Drawing Sprites

Once we figured out how to display the images with transparent background next thing was to actually manage all the sprites being displayed. Moreover, at scene multiple images(Sprites) need to displayed. So we need to figure out some way to do it. Solution to this problem was to use a common bitmap object to which images are displayed. Layering can also be applied on the same. This is because the ones which are copied to the bitmap image will be behave as the lower most sprite or layer.

For this we write a class called Canvas. This class has an object of type Bitmap. Bitmap is basically a class which stores images. During the updation period of the current frame, control program decide what to be displayed and object of the Canvas class is called. Which in turn copies the supplied image over the bitmap object. Once updation period is over, final bitmap object is given to the transparent framework. Hence the frame is now updated. Canvas class has following functions:

* Canvas Constructor: This function takes the values of width and height of canvas as input. It basically initializes all the objects that will be used inside the Canvas class itself.
* Resize: This function is used to change the current size of the bitmap object.
* GetBitmap: This function returns the current image hold by the canvas. Used to supply the frame to transparent framework.
* AddBitmap: This function is used to overlay image over the current frame. This function is made fast enough, as it will called multiple times per iteration.
* Clear: This function is used to empty the bitmap object.

using System;

using System.Drawing;

using System.Drawing.Imaging;

namespace Uni\_Form\_Trans\_Test\_01

{

public class Canvas

{

private Bitmap BitmapObject;

public int Width, Height;

Graphics gfx;

// Constructor for bitmap being passed as default value

public Canvas(Bitmap bmp)

{

BitmapObject = bmp;

gfx = Graphics.FromImage(BitmapObject);

}

// Constructor for only make canvas

public Canvas(int width, int height)

{

BitmapObject = new Bitmap(width, height);

this.Width = width;

this.Height = height;

gfx = Graphics.FromImage(BitmapObject);

}

public void Resize(int width, int height)

{

if (BitmapObject != null)

BitmapObject.Dispose();

BitmapObject = new Bitmap(width, height);

this.Width = width;

this.Height = height;

gfx = Graphics.FromImage(BitmapObject);

}

// Returns current bitmap

public Bitmap GetBitmap()

{

return BitmapObject;

}

// This is an important function. Overlays the new bitmap over the existing canvas

public void AddBitmap(ref Bitmap bmpObj, int x, int y)

{

gfx.DrawImage(bmpObj, new Point(x, y));

}

public void Clear()

{

gfx = Graphics.FromImage(BitmapObject);

gfx.Clear(Color.Transparent);

//gfx.DrawRectangle(Pens.Red, 0, 0, Width - 1, Height - 1);

}

public void FlipBitmap()

{

BitmapObject.RotateFlip(RotateFlipType.Rotate180FlipY);

}

public void SetTransparency(float alphaLevel)

{

Bitmap img = new Bitmap(BitmapObject);

Clear();

Graphics graphics = Graphics.FromImage(BitmapObject);

ColorMatrix colormatrix = new ColorMatrix();

colormatrix.Matrix33 = alphaLevel;

ImageAttributes imgAttribute = new ImageAttributes();

imgAttribute.SetColorMatrix(colormatrix, ColorMatrixFlag.Default, ColorAdjustType.Bitmap);

graphics.DrawImage(img, new Rectangle(0, 0, BitmapObject.Width, BitmapObject.Height), 0, 0, img.Width, img.Height, GraphicsUnit.Pixel, imgAttribute);

graphics.Dispose(); // Releasing all resource used by graphics

img.Dispose();

imgAttribute.Dispose();

}

}

}