Animation

By this point we had developed frameworks for displaying any image with transparent background. But for this project, we were required to develop an animation framework. One that can handle animations itself. Following are requirements for animation framework:

* It should be able load the whole animation sequences into the memory.
* One should only call to the update function to display the next animation frame, without worrying of which frame should be displayed.
* Animation framework should decide next frames.
* It should maintain a loop of animation.
* It should be able to keep a check FPS.

For this purpose, we wrote another class called animate. In the load function of the windows form application, we would load all the essential resources and create objects of type animate. Then when we need to add the frame of any of the animations, then all left to do is to call the animation object update function. That function would decide itself which frame to display, etc. All animation classes shares a common canvas object. So to add something to current frame, all need to do by the animation framework is to call the common canvas object.

using System;

using System.Drawing;

using System.Resources;

namespace Uni\_Form\_Trans\_Test\_01

{

public class Animate

{

#region Declarations

private string name;

private int length;

private Canvas canvas;

private int delay, idx;

private Point[,] eyeLocations;

private string[] fileNames;

static public float rescaleFactor = 1f;

#endregion

public Animate(string name, int width, int length, int delay, float rescaleFactor, ref Canvas canvas, ref ResourceManager rm)

{

this.name = name;

this.length = length;

this.canvas = canvas;

this.delay = delay;

idx = 0;

eyeLocations = new Point[length, 2];

fileNames = new string[length];

Animate.SetScale(rescaleFactor);

for (int i = 0; i < length; i++)

{

string fileName = name;

int l = (i.ToString()).Length;

for (int j = 0; j < width - l; j++)

fileName += '0';

fileName += i.ToString();

LoadBitmap2(fileName, ref fileNames[i]);

}

//Read Location Here

string fn = "Locations/" + name;

string loc = System.IO.File.ReadAllText(fn);

string temp = "";

int flag = 0;

float p = 0;

int spCount = 0;

string xc = "";

string yc = "";

for (int i = 0; i < loc.Length; i++)

{

if (loc[i] != ' ')

{

temp += loc[i];

}

else if (loc[i] == ' ')

{

spCount++;

if (spCount == 1)

{

xc = temp;

}

if (spCount == 2)

{

yc = temp;

spCount = 0;

eyeLocations[(int)p % length, flag].X = int.Parse(xc);

eyeLocations[(int)p % length, flag].Y = int.Parse(yc);

p = p + 0.5f;

flag = 1 - flag;

}

temp = "";

}

}

}

public void SetDelay(int delay)

{

this.delay = delay;

}

public bool Update(long cnt, int x, int y)

{

// Returns True If Current Animation Is Over

idx %= length;

if (cnt % delay == 0)

{

Bitmap tmp = (Bitmap)Bitmap.FromFile(fileNames[idx]);

Bitmap tmp2 = new Bitmap(tmp, new Size((int)(tmp.Width / rescaleFactor),

(int)(tmp.Height / rescaleFactor)));

canvas.AddBitmap(ref tmp2, x, y);

tmp.Dispose();

tmp2.Dispose();

idx++;

}

else

{

Bitmap tmp = (Bitmap)Bitmap.FromFile(fileNames[idx]);

Bitmap tmp2 = new Bitmap(tmp, new Size((int)(tmp.Width / rescaleFactor),

(int)(tmp.Height / rescaleFactor)));

canvas.AddBitmap(ref tmp2, x, y);

tmp.Dispose();

tmp2.Dispose();

}

if(idx == length)

{

return true;

}

return false;

}

public void Reset()

{

idx = 0;

}

public static void LoadBitmap(string fileName, ref Bitmap bmp,double rescaleFactor=1.0f)

{

fileName += ".png";

fileName = "Resources/" + fileName;

bmp = (Bitmap)Bitmap.FromFile(fileName);

Bitmap tmp = new Bitmap(bmp, new Size((int)(bmp.Width / rescaleFactor),(int)(bmp.Height / rescaleFactor)));

bmp.Dispose();

bmp = tmp;

}

public void LoadBitmap2(string fileName, ref string str)

{

fileName += ".png";

fileName = "Resources/" + fileName;

str = fileName;

}

public Point GetLocation(int flag)

{

Point ret = eyeLocations[Math.Max(0, idx - 1), flag];

ret.X = (int)(ret.X / rescaleFactor);

ret.Y = (int)(ret.Y / rescaleFactor);

return ret;

}

public static void SetScale(float scale)

{

rescaleFactor = scale;

}

}

}