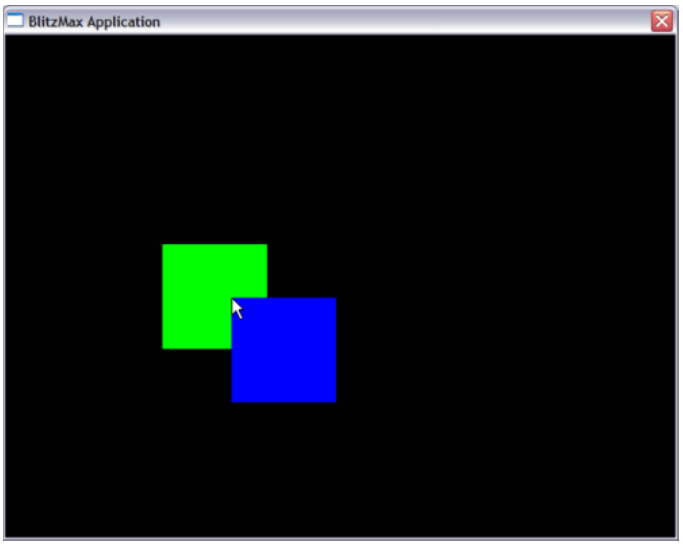
A simple way to look at a collision is that it is nothing more than when two objects overlap each other. Let us take a look at the simplest form of collision, when two rectangular object overlaps each other.

Rectangle Overlap

The simplest form of collision detection is the rectangle overlap.

Let us see an example on how this works:-

```
Graphics 640,480
While Not (KeyHit(key_escape) Or AppTerminate())
  Cls
  SetColor 0,255,0
  DrawRect 150,200,100,100
  SetColor 0,0,255
  DrawRect MouseX(),MouseY(),100,100
  Flip
Wend
End
```



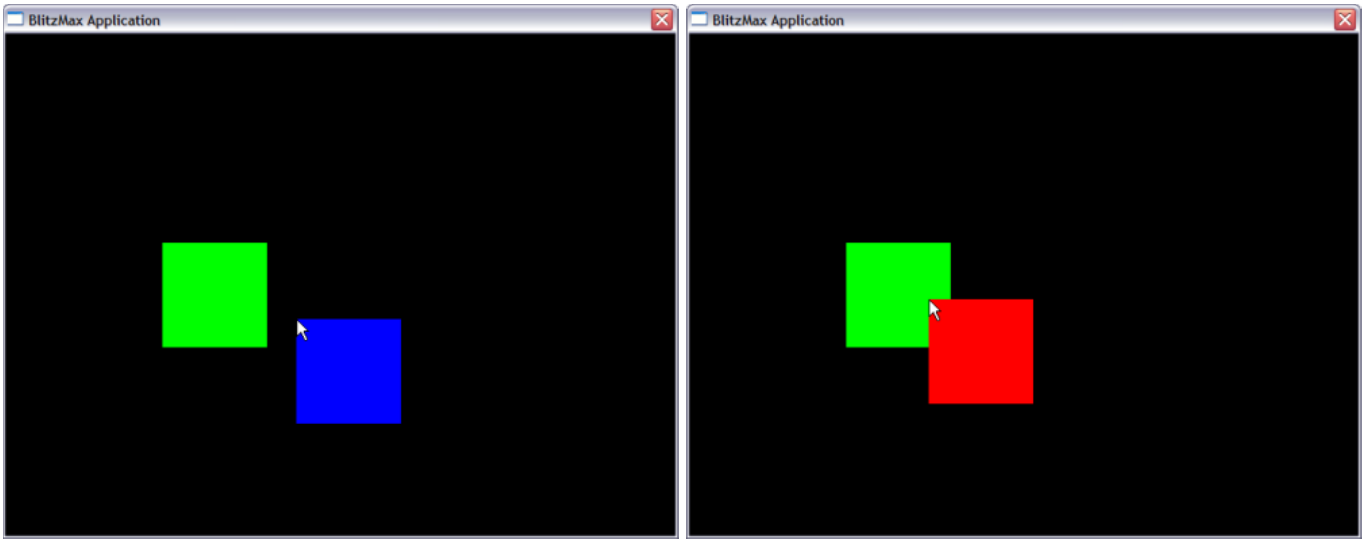
As you can see from running the above program as there are no collision detection in the code nothing happens when the two rectangle overlaps each other.

Let us now add a collision checking function called RectsOverlap (Rectangle Overlaps) and we change the color of the moving rectangle when an overlap happens.

```
Graphics 640,480
While Not (KeyHit(key_escape) Or AppTerminate())
  Cls
  SetColor 0,255,0
  DrawRect 150,200,100,100
  If RectsOverlap(150,200,100,100,MouseX(),MouseY(),100,100)
```

```
    SetColor 255,0,0
  Else
    SetColor 0,0,255
  EndIf
  DrawRect MouseX(),MouseY(),100,100
  Flip
Wend
End

Function RectsOverlap:Int(x0, y0, w0, h0, x2, y2, w2, h2)
   If x0 > (x2 + w2) Or (x0 + w0) < x2 Then Return False
   If y0 > (y2 + h2) Or (y0 + h0) < y2 Then Return False
   Return True
End Function
```



Now we can see the moving rectangle turning red whenver an overlap (collision) happens. So there we have it, our first working collision function :)

Circle Rectangle Overlap

Now if we were to modify our program a little bit,by having a moving oval instead of a rectangle:-

```
Graphics 640,480
While Not (KeyHit(key_escape) Or AppTerminate())
  Cls
  SetColor 0,255,0
  DrawRect 150,200,100,100
  If RectsOverlap(150,200,100,100,MouseX(),MouseY(),100,100)
     SetColor 255,0,0
  Else
     SetColor 0,0,255
  EndIf
  DrawOval MouseX()-50,MouseY()-50,100,100
  Flip
Wend
End

Function RectsOverlap:Int(x0, y0, w0, h0, x2, y2, w2, h2)
   If x0 > (x2 + w2) Or (x0 + w0) < x2 Then Return False
   If y0 > (y2 + h2) Or (y0 + h0) < y2 Then Return False
   Return True
End Function
```

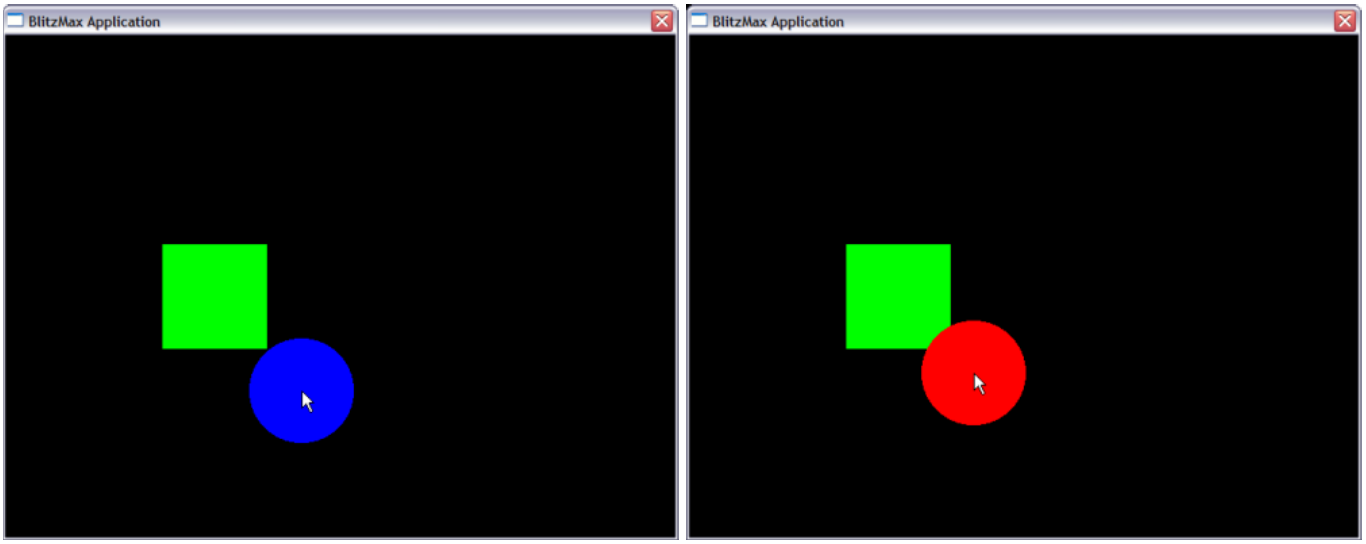
Our collision checking does not work properly anymore. We now need another collision function, specifically for checking a circle versus rectangle overlap.

```
Graphics 640,480
While Not (KeyHit(key_escape) Or AppTerminate())
  Cls
  SetColor 0,255,0
  DrawRect 150,200,100,100
  If CircRectsOverlap(150,200,100,100,MouseX(),MouseY(),50)
     SetColor 255,0,0
  Else
     SetColor 0,0,255
  EndIf
```

```
  DrawOval MouseX()-50,MouseY()-50,100,100
  Flip
Wend
End

Function CircRectsOverlap(x0, y0, w0, h0, cx, cy, r)
   testX=cX
   testY=cY
   If TestX < x0 Then TestX=x0
   If TestX > (x0+w0) Then TestX=(x0+w0)
   If TestY < y0 Then Testy=y0
   If TestY > (y0+h0) Then Testy=(y0+h0)

   Return ((cX-TestX)*(cX-TestX)+(cY-TestY)*(cY-TestY))<r*r
End Function
```



BlitzMax Collision Functions

If we need to have a different collision function for each different pairs of objects that we have on the screen then we could be in serious trouble. You can also see that once we deviate from the simple rectangular overlap the maths get hairier as well.

Fortunately for us BlitzMax (specifically Max2D) provides several collision functions to simplify things for us. Let us take a look at how we would have coded the above collision detection using the builtin Max2D collision function.

ImagesCollide

The simplest collision function provided by BlitzMax is the ImagesCollide function. It has the following syntax

```
Function ImagesCollide(image1:TImage,x1,y1,frame1,image2:TImage,x2,y2,frame2)
```

Basically what the function requires is for us to give the two images we want to check for collision with their respective positions on the screen. We can ignore the frame parameter for the moment by putting a zero.

```
ImagesCollide(Alien,150,200,0,Player,MouseX(),MouseY(),0)
```

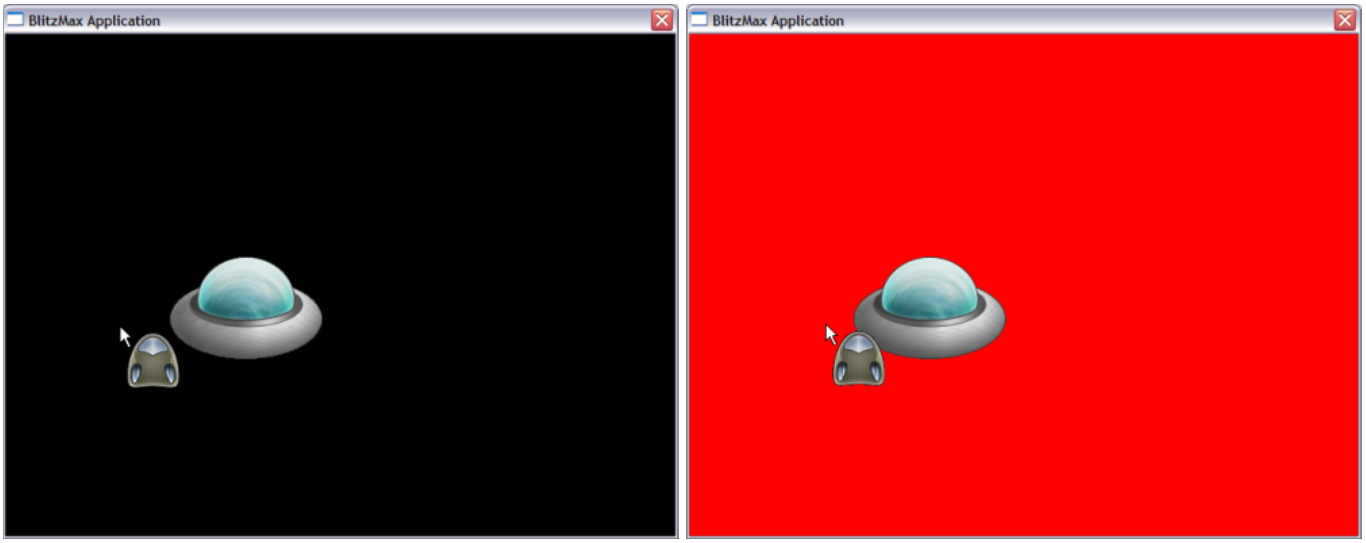
So in our usage above, our Alien is at location 150,200 and our Player is at where the mouse is.

Build and run the program below and see the collision function in action.

```
Graphics 640,480
Local URL:String="http::www.2dgamecreators.com/tutorials
/gameprogramming/basic/"
Local Player:TImage=LoadImage(LoadBank(URL+"blobship_1-1.png"))
Local Alien:TImage=LoadImage(LoadBank(URL+"cartoonufo_1-1.png"))

While Not (KeyHit(key_escape) Or AppTerminate())
  Cls
  DrawImage Alien,150,200
  DrawImage Player,MouseX(),MouseY()
  If ImagesCollide(Alien,150,200,0,Player,MouseX(),MouseY(),0)
     SetClsColor 255,0,0
  Else
     SetClsColor 0,0,0
```

```
  EndIf
  Flip
Wend
End
```



As you can see, using this function is pretty straightforward.

ImagesCollide with Animation

Now let us take a look at the Frame parameter which was skipped earlier. If you build and run the program below, you will see an explosion animation which collides properly with our spaceship.

```
Graphics 640,480
Local URL:String="http::www.2dgamecreators.com/tutorials
/gameprogramming/basic/"
Local Player:TImage=LoadImage(LoadBank(URL+"blobship_1-1.png"))
Local
Alien:TImage=LoadAnimImage(LoadBank(URL+"exp1.png"),64,64,0,16)
Local Frame:Int=0
Local AnimDelay:int=10

While Not (KeyHit(key_escape) Or AppTerminate())
  Cls
  DrawImage Alien,150,200, Frame
  DrawImage Player,MouseX(),MouseY()
  If ImagesCollide(Alien,150,200,Frame,Player,MouseX(),MouseY(),0)
     SetClsColor 255,0,0
  Else
     SetClsColor 0,0,0
  EndIf
  Flip
  If AnimDelay<0 Then
     Frame :+ 1
     If Frame>15 Then Frame=0
     AnimDelay=10
  EndIf
  AnimDelay :- 1


Wend
End
```

To use the ImagesCollide function with animated images, we just need to draw the images with their frame count and then use the same frame count in the ImagesCollide parameter. The AnimDelay and Frame incrementing part of the code is required for proper animation.

ImagesCollide with Scaling

Blitzmax allows for real time scaling of the images that we draw on-screen using the SetScale function which has the following syntax:-

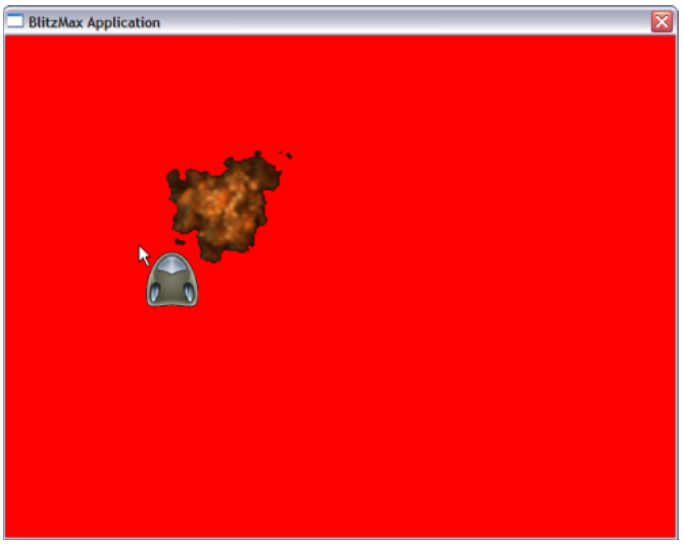
```
Function SetScale( scale_x#,scale_y# )
```

Unfortunately, the ImagesCollide function does not work properly when scaling is involved. Try out the following code and observe how the collision is not working properly anymore.

```
Graphics 640,480
Local URL:String="http::www.2dgamecreators.com/tutorials
/gameprogramming/basic/"
Local Player:TImage=LoadImage(LoadBank(URL+"blobship_1-1.png"))
Local
Alien:TImage=LoadAnimImage(LoadBank(URL+"exp1.png"),64,64,0,16)
Local Frame:Int=0
Local AnimDelay:int=10
Local PlayerSize:Int=1
Local AlienSize:Int=2

While Not (KeyHit(key_escape) Or AppTerminate())
  Cls
  SetScale AlienSize,AlienSize
  DrawImage Alien,150,100,Frame
  SetScale PlayerSize,PlayerSize
  DrawImage Player,MouseX(),MouseY()
  If ImagesCollide(Alien,150,100,Frame,Player,MouseX(),MouseY(),0)
     SetClsColor 255,0,0
  Else
     SetClsColor 0,0,0
  EndIf
  Flip
  If AnimDelay<0 Then
    Frame :+ 1
    If Frame>15 Then Frame=0
    AnimDelay=10
  EndIf
  AnimDelay :- 1

Wend
End
```



Fortunately, BlitzMax has a variant of the **ImagesCollide** function called, imaginatively, **ImagesCollide2** which has the following syntax:-

```
Function
ImagesCollide2(image1:TImage,x1,y1,frame1,rot1#,scalex1#,scaley1#,image2:TImage,x2,y2,frame2,rot2#,scalex2#,scaley2#)
```

Note that is complicated function allows for rotation and scale factors to be part of its parameters. So all we need to do is used the ImagesCollide2 function as follows:-

```
If
ImagesCollide2(Alien,150,100,Frame,0,AlienSize,AlienSize,Player,MouseX(),MouseY(),0,0,PlayerSize,PlayerSize)
```

So go ahead and replace the ImagesCollide function with the above ImagesCollide2 function and see the collision working properly.

ImagesCollide with Rotation

I'll leave it as an exercise for you to figure out how the rotation functionality should be used :)

Summary

In this tutorial we saw two simple (but fast) way to check for collision using the Rectangle Overlaps and Circle Rectangle Overlaps functions. But these functions work only for rectangles and circles.

As our games typically have images instead, we need far more sophisticated functions. Luckily for us BlitzMax provides these pixel-perfect collision functions called ImagesCollide and ImagesCollide2.

In the next tutorial, we are going to see how to use a more sophisticated collision system provided by BlitzMax.

Back to the index, Next Tutorial