Learning 2D Game Programming: Basic Game Framework Part 1 - Creating the Player Object

Table of Contents

Introduction

There are several things that you will always see in a BlitzMax game program (or any game program for that matter).

- Instructions to set-up the conditions for the game
- A Main Loop
- A way for the player to control his player object
- Instructions to update the states of the player, the enemies, the bullets etc
- Instructions to display the game objects onto the graphic screen
- A way to end the program

Without further ado, let us get's started on these items so that we can quickly get to the fun stuff. We need to introduce some more basic concepts along the way though :(

Introduction to User Defined Types

Any game must have a player controlled object. In the game we are going to create, it will be a spaceship. BlitzMax allows us to create a player object using a construct called User Defined Types.

In the context of our simple game, we know that our spaceship will need to be displayed on a graphic screen. We saw from the previous tutorial that the BlitzMax drawing functions requires x,y parameters to know where exactly objects are to be displayed on the screen.

So minimally, our spaceship object will require attributes of display locations (also known as coordinates). Let us define our spaceship Object as follows:-

```
Type TSpaceShip
   Field X:Int
   Field Y:Int
End Type
```

Types must always be defined within a **Type ... End** Type block and must have a name (in the example above we have chosen to call our type TSpaceShip). Note that the T in front of the SpaceShip is just good practice. Types can can start with any alphabet.

```
Type TSpaceShip
   Field x:Int
   Field y:Int
End Type
```

So in the example above, our Type is called TSpaceShip and it has two attributes called x and y representing the SpaceShip's display location on our graphic screen.

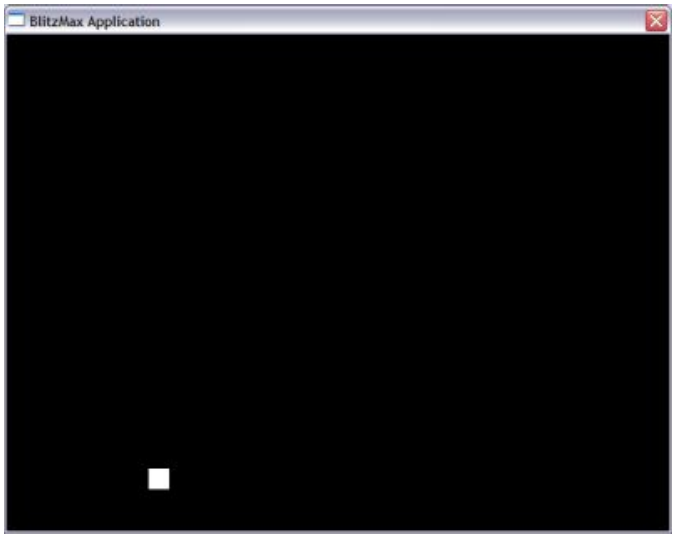
```
Type TSpaceShip
   Field X:Int
   Field Y:Int
End Type
```

Note that I have defined the variables X and Y with :**Int** attached to them. This indicate that we are defining an **Int**eger variable (ie a variable that will hold a number)

Now we can combine what we have learnt from the previous Tutorial with this new concept of Types and start moving our spaceship around the screen.

```
Graphics 640,480,0

Type TSpaceShip
   Field X:Int
   Field Y:Int
End Type

SpaceShip:TSpaceship=New TSpaceShip
HideMouse()

Repeat
   Cls
   SpaceShip.X=MouseX()
   spaceship.Y=420
   DrawRect spaceship.X, spaceship.Y,20,20
   Flip
Until MouseHit(1)
End
```

compile and run the program by clicking the "Build and Run" button as per normal. You should now be able to move the 20x20 white block (representing our Spaceship) along the horizontal section of the screen:-



Let us try to understand what we have done. The instruction **SpaceShip:TSpaceShip=New TSpaceShip** creates an instance called **SpaceShip** from the **TSpaceShip** Type.

You can imagine the **TSpaceShip** Type as a blueprint from which SpaceShips can be created from (or think of TSpaceShip as a factory to make SpaceShip objects).

```
Graphics 640,480,0

Type TSpaceShip
   Field X:Int
   Field Y:Int
End Type

SpaceShip:TSpaceship=New TSpaceShip
HideMouse()

Repeat
```

```
    Cls
    SpaceShip.X=MouseX()
    spaceship.Y=420
    DrawRect spaceship.X, spaceship.Y,20,20
    Flip
Until MouseHit(1)
End
```

the next instruction, HideMouse() simply hides the normal mouse pointer so that it will not clutter our game. You can try deleting this line and see what happens when you build and run the subsequently changed program.

```
    HideMouse()
```

We have seen the **Repeat, Cls, Flip** and **Until MouseHit(1)** instructions in the previous tutorial..

The X position of the SpaceShip is assigned from the X position of the Mouse. Notice how the location X of SpaceShip is written as **SpaceShip.X** (not TSpaceShip.X as we want to update the location of our created SpaceShip, not the one in the factory)

In a similar vein, location Y of SpaceShip is assigned a constant number 420 via the **SpaceShip.Y = 420** instruction. The 20x20 square is then drawn using the **DrawRect** function using the SpaceShip X and Y position.

```
Repeat
    Cls
    SpaceShip.X=MouseX()
    SpaceShip.Y=420
    DrawRect SpaceShip.X, SpaceShip.Y,20,20
    Flip
Until MouseHit(1)
End
```

Types and Methods

We saw in the previous section how Types allow us to create blueprints for our game objects. In the previous section we created attributes of positions in our TSpaceShip type using the **Field** keyword. Another very powerful feature of Types is the ability to allow behaviours of Types to be programmed as part of the blueprint as well.

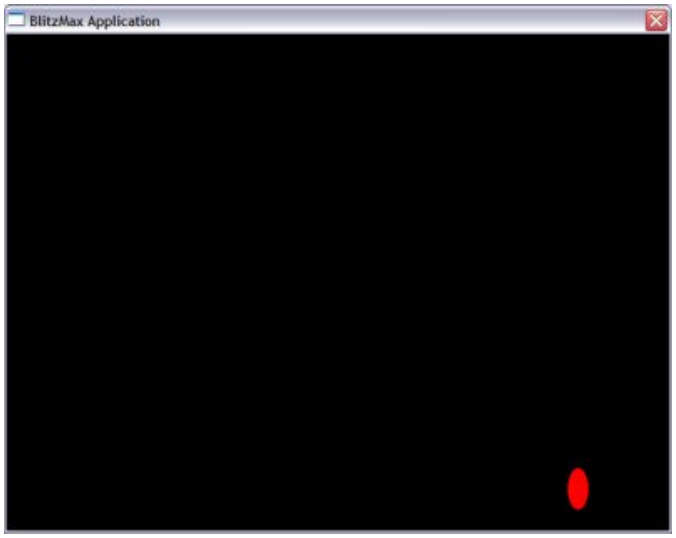
Let us demonstrate this capability with a simple example (note the new additions to our previous program):-

```
Graphics 640,480,0

Type TSpaceShip
    Field X:Int
    Field Y:Int

    Method UpdateState()
        If X>500 Then
            SetColor 255,0,0
        Else
            SetColor 255,255,255
        EndIf
    End Method

End Type

SpaceShip:TSpaceship=New TSpaceShip
HideMouse()

Repeat
    Cls
    SpaceShip.X=MouseX()
    spaceship.Y=420
```

```
    SpaceShip.UpdateState()
    DrawOval SpaceShip.X, SpaceShip.Y,20,40
    Flip
Until MouseHit(1)
End
```

Now build and run the above program. Now we can see our little SpaceShip (which is now an oval rather than a rectangle) turns red when it moves too far to the right

.



The new instructions of interest are as indicated below. Behaviours of our object can be programmed within the **Method** ... **End Method** block.

This particular **Method**, which we have chosen to call **UpdateState**, checks for the X position of the SpaceShip object and sets the drawing color to red (remember our previous Tutorial: **SetColor Red, Green, Blue**) if the value in the variable X is greater than 500, otherwise (else) it sets the drawing color to White.

```
Method UpdateState()
   If X>500 Then
      SetColor 255,0,0
   Else
      SetColor 255,255,255
   EndIf
End Method
```

In the previous tutorial, we were introduced to the **IF** statement, this is another type of conditionals, called the **If, Else EndIf** statement.

To invoke the UpdateState behaviour, we issue the instruction **SpaceShip.UpdateState**:-

```
Repeat
   Cls
   SpaceShip.X=MouseX()
   spaceship.Y=420
   SpaceShip.UpdateState()
   DrawOval SpaceShip.X, SpaceShip.Y,20,40
   Flip
Until MouseHit(1)
```

Now to make matters a bit more interesting, let us re-arrange our program a little better by using an actual spaceship image instead of trying to draw our own.

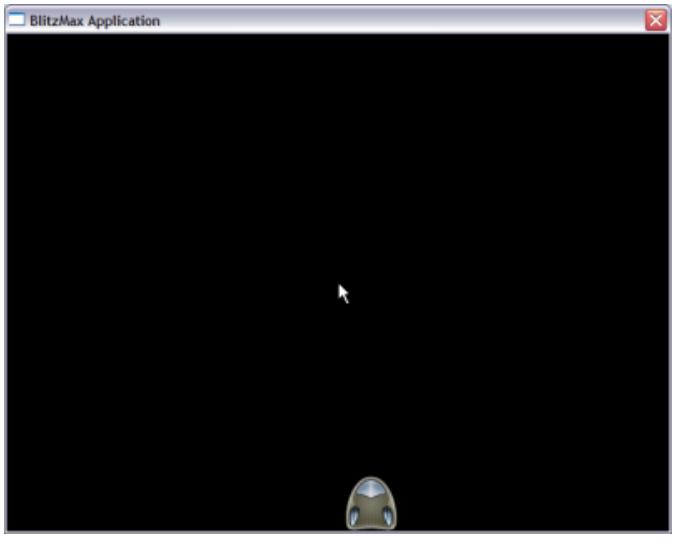
** Note that some of the images on tutorial came from www.limefly.net (link to their copyright faq here) but they are free to be used for non-commercial purposes.

We need a new field to hold the image and a new DrawSelf method and a way to load the image onto the

SpaceShip instance:-

```
Graphics 640,480,0

Type TSpaceShip
    Field X:Int
    Field Y:Int
    Field  Image:TImage

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

    Method UpdateState()
        If X>500 Then
            SetColor 255,0,0
        Else
            SetColor 255,255,255
        EndIf
    End Method

End Type

SpaceShip:TSpaceship=New TSpaceShip
HideMouse()

ImageName:string="http::www.2dgamecreators.com/tutorials/gameprogramming/basic
/blobship_1-1.png"
SpaceShip.Image:TImage=LoadImage(LoadBank(ImageName))

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

Repeat
    Cls
    SpaceShip.X=MouseX()
    spaceship.Y=420
    SpaceShip.UpdateState()
    SpaceShip.DrawSelf()
    Flip
Until MouseHit(1)
End
```

Now we have a proper space ship on display :)



Let us make sure we understand what we have just done.

```
Graphics 640,480,0

Type TSpaceShip
   Field X:Int
   Field Y:Int
   Field  Image:TImage

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

We add a new attribute to our type called Image to hold our image of type TImage which is a built-in BlitzMax variable type to hold images.

We then created a new method called DrawSelf which simply draws the image of the spaceship onto the graphics screen via BlitzMax's **DrawImage** function

```
ImageName:string="http::www.2dgamecreators.com/tutorials/gameprogramming/basic
/blobship_1-1.png"
SpaceShip.Image:TImage=LoadImage(LoadBank(ImageName))
```

The above two lines looks complicated but all it does is allow us to load an image from a website (a powerful BlitzMax feature). A slightly less complicated way (if you have the image on the hard-disk or download this ) is:-

```
ImageName:string="blobship_1-1.png"
SpaceShip.Image:TImage=LoadImage(ImageName)
```

This piece of code below checks to make sure that the image loads properly. Sometimes we lose connection to the internet or get the name of the imagefile wrong.

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

Then lastly, we execure the DrawSelf method in our main loop.

```
Repeat
  Cls
```

```
        SpaceShip.X=MouseX()
        spaceship.Y=420
        SpaceShip.UpdateState()
        SpaceShip.DrawSelf()
        Flip
    Until MouseHit(1)
    End
```

I

Now before we leave this section, I would like to re-arrange our program a little bit to make it more readable.

```
Rem
Our First BlitzMax Game

----------SETUP GAME CONDITION----------
EndRem
Graphics 640,480,0
HideMouse()

SpaceShip:TSpaceship=New TSpaceShip
ImageName:string="http::www.2dgamecreators.com/tutorials/gameprogramming
/basic/blobship_1-1.png"
SpaceShip.Image:TImage=LoadImage(LoadBank(ImageName))

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

' ----------MAIN LOOP----------
Repeat
    Cls
    SpaceShip.X=MouseX() '<-----User Input
    spaceship.Y=420
    SpaceShip.UpdateState()
    SpaceShip.DrawSelf()
    Flip
Until MouseHit(1) '<-----Check End Game Here
End

' ----------TYPES----------
Type TSpaceShip
    Field X:Int
    Field Y:Int
    Field  Image:TImage

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

    Method UpdateState()
       If X>500 Then
          SetColor 255,0,0
       Else
          SetColor 255,255,255
       EndIf
    End Method

End Type
```

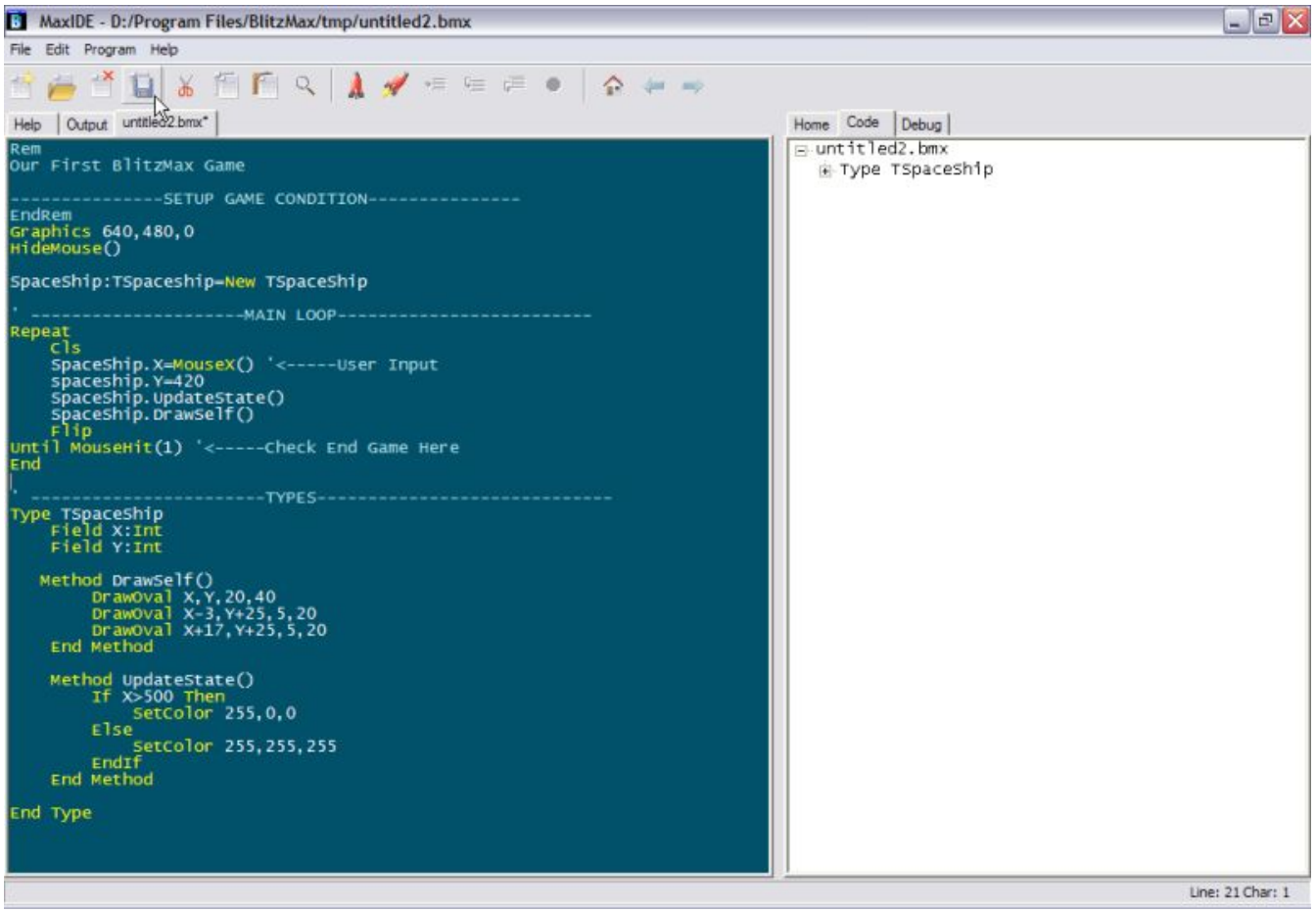
I have now introduced something new, the **Remark Block** which is everthing contained within the **Rem** .... **End Rem** block. This allow us to make notes or explain parts of our program to others (and ourself, as we tend to forget

stuff).

Remarks (or comments) can also be done by using the apostrophe (') at the start of the line. BlitzMax will ignore anything after the apostrophe or within the Remark Block.

As our program is now getting bigger, we don't want to continually type in the program as we go along. Some of you may have been cutting and pasting from this page into the MaxIDE (which is also a good idea) but saving the file and reloading it would probably constitute a good idea as well.

So to save the program you have just typed, save it onto a known location by clicking the save button as shown below:-



Summary

At the beginning of this tutorial I mentioned that the elements of a game are as follows:-

- Instructions to set-up the conditions for the game environment
- A Main Loop
- A way for the player to control his player object
- Instructions to update the states of the player, the enemies, the bullets etc
- Instructions on how to load an image into our player type
- Instructions to display the game objects onto the graphic screen
- A way to end the program
- A way to save the file we are working on

If you look at the re-arranged program, I have commented the program to reflect the various game elements. We still do not have any bullets or enemies but the framework of a game is now beginning to take shape. In the next tutorial we will built our game up more.

We must also not forget that we have touched on one of the most powerful feature of BlitzMax, the **Type** construct. A fair amount of discussion has arised because of this feature. We will not get into thatdebate but rather concentrate on using it as just another BlitzMax feature.

Blitzmax commands introduced in this tutorial

We now know more commands:-

Function **HideMouse**()
Make the mouse pointer invisible.

Keyword **Type**
Begin a user defined Type declaration.

Keyword **EndType**
End a user defined Type declaration.

Keyword **Field**
Declare a Field variable.

Keyword **Method**
Begin a Method declaration.

Keyword **EndMethod**
End a Method declaration.

Keyword **Else**
Else provides the ability For an **If Then** construct to execute a second block of code when the **If** condition is False.

Keyword **EndIf**
Marks the End of an If Then block.

Function **DrawImage**(image:TImage,x#,y#,frame=0)
Draw an image to the back buffer.

Function **LoadImage**:TImage(url:Object,flags=-1)
Load an image.

Function **LoadBank**:TBank(url:Object)
A bank containing the binary contents of **url**, or null if **url** could not be opened.

Keyword **Rem**
Begin a remark block.

Keyword **EndRem**
End a remark block.

Back to the index, Next Tutorial.