Learning 2D Game Programming: Images Part 1 - Loading and Drawing

Part 1, Part 2, Part 3

Table of Contents

Introduction

The BlitzMax Help documentation has this to say about images:-

*Images are pre-rendered chunks of graphics that can be efficiently drawn using a single **DrawImage** command. Images are typically stored in png, bmp or jpg format, and can be loaded using the **LoadImage** command.*

We are all familiar with image files, either from digital photos, created via programs such as photoshop or downloaded from the internet. BlitzMax has functionalities to load, draw and manipulate these images.

LoadImage

To load an image into a BlitzMax program, we use the **LoadImage** function. It has the following syntax:-

```
Function LoadImage:TImage( url:Object,flags=-1 )
```

Images, when loaded into a BlitzMax program, are stored in a TImage data type. Note that the **LoadImage** function has two parameters; url and flags. We'll ignore the second parameter for a while and simply concentrate on the first.

Location, Location, Location

It is very important that we know where the image to be loaded resides. The safest (not necessarily the best way) is to use a fully qualified location of the image. For example:-

```
Local Alien:TImage=LoadImage("D:/Program Files/BlitzMax
/tmp/cartoonufo_1-1.png")
```

Note that in order to ensure cross-platform compatibility, use the forward slash instead of the backslash.

If we do not fully qualify the path, then it is assumed to be relative to the current folder of the executing program. You can know your application directory from the AppDir global variable. Try and run the following short program:-

```
Print AppDir
```

So if the executing program and your image is in the same folder, you can simply use the following construct to load a png image file called cartoonufo_1-1.png.

```
Local Alien:TImage=LoadImage("cartoonufo_1-1.png")
```

The image file can also be loaded from the internet but needs to be loaded into a **bank** first as follows:-

```
Local URL:String="http::www.2dgamecreators.com/tutorials
/gameprogramming/basic/"
Local Bank:TBank=LoadBank(URL+"blobship_1-1.png")
Local Player:TImage=LoadImage(Bank)
```

Note that the normal form of a URL ie **http://** will not work. You need to use **http::** instead

Another way is to load from an IncBinned image. **Incbin** (include binary) embeds an external data file in a BlitzMax program that can then be read using the "incbin::" device name

```
Incbin "D:/Program Files/BlitzMax/tmp/blobship_1-1.png"
Local Player:TImage=LoadImage("incbin::blobship_1-1.png")
```

It is good practice to check that the image loaded without error. Error could be caused either by a wrong filename, wrong location or unrecognised format.

```
Graphics 640,480
Incbin "D:/Program Files/BlitzMax/tmp/blobship_1-1.png"
Local Player:TImage=LoadImage("incbin::blobship_1-1.png")
If Player=Null Then
   Print "Error, File Cannot be Loaded."
```

```
  End
EndIf
```

DrawImage, backbuffer

Now that we have learnt how to load an image into a BlitzMax program, we can start to display the image on the graphic screen.

The **DrawImage** function has the following syntax:-

```
Function DrawImage( image:TImage,x#,y#,frame=0 )
```

For the moment, let us ignore the Frame parameter and concentrate on the first three parameters which are nothing more than

- The Image to be displayed as per the TImage variable returned by the above **LoadImage** function
- The X,Y location where that image is to be displayed on the graphic screen:-

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

Repeat
  Cls
  DrawImage Player,MouseX(),MouseY()
  Flip
Until KeyHit(key_escape) Or AppTerminate()
End
```

When we run the above program, we can use the mouse to move the spaceship around the graphic screen.
If we were to remove the clearscreen command (CLS), we will get the image on the right

This illustrates what happens when we issue the drawimage command. The Image is drawn onto a drawing buffer (in this case also known as the backbuffer) and we only see the result of the drawing when the **flip** command is called.

If we perform a clearscreen of the backbuffer each time prior to drawing our image and we change the image's drawn location, we create the illusion of motion. If we dont clear the screen, as in the second example, the trail of the previous images drawn remained, showing a nice pattern but not the effect that we want.

Fractional positions

The sharp eyed amongst you would have noticed that you can supply fractional numbers to DrawImage locations as the x,y parameters are of type float instead of integers. This at first seems strange as pixels are by its nature integers
.
Modern 3D cards actually will "smear" (anti-alias) images drawn at fractional positions to allow for smoother animation. If you peer careful at the two images from the program below, you'll notice that the second spaceship appears a bit more blurry from the anti-aliasing.

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

Repeat
  Cls
  DrawImage Player,MouseX(),MouseY()
  DrawImage Player,MouseX()+50.5,MouseY()+0.3
  Flip
Until KeyHit(key_escape) Or AppTerminate()
End
```

ImageWidth and ImageHeight

Once an image has been loaded, we can use the **ImageHeight** and **ImageWidth** function to retrieve the width and height (in pixels) of the image respectively.

The ImageHeight function has the following syntax and a similar syntax for ImageWidth:-

```
Function ImageHeight( image:TImage )
```

Image Handle: MidHandleImage and AutoMidHandle

Remember that the DrawImage function has x,y location parameters. Where the image is drawn is also affected by the **handle** of the image.

Normally the handle is at the top left hand corner of the image (which by its nature are really rectangluar in shape) .
Let's illustrate what I'm trying to say

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

Repeat
  Cls
  DrawRect MouseX(),MouseY(),ImageWidth(Player), ImageHeight(Player)
  DrawImage Player,MouseX(),MouseY()
  Flip
Until KeyHit(key_escape) Or AppTerminate()
End
```

Run the above program and the program below which has a MidHandleImage statement and see the difference between the two:-

```
Graphics 640,480
Local URL:String="http::www.2dgamecreators.com/tutorials
/gameprogramming/basic/"
Local Player:TImage=LoadImage(LoadBank(URL+"blobship_1-1.png"))
MidHandleImage Player
Repeat
  Cls
  DrawRect MouseX(),MouseY(),ImageWidth(Player), ImageHeight(Player)
  DrawImage Player,MouseX(),MouseY()
  Flip
Until KeyHit(key_escape) Or AppTerminate()
End
```

We can see the difference in the location of the drawn spaceship images relative to the mouse cursor. Notice that the white rectangle is still drawn the same in both instance ie top left corner is at the mouse position.

What the **MidHandleImage** function does is to set the handle to the middle of the image. Thus we see the image drawn in such a way as to have the mouse location at the center (middle) of the image.

If we were to use the **AutoMidHandle** (true) function before loading our images, all image handles will be set to the center of the image. This is frequently the recommended way to load our images.

Note that the **SetHandle** function does not affect the **DrawImage** command.

SetOrigin

Another setting which has an impact on the DrawImage command is **SetOrigin** a,b
The DrawImage is offset by the amount a and b (added to x and y)

For example

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

Repeat
  Cls
  SetOrigin 0,0
  DrawRect MouseX(),MouseY(),ImageWidth(Player), ImageHeight(Player)
  SetOrigin 100,100
  DrawImage Player,MouseX(),MouseY()
  Flip
Until KeyHit(key_escape) Or AppTerminate()
End
```

Notice the offset from the mouse cursor (of 100+y and 100+x) caused by the SetOrigin setting

We can have a bit of fun with the SetOrigin command.
health Warning; Do not try this if you don't have broadband (or substitute with your wallpaper image on your hard-disk)

```
Graphics 640,480
Local URL:String="http::www.2dgamecreators.com/tutorials
/gameprogramming/basic/"
Local player:TImage=LoadImage(LoadBank(URL+"blobship_1-1.png"))
url="http::imgsrc.hubblesite.org/hu/db/2001/25/images/a/formats
/1280_wallpaper.jpg"
Local Background:TImage=LoadImage(LoadBank(url))
Local x:Float, y:Float

Repeat
  Cls

  If MouseX()>500 x :- 1
  If MouseX()<100 x :+ 1

  If MouseY()>350 y :- 1
  If MouseY()<100 y :+ 1

  SetOrigin x,y
  DrawImage Background,0,0
  SetOrigin 0,0
  DrawImage player,MouseX(),MouseY()
  Flip
Until KeyHit(key_escape) Or AppTerminate()
End
```

Move the SpaceShip to the edges of the screen to see the scrolling effect

Summary

In this tutorial we saw how easy it is to load an image and then display the loaded image onto a graphics screen. The common mistake most beginners make is not specifying the correct location of the image file in the LoadImage function.

Once the image is loaded, the Cls, drawimage and flip commanset is all that is needed to display an image on the graphic screen.

In the next tutorial we will be looking at functions which allows us to perform realtime scaling and rotation of our loaded image.

Back to the index, Next Tutorial