Adding an actor with animation

# Get a game template

A game template (the same as the one you made last week) has been created for you. It currently has three types of image defined in the Game.cs file:

* empty (there is no image in this cell – the background image will show through
* collectable (an image for an item that will be collected during the game)
* enemy (an image to represent an enemy to the main character in the game).
* mainActor (an image for the main character in the game)

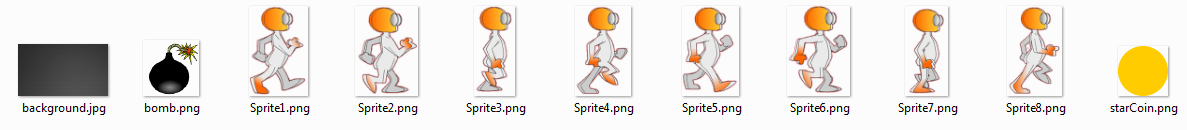
You can get it from GitHub – <https://github.com/Karsco/BasicAnimatedGameCSharp.git>

At the moment is has no graphics and the assets have not been loaded. Once the graphic resources have been added and the assets loaded, it will draw one of each on the screen (with the exception of the mainCharacter as you will be setting this up now).

Press F5 to run and debug the program – you will see a grey window. In the error log you will notice that it tells you that some image references are null.

# Add some graphics to the project resources

There is a folder on Moodle (in Learning Materials) with a set of sprite images, a background and collectable (star) and enemy (bomb) images.



Add these resources to your project.

# Add the code to set the background, enemy and collectable images (one of each)

This code will go in the GameGraphicsEngine.cs class.

Add the code to load the bitmap assets to the game:

private void loadAssets() //load assets for game - images and sounds - here

{

**bg = SingleLevelGame.Properties.Resources.background;**

**gameCollectable = SingleLevelGame.Properties.Resources.starCoin;**

**gameEnemy = SingleLevelGame.Properties.Resources.bomb;**

}

**\*\*\*\*TEST THAT YOUR GAME WORKS – PRESS F5 (you should see one bomb and one starCoin)\*\*\*\***

# Add a mainCharacter object class

We will call this character an Actor and use the code below to create a template.

MAKE A NEW CLASS CALLED **Actor.cs** and add the following code

1. Code to add the drawing libraries so that you can set up the bitmap images

using System.Drawing;

1. Some variables to hold the Actor’s data (it’s position, number of available sprites, a list to store the sprite set and the current sprite image to be shown

//--------Class variables---------//;

private static int x = 0;

private static int y = 0;

private static int maxSprites;

private static List<Bitmap> spriteSet = new List<Bitmap>(); //sprite set

private static int currImage = 1; //current sprite image in animation series

1. Some class properties so that other objects can find out where the actor is and what image it should currently be displaying

//---------Class properties--------//

public int xPos //property holds current x position

{

get { return x; }

set { x = value; }

}

public int yPos //property holds current x position

{

get { return y; }

set { y = value; }

}

public Bitmap currentImage //property holds the current animation sprite

{

get { return getCurrentImage(); }

}

1. Add a constructor method with two parameters so that the game code can set where it wants the Actor to be placed.

//--------Class functions--------//

public Actor(int startX, int startY, String gameSpriteName, int numSprites) //constructor sets first image and starting position

{

x = startX;

y = startY;

loadSpriteSet(gameSpriteName, numSprites);

maxSprites = numSprites;

currImage = 1;

}

//load individual sprites from files into list

private void loadSpriteSet(String spriteName, int numSprites)

{

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

{

int currSpriteNum = i + 1;

String resourcePath = "..\\..\\Resources\\Sprite" + currSpriteNum.ToString() + ".png";

Bitmap thisSprite = new Bitmap(Image.FromFile(resourcePath));

spriteSet.Add(thisSprite);

}

}

private Bitmap getCurrentImage() //return the bitmap for the image that should be showing at the moment

{

Bitmap thisSprite = spriteSet.ElementAt(currImage - 1);

return thisSprite;

}

**//\*\*\* YOUR CODE WILL GO HERE LATER \*\*\*//**

**\*\*\*\*TEST YOUR GAME – DOES IT DO ANYTHING DIFFERENT YET? WHY NOT? \*\*\*\***

# Add your main character to the game

You will need to tell the game that you want a main character to be added. You will need to do three things:

1. Tell the game that you want a main character added. Add the following to the class variables area:

/\*----------Class variables----------\*/

private GameGraphicsEngine gEngine;

**private static Actor mainCharacter;**

1. You will need to add the main character to the game. The main character will operate on a different game layer from the game level and so you should not add the code to the loadLevel() function BUT you should add a new function just below it to add the character:

//main character operates on a different layer so deal with it separately

//add a new actor at position 0,0 in grid with 8 sprites named “Sprite” followed by 1 to 8

public void loadMainCharacter()

{

mainCharacter = new Actor(0, 0, "Sprite", 8);

}

1. Make sure that the loadMainCharacter() function is run just after the loadLevel() function. Add the line to startGraphics

public void startGraphics(Graphics g) //start graphics engine

{

loadLevel();

**loadMainCharacter();**

gEngine = new GameGraphicsEngine(g);

gEngine.initialise();

}

**\*\*\*\*TEST YOUR GAME – DOES IT DO ANYTHING DIFFERENT YET? WHY NOT? \*\*\*\***

# Add the instructions to the GameGraphicsEngine to draw the main character

The GameGraphicsEngine will need to draw the main character on top of the level. To do this it must be given the main character’s bitmap and its position.

For this, two things are needed:

1. The Game will need to pass the character’s bitmap and position to the render function. Change the instruction to render the game (in the Game.cs code) so that it passes on the main character (there is just one character at the moment but later there may be more).

public void renderGraphics(Graphics g) //start graphics engine

{

**gEngine.render(mainCharacter);**

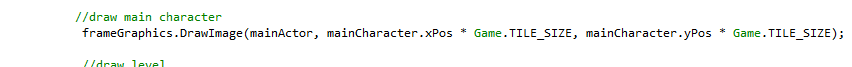
**}**

Change the instruction in the Game.cs code that runs the rendering process:

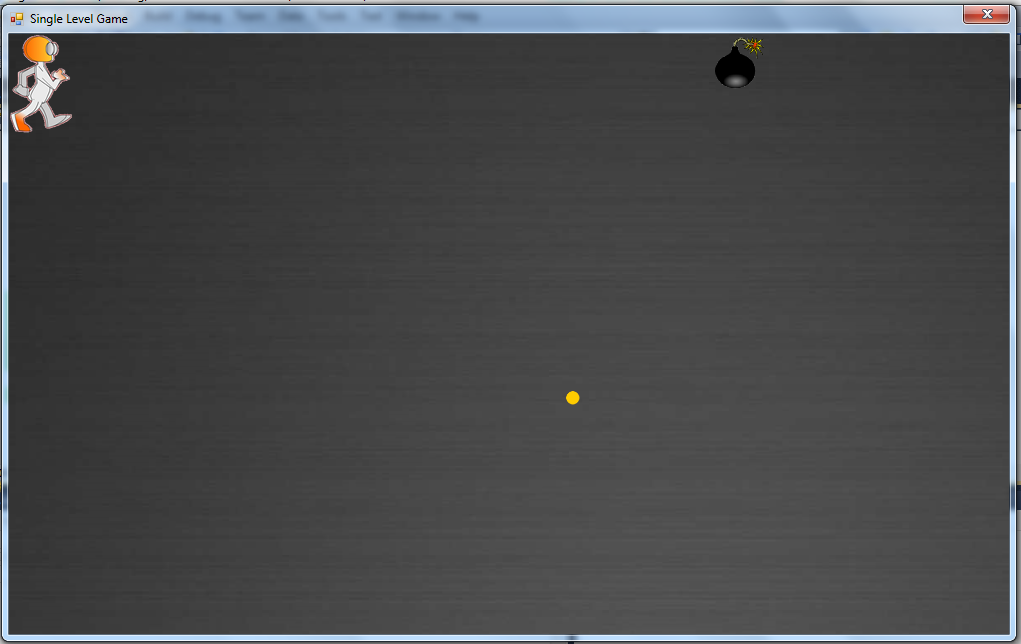
1. Now change the render code in the GameGraphicsEngine.cs code to match this

public void render(**Actor mainCharacter**)//draws graphics frame by frame, background, enemies, main character

1. JUST BEFORE the line that draws the level in the render function, add the following code to draw the main character:



**\*\*\*\*TEST YOUR GAME – DOES IT LOOK LIKE THIS? \*\*\*\***

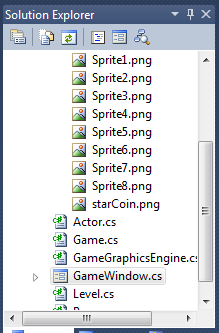
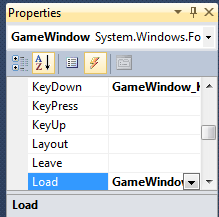


IF NOT – YOU WILL NEED TO MAKE SURE YOU GET SOME HELP TO GET IT WORKING.

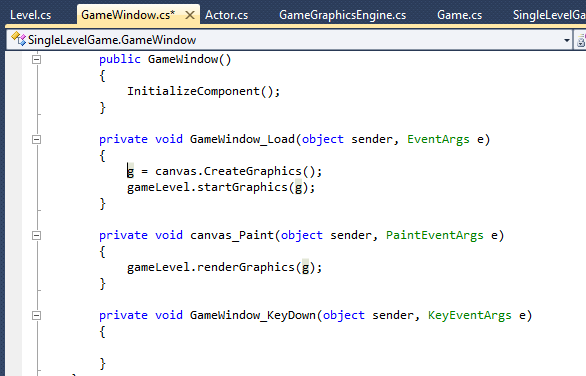
# Let’s get the actor to move on a keypress

The KeyDown event should be detected in the main window.

Open the form design window (if it isn’t already available, double click on GameWindow.cs in the solution explorer) and then select the events view in the Properties window and double-click on KeyDown.

This will open up the code window for the form and will have added a KeyDown event handler:



When a key is pressed it should be passed to the Game so that the game can control what happens when the key is pressed.

private void GameWindow\_KeyDown(object sender, KeyEventArgs e)

{

gameLevel.changeGameState(e.KeyCode);

}

You will now need to add a function to Game.cs to chanegeGameState. The function is being passed the code for the key that was pressed.

1. Add the library for forms so that the library functions for dealing with keys are available.

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Drawing;

**using System.Windows.Forms;**

1. Add a function called changeGameState that will take a key code and get the actor to move in response:

public void changeGameState(Keys keyPressed)

{

}

1. Let’s add some code to get the Actor to move right if the right arrow key is pressed.

public void changeGameState(Keys keyPressed)

{

if (keyPressed == Keys.Right)

{

mainCharacter.move(1, 0); //move character 1 in x and 0 in y direction

}

}

Of course, it won’t recognize .move(1,0) because you haven’t written it yet – it needs to be added to the Actor code.

Open the Actor.cs code window and add some code to move the character one space to the right.

public void move(int xMove, int yMove) //move actor one place in given direction

{

x = x + xMove;

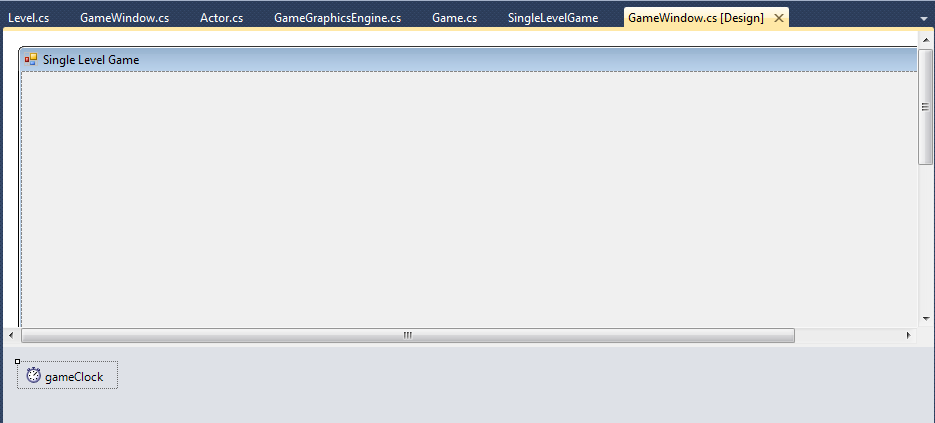
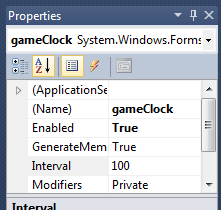
y = y + yMove;

}

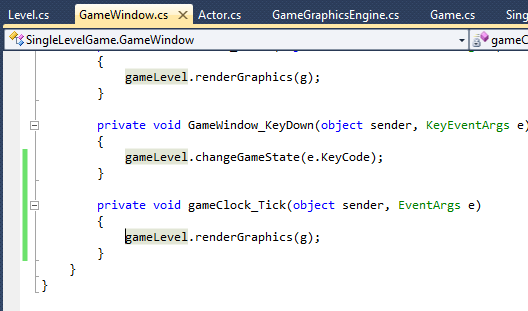
**\*\*\*\*TEST YOUR GAME – DOES IT ANIMATE YET? WHY NOT? \*\*\*\***

There is no code yet to get the gameGraphicsEngine to render the canvas over and over again.

This needs to be controlled by the game. Add a timer to the GameWindow (design), name it gameClock and set it’s Enabled property to true.

Add the code to the GameWindow.cs to make the game re-render the graphics:

 Double-click on the timer to get this code to be inserted.

Now it’s your turn

Get the character to move left, right, up and down on the relevant key press.

Adding the animation

The final part is to add the animation to the character.

The sprite images are stored in a List and there are currently 8 of them.

When a key is pressed, the Actor must move one space in the relevant direction and we now want it to also change its costume so that animation is added.

This is quite straightforward. The current image is stored as a number. Image 1 (or Sprite1) shows first. As the actor moves to the right, the current image number should go up by 1 (as long as it doesn’t go above 8, which is stored in maxSprites – the maximum number of sprites).

public void move(int xMove, int yMove) //move actor one place in given direction

{

x = x + xMove;

y = y + yMove;

**//change to next image when actor is moving to right**

**if (xMove > 0)**

**{**

**if (currImage < maxSprites)**

**currImage++;**

**else**

**currImage = 1;**

**}**

**else if (xMove < 0) //change to previous image when going left**

**{**

**//YOU CAN ADD THIS CODE**

**}**

}

Challenges

Now you have the basics of animation of the character. You can create your own sprite set (or search for some creative commons sprite sets).

How about trying to get the images to change to face left when the actor is moved to the left.

Try to get the actor to stop at the left and right of the screen (the final screen position on the right is Game.LEVEL\_WIDTH and the left is 0).

How about changing the image to a crouching position when the space bar is pressed?