Learning 2D Game Programming: Max 2D Collision Part 3

(c) Assari 2006

Part 1, Part 2, Part 3

Table of Contents

Introduction

In the previous tutorial we saw the how to check for collision using the CollideImage function. The CollideImage function performs the collision checking in a pixel perfect manner. We can easily check for this by slowly moving our player towards the spaceship and noting exactly when the color changes.

But as to be expected, these pixel perfect checking takes its toll on computing power and done excessively will slow your game down.

BlitzMax provides another function similar to CollideImage called CollideRect. Let us see how this works.

CollideRect

The syntax for CollideRect is similar to CollideImage

```
Function CollideRect:Object[](x,y,w,h,collidemask%,writemask%,id:Object=Null)
```

Let us re-write our previous example using the CollideRect function.

```
Strict
Graphics 640,480
AutoMidHandle True

Local URL:String="http::www.2dgamecreators.com/tutorials
/gameprogramming/basic/"
Local SpaceShip:TImage=LoadImage(LoadBank(URL+"blobship_1-1.png"))
Local AlienShip1:TImage=LoadImage(LoadBank(URL+"cartoonufo_1-1.png"))
Local AlienShip2:TImage=LoadImage(LoadBank(URL+"cartoonufo_1-1.png"))

Local w:int=ImageWidth(SpaceShip)
Local h:int=ImageHeight(SpaceShip)

Repeat
 Cls
 Local R:Int=0
 Local G:Int=0
 Local B:Int=255
 ResetCollisions()
 SetColor 255,255,255
 DrawImage AlienShip1, 250,100
 CollideImage(AlienShip1,250,100,0,0,1,AlienShip1)

 DrawImage AlienShip2, 400,100
 CollideImage(AlienShip2,400,100,0,0,1,AlienShip2)
 Local p:Object[]=CollideRect(MouseX()-w,MouseY()-h,100,100,1,0)
 For Local i:TImage=EachIn p
   Select i
   Case AlienShip1
      R=255
   Case AlienShip2
      G=255
   End Select
 Next
 SetColor R,G,B
 DrawRect  MouseX()-w,MouseY()-h,100,100
 DrawImage SpaceShip,MouseX(),MouseY()
 Flip

Until KeyDown(KEY_ESCAPE) Or AppTerminate()
```

Notice how the collision is detected as soon as the rectangle touches the AlienShip with the SpaceShip no where near the AlienShip. Try running the above program without the drawrect statement, the collision behaviour will remain the same as collision checking is done by the CollideRect function.

Now the cool thing about the CollideRect function is because it returns the object(s) overlapped by the rectangle, you can then re-checked these objects for pixel perfect collision. The code will look something like this:-

```
Local p:Object[]=CollideRect(MouseX()-w,MouseY()-h,100,100,1,0)
 For Local i:TImage=EachIn p
   If ImagesCollide(i,150,200,0,SpaceShip,MouseX(),MouseY(),0)
        SetClsColor 255,0,0
    Else
        SetClsColor 0,0,0
    EndIf
 Next
```

AutoMidHandle

Note that the starting coordinates of the drawing image is very important for proper collision. The x and y locations and their midhandle used in the CollideImage/CollideRect and the DrawImage functions must be the same. Note that I realign the MouseX(), MouseY() positions with the width and height of the image to ensure that the collision rectangle are within the proper proximity of the image.

Using CollideRect For Mouse Click Selection

This is a short code on how you can use CollideRect to determine selection. Don't forget to 'write' the images to be picked by the cursor into the collision layer.

```
If CollideRect(MouseX(),MouseY(),2,2,1,0) And MouseDown(1)
   <Do your stuff here>
EndIf
```

For a more sophisticated version, you can use the returned object to further act on.

```
Local p:Object[]=CollideRect(MouseX(),MouseY(),2,2,1,0) And MouseDown(1)
 For Local i:TImage=EachIn p
   Select i
    Case Alien
       <Do your stuff here>
    Case Player
        <Do your stuff here>
    EndSelect
 Next
```

Summary

The third tutorial introduces you to the CollideRect function which provides for a slightly quicker way to check for collision. You need to experiment for yourself whether this technique is required or not. Just using CollideImage is obviously simpler.

OK. Thats it. There's some more stuff to read if you are eager to learn more :)
If you wish to read more on collision techniques (not specific to BlitzMax), check out the further reading lists below.

Links to More BlitzMax Collision Codes

Collection of Collision functions by Oddball in Code Archives
Special thanks to Tom for the code example that gave the aha experience. (link)
Code to demonstrate using CollideImage with many objects (link)
Code to demonstrate using CollideImage returning identifying objects (link)

Further Reading

http://vband3d.tripod.com/visualbasic/tut_mixedcollisions.htm - Tutorial on Collisions Between Mixed Bounding Types – Circles versus Rectangles and Triangles
http://www.gamasutra.com/features/19991018/Gomez_4.htm - An Implementation of Arvo's Algorithm
http://www.harveycartel.org/metanet/tutorials/tutorialA.html - Collision Tutorial
http://imrtechnology.ngemu.com/downloads/tutorial.pdf - 2D Collision Tutorial
http://www.yov408.com/html/tutorials.php?&s=54 - List of Collision Tutorials
http://www.flipcode.com/articles/article_basiccollisions.shtml - Basic Collision Tuorial
http://www.edenwaith.com/products/pige/tutorials/collision.php

Back to the index, Previous Tutorial