XNA Programming Lecture 4

Contents

Introduction

First off grab this zip file from Blackboard, or my website “WindowsGame1_InClass_Wed_Wk2.zip”, make a new folder on your hard drive called XNA_Week4 and unpack it to the folder.

My website:

http://drewfx.com/TAFE/XNA/WindowsGame1_InClass_Wed_Wk2.zip

Blackboard:

Apply introductory object-oriented language skills [D0053|ICAB4219B]

NOTE! Please use this code and NOT YOUR OWN – This lesson is designed to work from this code only. You will run into problems if you don’t USE THIS CODE.

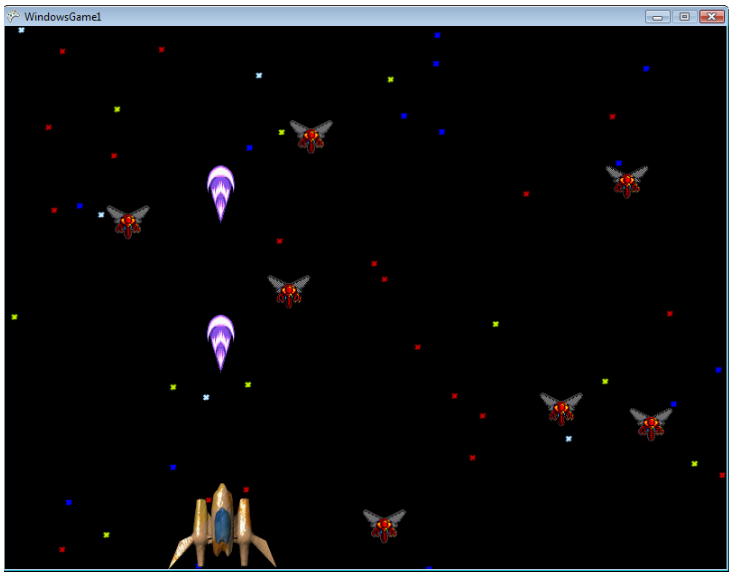
Open it up in Games Studio 3.1 from the Start Menu.

You will find it here...

<your hard Drive>\WindowsGame1_InClass_Wed_Wk2\WindowsGame1\WindowsGame1.csproj

Design Discussion.

So here's what we have so far...

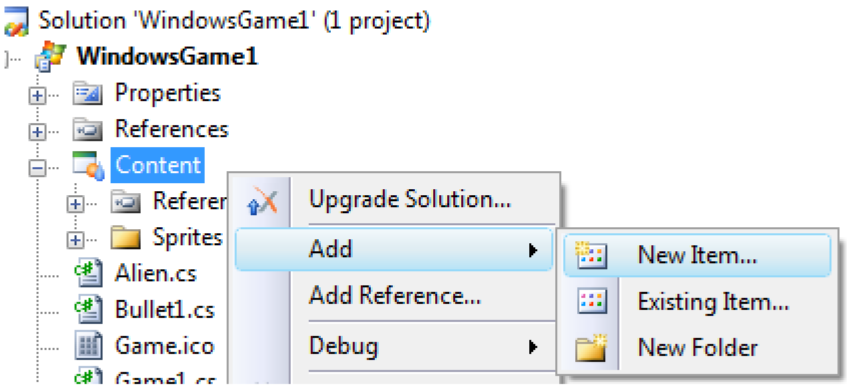


I've tweaked it a little so the bullet fires from the nose plus it's speed so by the time we see it, it comes out of the nose.

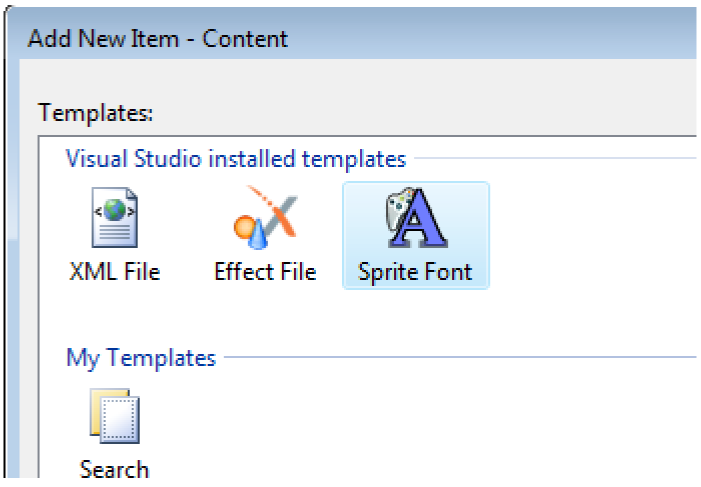
We want to add score, explosions and level detect. Here's how.

Adding Fonts.

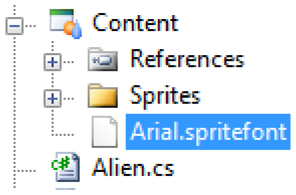
To draw score we need a font object that get saved with project. Firstly, go to the Content tree in Solution Explorer, the right click on Content, select Add->New Item...



Now select Sprite Font...

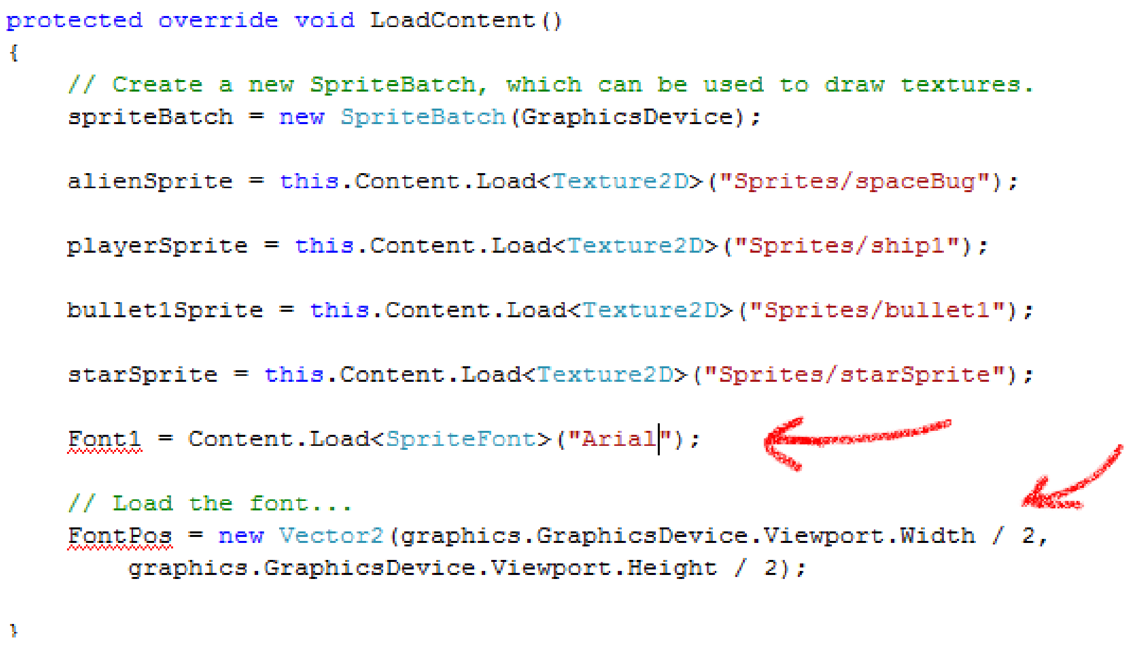


Find the item in the Content tree and rename it Arial...



MAKE SURE TO KEEP THE .SPRITEFONT suffix.

Now you'll need to load it, so go to LoadContent in Game1.cs and type this...



```
protected override void LoadContent()
{
    // Create a new SpriteBatch, which can be used to draw textures.
    spriteBatch = new SpriteBatch(GraphicsDevice);

    alienSprite = this.Content.Load<Texture2D>("Sprites/spaceBug");

    playerSprite = this.Content.Load<Texture2D>("Sprites/ship1");

    bullet1Sprite = this.Content.Load<Texture2D>("Sprites/bullet1");

    starSprite = this.Content.Load<Texture2D>("Sprites/starSprite");

    Font1 = Content.Load<SpriteFont>("Arial");

    // Load the font...
    FontPos = new Vector2(graphics.GraphicsDevice.Viewport.Width / 2,
        graphics.GraphicsDevice.Viewport.Height / 2);

}
```

You'll notice the red wavy lines under the Font1 and FontPos... this is because they need to be prototyped above.

Scroll to the top of your class and find these lines to add the two attributes...

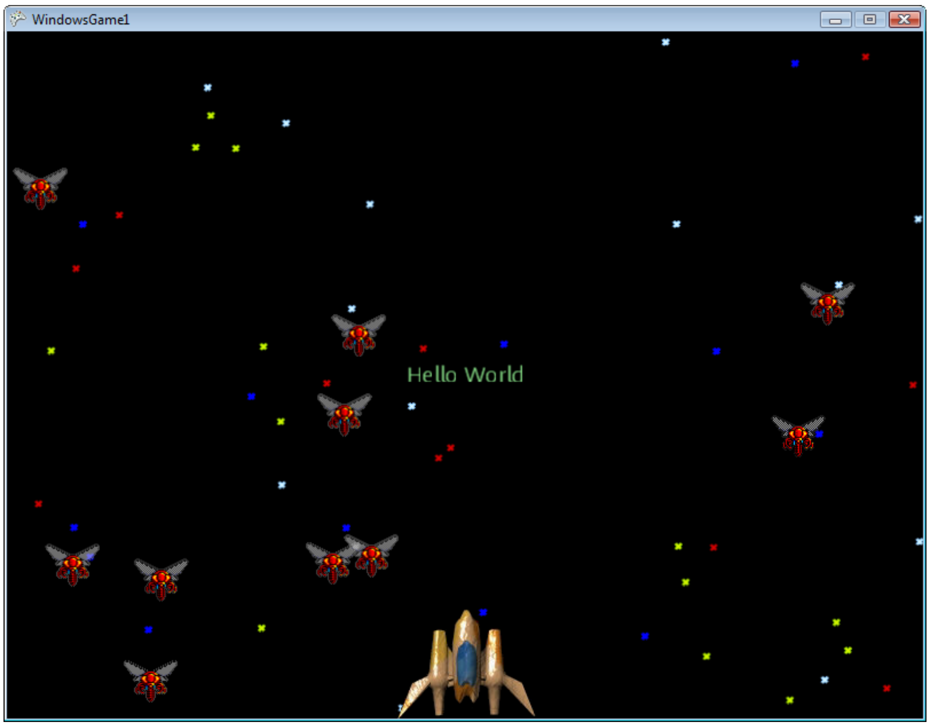
```
private static Texture2D starSprite;

int screenWidth;
int screenHeight;

List<Bullet1> bulletList = new List<Bullet1>();

Star[] myStars = new Star[50];
Alien[] myAliens = new Alien[10];
SpaceShip mySpaceShip = new SpaceShip();

SpriteFont Font1;
Vector2 FontPos;
```

Finally you want to use them. So go to the Draw method in Game1.cs

```
spriteBatch.Begin();

// Draw Hello World
string output = "Hello World";

// Find the center of the string
Vector2 FontOrigin = Font1.MeasureString(output) / 2;
// Draw the string
spriteBatch.DrawString(Font1, output, FontPos, Color.LightGreen,
    0, FontOrigin, 1.0f, SpriteEffects.None, 0.5f);
```

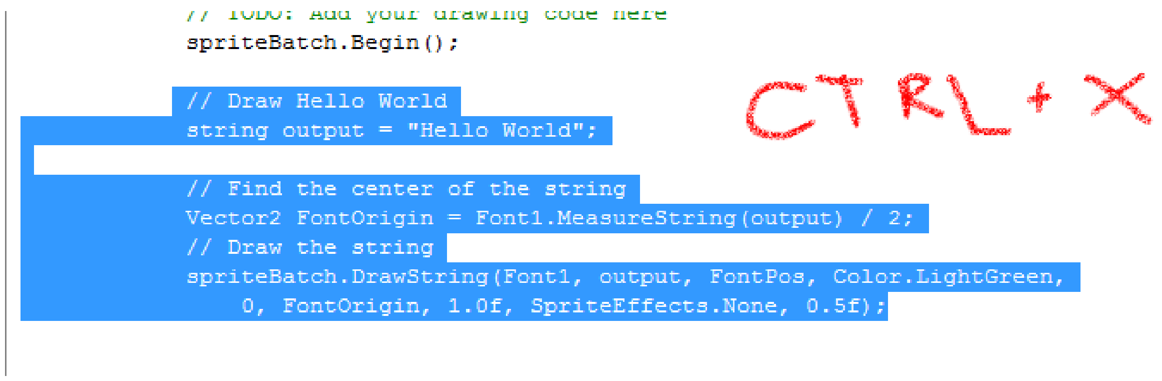
Run the game by clicking on the play button or hit CTRL + F5.

You'll see this...



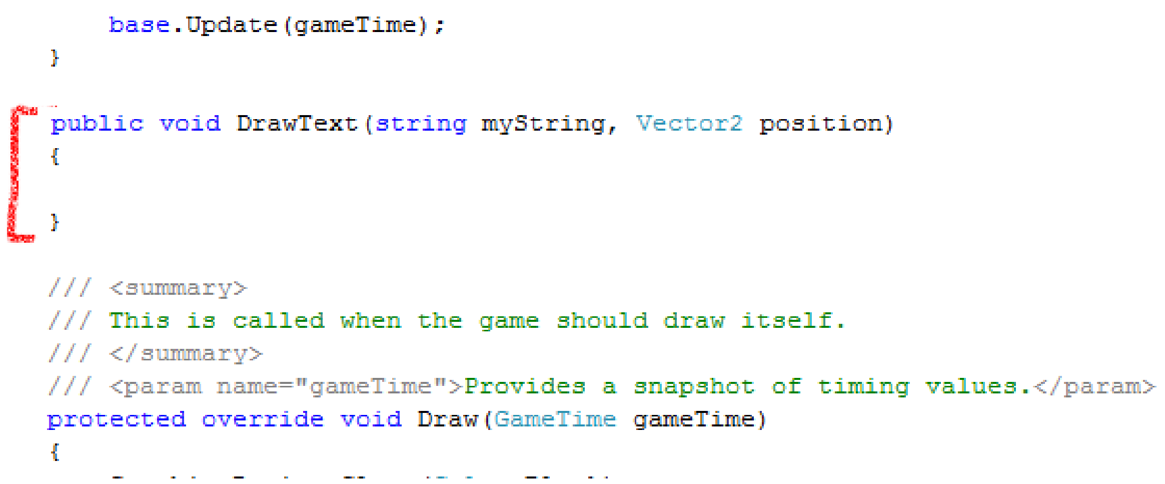
This is all very well but we need something more generic so we can draw loads of text on the screen. Let's chop that out and make a new method for generalised text drawing. So follow these steps...

Select the code you just entered and press CTRL+X (which will cut it to the clipboard)



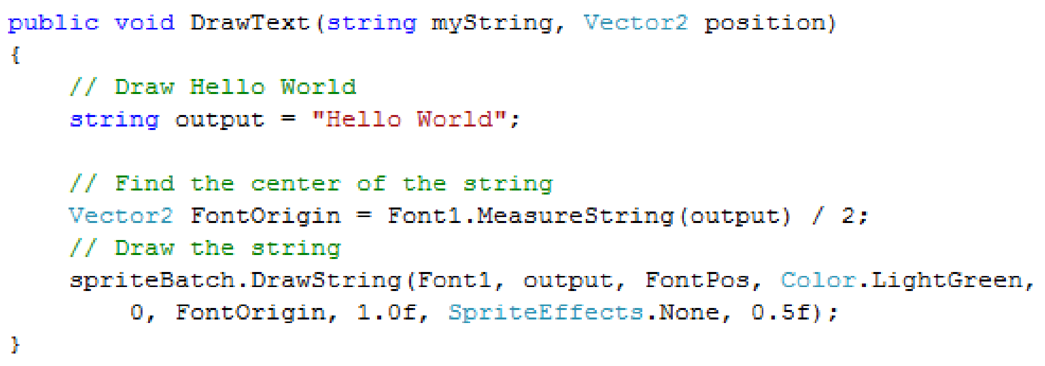
```
            // TODO: Add your drawing code here
            spriteBatch.Begin();

            // Draw Hello World
            string output = "Hello World";

            // Find the center of the string
            Vector2 FontOrigin = Font1.MeasureString(output) / 2;
            // Draw the string
            spriteBatch.DrawString(Font1, output, FontPos, Color.LightGreen,
                0, FontOrigin, 1.0f, SpriteEffects.None, 0.5f);
```

Now scroll up a little before the Draw method and add an empty DrawText method...



```
            base.Update(gameTime);
        }

        public void DrawText(string myString, Vector2 position)
        {

        }

        /// <summary>
        /// This is called when the game should draw itself.
        /// </summary>
        /// <param name="gameTime">Provides a snapshot of timing values.</param>
        protected override void Draw(GameTime gameTime)
        {
```

In between the { } braces click the cursor, and paste the code you just cut so the method now looks like this...



```
        public void DrawText(string myString, Vector2 position)
        {
            // Draw Hello World
            string output = "Hello World";

            // Find the center of the string
            Vector2 FontOrigin = Font1.MeasureString(output) / 2;
            // Draw the string
            spriteBatch.DrawString(Font1, output, FontPos, Color.LightGreen,
                0, FontOrigin, 1.0f, SpriteEffects.None, 0.5f);
        }
```

Now we can modify it. Change string output = "Hello World" to this...

```
public void DrawText(string myString, Vector2 position)
{
    // Draw Hello World
    string output = myString;
```

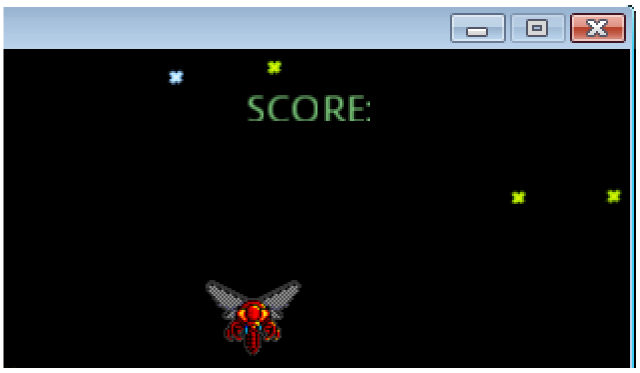
And change the Fontpos to position, FontOrigin = new Vector2(0,0)...

```
    // Find the center of the string
    Vector2 FontOrigin = new Vector2(0,0);
    // Draw the string
    spriteBatch.DrawString(Font1, output, position, Color.LightGreen,
        0, FontOrigin , 1.0f, SpriteEffects.None, 0.5f);
}
```

Now to check it go back to the Draw method and lets add a text place holder for score top right.

```
protected override void Draw(GameTime gameTime)
{
    GraphicsDevice.Clear(Color.Black);

    // TODO: Add your drawing code here
    spriteBatch.Begin();


    DrawText("SCORE:", new Vector2((screenWidth / 5) * 4, 30));
```

You'll see score appear top right..



Now let's track the score. Go to the top of the program and enter a prototype for myScore...

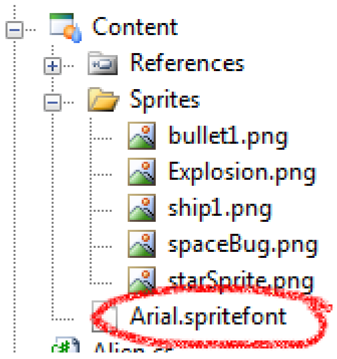
```
  SpaceShip mySpaceShip = new SpaceShip();

  SpriteFont Font1;
  Vector2 FontPos;

  int myScore = 0;

  public Game1()
  {
```

And in the Draw method just below the Score string add the following...

```
DrawText("SCORE:", new Vector2((screenWidth / 5) * 4, 30));
DrawText(myScore.ToString(), new Vector2((screenWidth / 5) * 4 + 80, 30));
```

Now let's tweak the XML for the font to look like it should, Arial, Bold, Italic.

Find the font in the Content tree...



```
Content
    References
    Sprites
        bullet1.png
        Explosion.png
        ship1.png
        spaceBug.png
        starSprite.png
    Arial.spritefont
```

Double left click on it to bring up the XML. Now find the first entry called <FontName> and change it to this...

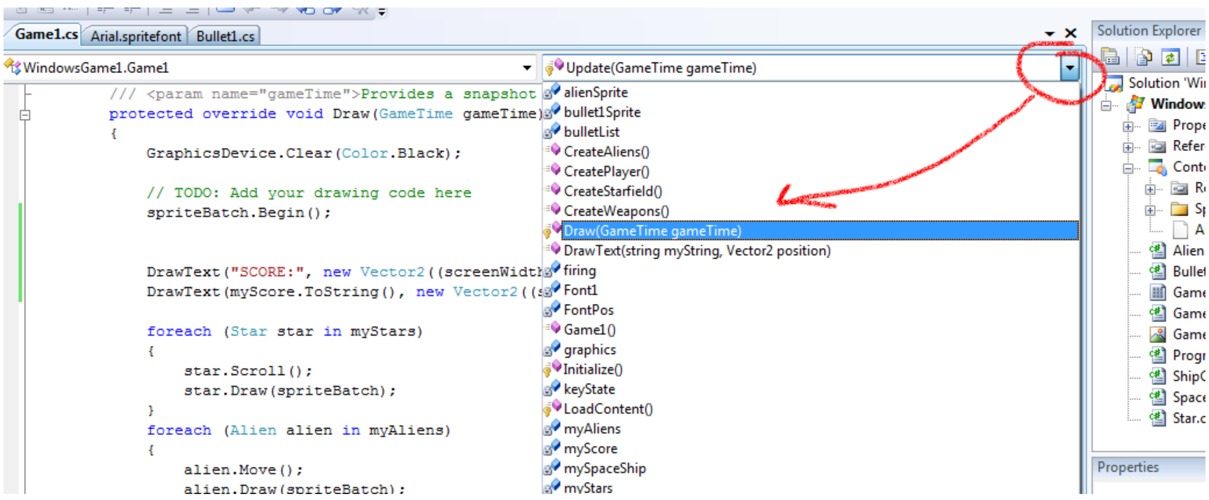
```
<!--
Modify this string to change the font that will be imported.
-->
<FontName>Arial</FontName>
```

Scroll down and find <Style> and change it to Bold, Italic...

```
<!--
Style controls the style of the font. Valid entries are "Regular", "Bold", "Italic",
and "Bold, Italic", and are case sensitive.
-->
<Style>Bold, Italic</Style>
```

Scoring

Now let's get some score when we hit the aliens. Find the function Draw in the Member browser...



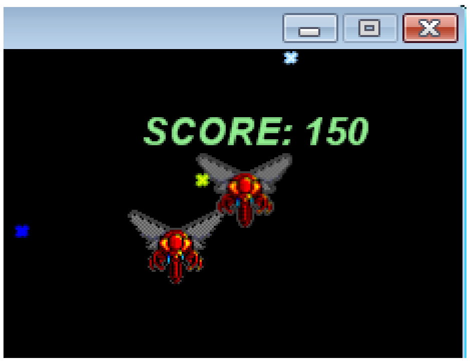
```
        /// <param name="gameTime">Provides a snapshot
        protected override void Draw(GameTime gameTime)
        {
            GraphicsDevice.Clear(Color.Black);

            // TODO: Add your drawing code here
            spriteBatch.Begin();


            DrawText("SCORE:", new Vector2((screenWidt
            DrawText(myScore.ToString(), new Vector2((s

            foreach (Star star in myStars)
            {
                star.Scroll();
                star.Draw(spriteBatch);
            }
            foreach (Alien alien in myAliens)
            {
                alien.Move();
                alien.Draw(spriteBatch);
```

```
alienSprite
bullet1Sprite
bulletList
CreateAliens()
CreatePlayer()
CreateStarfield()
CreateWeapons()
Draw(GameTime gameTime)
DrawText(string myString, Vector2 position)
firing
Font1
FontPos
Game1()
graphics
Initialize()
keyState
LoadContent()
myAliens
myScore
mySpaceShip
myStars
```

Scroll down and find the loop through the bullets and change the following code…

```
    mySpaceShip.Draw(spriteBatch);

    foreach (Bullet1 bullet in bulletList)
    {

        bullet.Draw(spriteBatch);

        // iterate through all the aliens
        // and check if their rect collides with this bullet's rect
        foreach (Alien alien in myAliens)
        {

            if (alien.AlienDead == false)
            {

                if (bullet.BoundingRectangle.Intersects(alien.BoundingRectangle))
                {
                    bullet.BulletDead = true;
                    alien.AlienDead = true;
                    myScore += 150;
                }
            }
        }
```

This will increment myScore every time you kill an alien. Have a go. Press F6 to compile (fix any errors) and CTRL+F5 to run. Shoot some aliens to increment your score.



Cool hey?

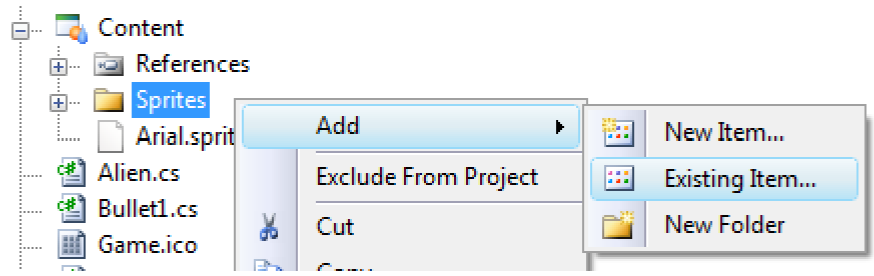
Add explosions.

Explosions are animated sprites. We load a big sprite and show only a little bit of it at a time.

Grab this sprite and save it to your current folder...

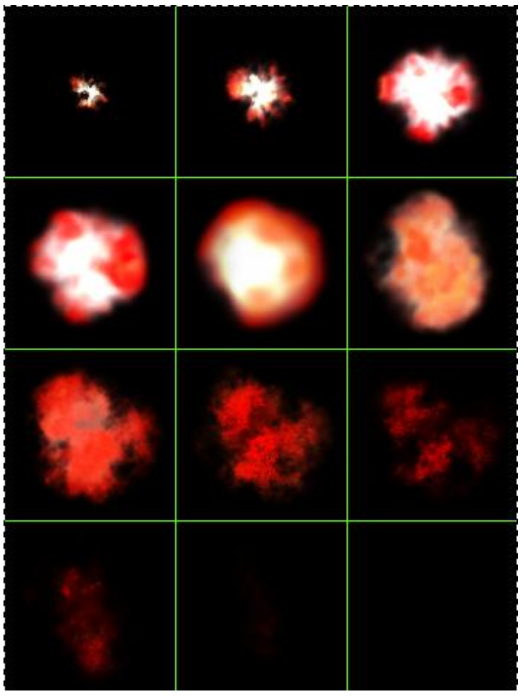
http://drewfx.com/TAFE/XNA/Explosion.png

Go to the Sprite section of the Content in Solution Explorer...right click on Sprites and Add->Existing Item...

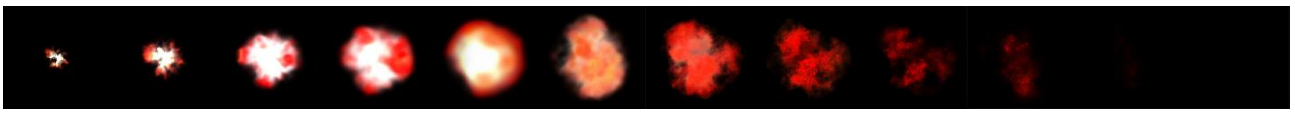


Select the Explosion.png sprite you just downloaded...

I've added a grid to show you how it's divided in memory...



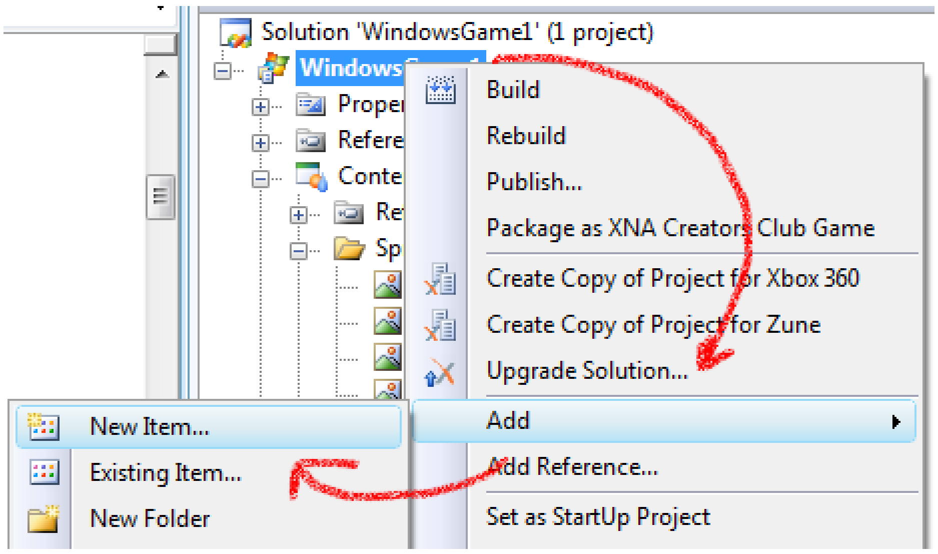
But I wanted to make it easy to animate so I cut and pasted it into one row...



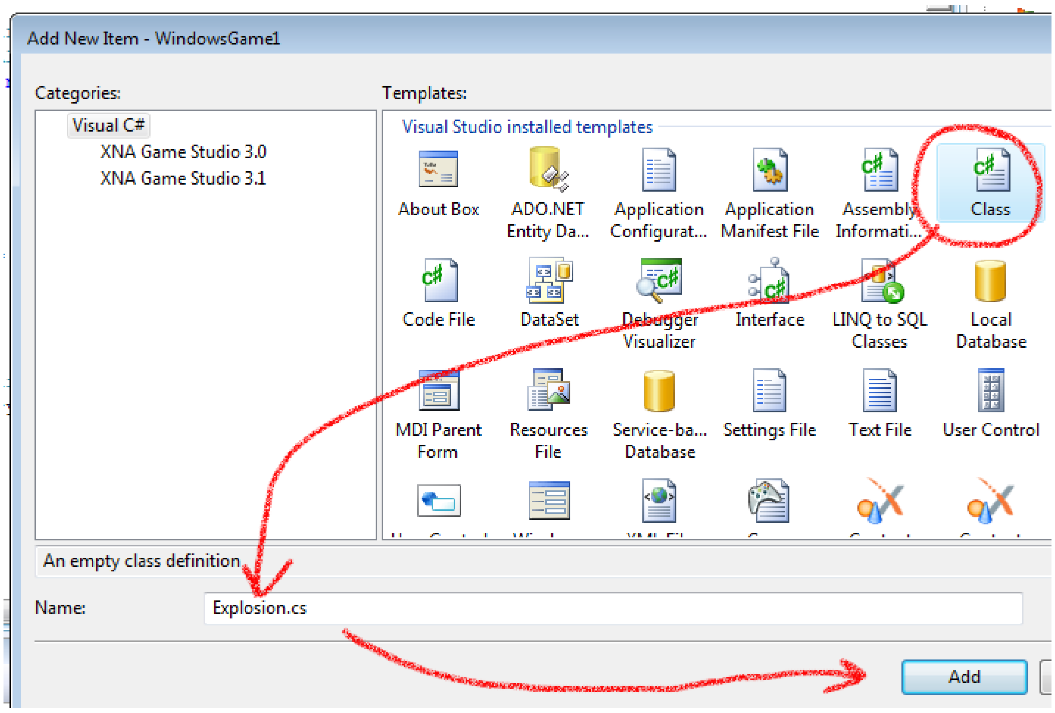
Next, go to Load Content in Game1.cs and add this line...

```
        starSprite = this.Content.Load<Texture2D>("Sprites/starSprite");

        Font1 = Content.Load<SpriteFont>("Arial");

        // Load the font...
        FontPos = new Vector2(graphics.GraphicsDevice.Viewport.Width / 2,
            graphics.GraphicsDevice.Viewport.Height / 2);

        explosionSprite = this.Content.Load<Texture2D>("Sprites/Explosion");

    }
```

We'll need to add a class so we can spawn many explosions. So right click on WindowsGame1 and Add->New Item...



Now select class and name it Explosion.cs...


Add New Item - WindowsGame1

Categories:

Visual C#
XNA Game Studio 3.0
XNA Game Studio 3.1

Templates:

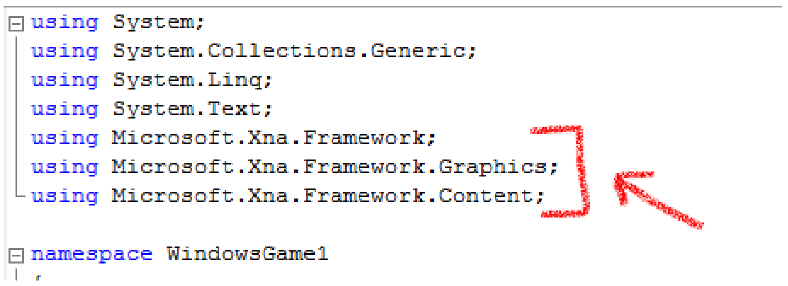
Visual Studio installed templates

An empty class definition

Name: Explosion.cs

Explosion is going to need some attributes to work...

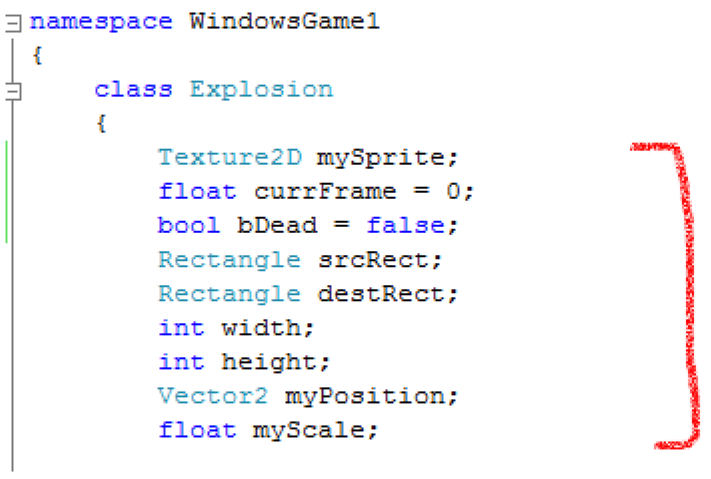
Scroll to the top of Explosion.cs and add this code...



```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Content;

namespace WindowsGame1
```

This is so we can access the XNA Framework.

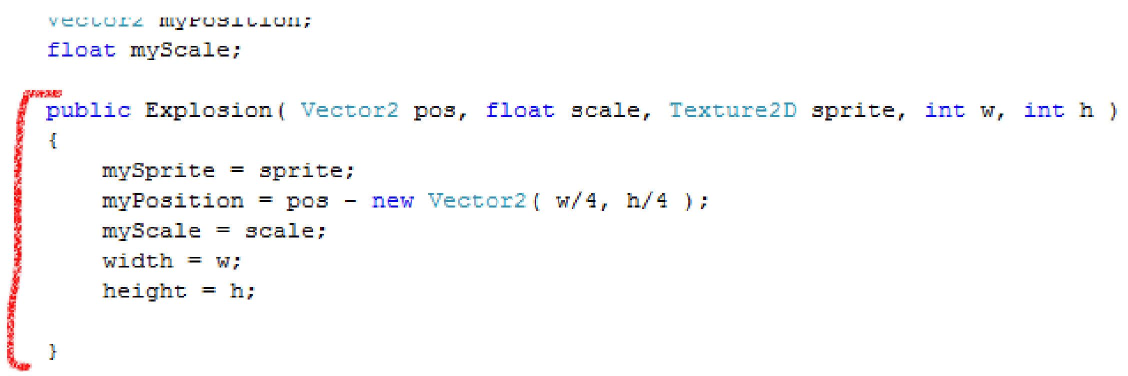
Then scroll down a bit and add this code...



```
namespace WindowsGame1
{
    class Explosion
    {
        Texture2D mySprite;
        float currFrame = 0;
        bool bDead = false;
        Rectangle srcRect;
        Rectangle destRect;
        int width;
        int height;
        Vector2 myPosition;
        float myScale;
```

These are all the attributes we'll need to control the explosions.

Now we need to write a constructor, so after that code add this...



```
Vector2 myPosition;
float myScale;

public Explosion( Vector2 pos, float scale, Texture2D sprite, int w, int h )
{
    mySprite = sprite;
    myPosition = pos - new Vector2( w/4, h/4 );
    myScale = scale;
    width = w;
    height = h;

}
```

This constructor will take a position from the object you want it to appear on top of (the alien), a sprite for it, the scale of it and the width and height of it. This gives a bit of flexibility for different size explosions and different looking explosions.

Let's add a method to test if it's finished its animation sequence, and the animation code.

```
public bool isDead()
{
    return bDead;
}

public void Update()
{
    if (currFrame < 12)
    {
        currFrame += 0.3f;
    }
    else
    {
        bDead = true;
    }
}
```

The 0.3f modifier will animate it a bit slower than the frame rate so it stays around for a while. Finally we'll add the draw method...

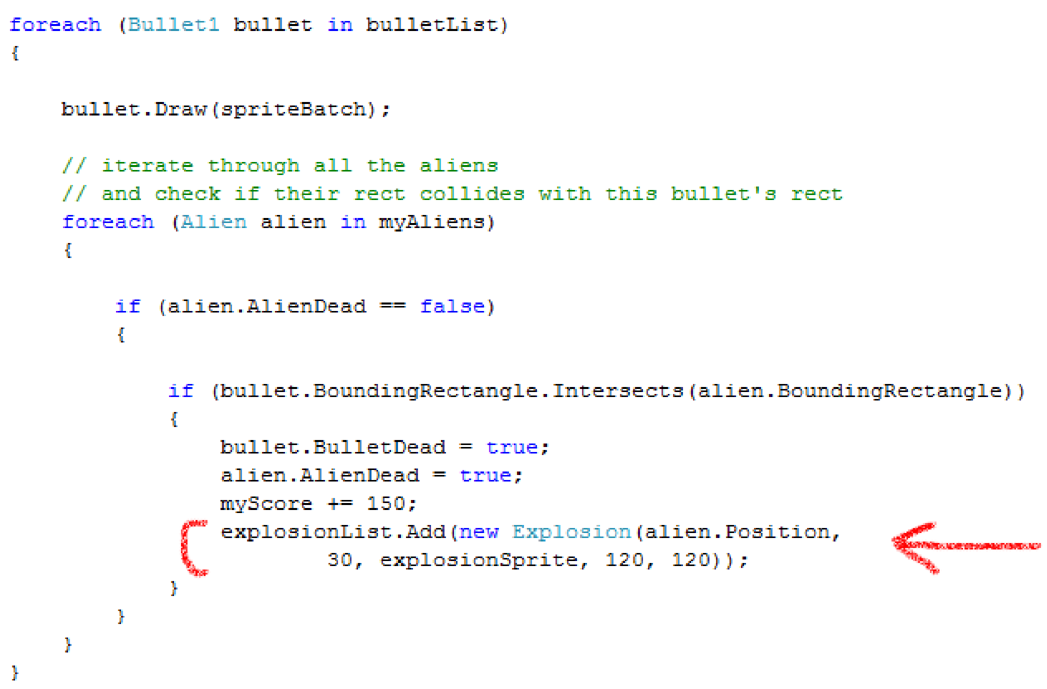
```
public void Draw( SpriteBatch spriteBatch )
{
    srcRect = new Rectangle( (int)Math.Abs(currFrame) * width, 0, width, height );
    destRect = new Rectangle(Convert.ToInt16(myPosition.X),
                             Convert.ToInt16(myPosition.Y),
                             width, height);
    spriteBatch.Draw(mySprite, destRect, srcRect, Color.White);
}
```

This will draw only a small part of the larger texture sheet. The currFrame animates across the frame over time, and gives the impression of a fluid explosion.

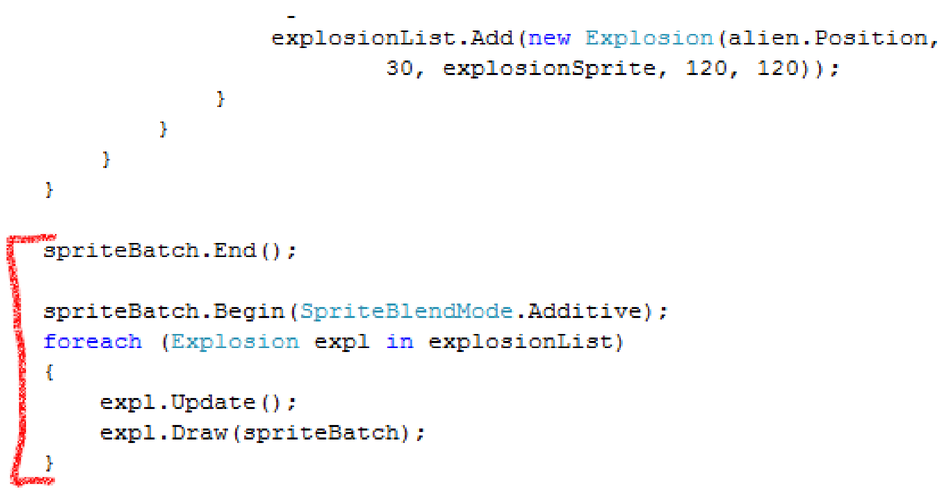
We now need to allow access in Game1.cs file. Go to the top of Game1.cs and add this code...

```
private static Texture2D starSprite;
private static Texture2D explosionSprite;

int screenWidth;
int screenHeight;

List<Bullet1> bulletList = new List<Bullet1>();

List<Explosion> explosionList = new List<Explosion>();
```

Go down to the Draw method and find the loop through the bullets. Change the code as indicated...


```
foreach (Bullet1 bullet in bulletList)
{

    bullet.Draw(spriteBatch);

    // iterate through all the aliens
    // and check if their rect collides with this bullet's rect
    foreach (Alien alien in myAliens)
    {

        if (alien.AlienDead == false)
        {

            if (bullet.BoundingRectangle.Intersects(alien.BoundingRectangle))
            {
                bullet.BulletDead = true;
                alien.AlienDead = true;
                myScore += 150;
                explosionList.Add(new Explosion(alien.Position,
                        30, explosionSprite, 120, 120));
            }
        }
    }
}
```

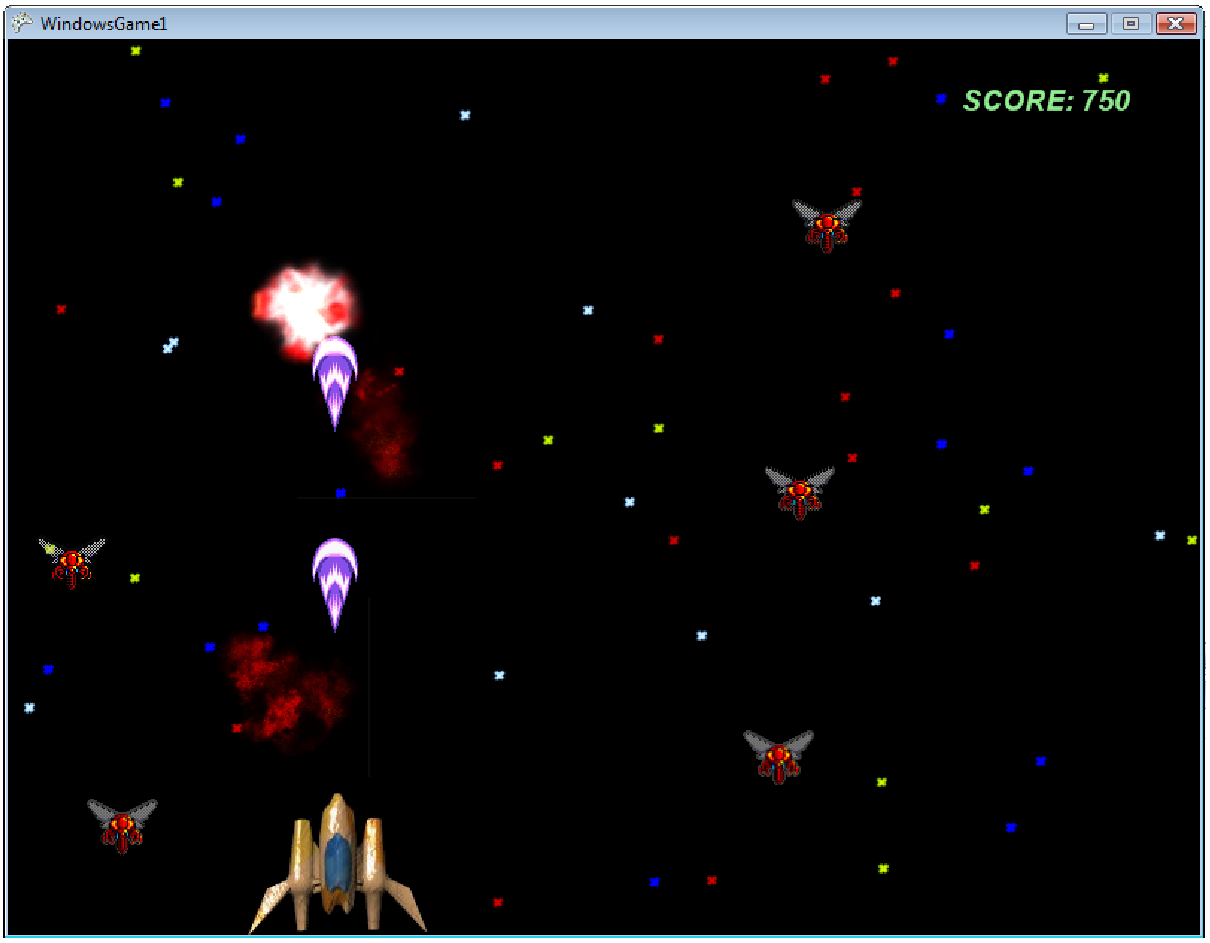
This works like bullets. We just add one as we like, and in this case we add one when the alien dies. We need to draw it next. We want to overlay it with transparency and an additive blend so we stop the other spritedraw batch and start a new one.


```
                explosionList.Add(new Explosion(alien.Position,
                        30, explosionSprite, 120, 120));
            }
        }
    }
}

spriteBatch.End();

spriteBatch.Begin(SpriteBlendMode.Additive);
foreach (Explosion expl in explosionList)
{
    expl.Update();
    expl.Draw(spriteBatch);
}
```

Finally like the bullets we need to clean them up when they have finished animating. Just under where the bullets are cleaned up, add this code...

```
    cnt += 1;
}
// cleanup explosions
START2:
cnt = 0;
foreach (Explosion expl in explosionList)
{

    if (expl.isDead())
    {
        explosionList.RemoveAt(cnt);
        goto START2;
    }
    cnt += 1;
}
```

Okay run it. CTRL+F5. When you shoot the aliens they will leave an impressive explosion.



Level Control

On to the last hurdle for this week. We want to be able to control when the level is completed so we need to keep track on how many aliens have been destroyed, show a little message for the end of level, then generate a new one.

Go to the top of Game1.cs and add this.

```
SpriteFont Font1;
Vector2 FontPos;

int myScore = 0;
int level = 1;
float messageCue = 0;
string currentMessage = "Level 1";
int numAliensShot = 0;
```

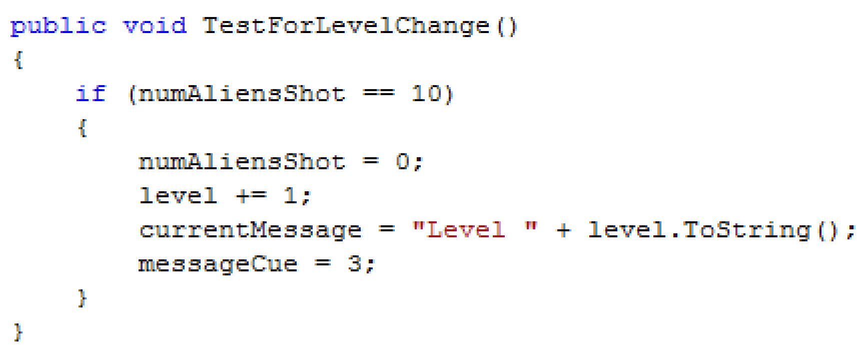
We need these to control the level and to delay the generation of new aliens.

Just below LoadContent add this function. This will handle the messaging system for us, which also controls the level generation.

```
public void HandleMessages(GameTime gameTime)
{


    if (messageCue == 0) return;

    TimeSpan ts = gameTime.ElapsedGameTime;
    messageCue -= (float)ts.TotalSeconds;

    if (messageCue > 0)
    {
        Vector2 pos = new Vector2(screenWidth / 2, screenHeight / 2);

        // Find the center of the string
        Vector2 FontOrigin = Font1.MeasureString(currentMessage) / 2;

        DrawText( currentMessage, pos - FontOrigin);
    }
    else
    {
        messageCue = 0;
        CreateAliens();
    }
}
```

If messageCue is 0 it pops out and lets you shoot the current wave of aliens. If messageCue > 0 then it ticks down based on the time since last update so it is time accurate. It shows the latest message in the middle of the screen, and if messageCue was >0 and ticks down to below zero it generates some more aliens.

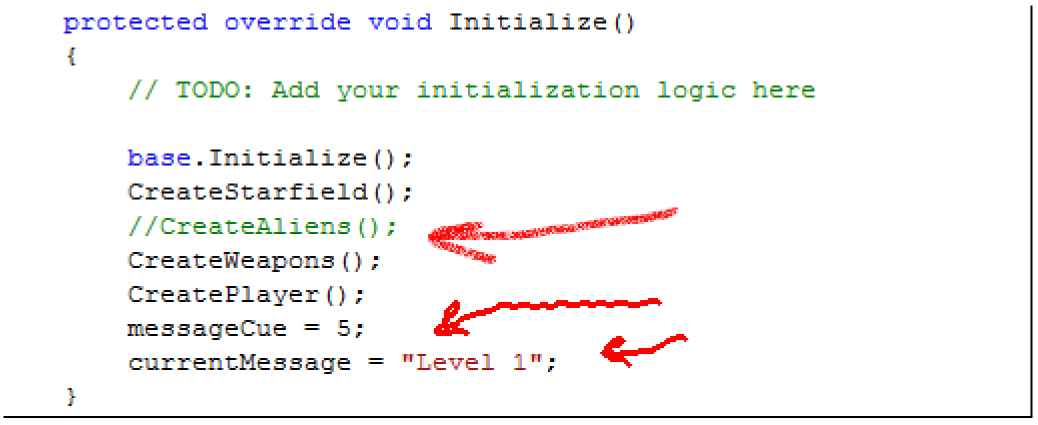
Add this method directly below that one.



```
public void TestForLevelChange()
{
    if (numAliensShot == 10)
    {
        numAliensShot = 0;
        level += 1;
        currentMessage = "Level " + level.ToString();
        messageCue = 3;
    }
}
```

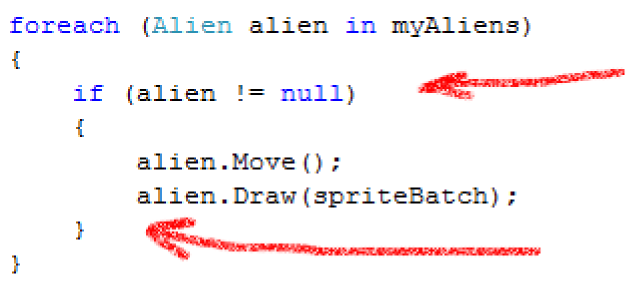
This is called when an alien is shot. If the number of aliens killed is the maximum generated then it's time for a new level. We update the message and show the next message for 3 seconds. We know now that after another 3 seconds it will generate more aliens.

We don't want aliens spawning on Initialize so scroll down to that function and comment out the called to CreateAliens.   Also add the messageCue and currentMessage settings below...



```
protected override void Initialize()
{
    // TODO: Add your initialization logic here

    base.Initialize();
    CreateStarfield();
    //CreateAliens();
    CreateWeapons();
    CreatePlayer();
    messageCue = 5;
    currentMessage = "Level 1";
}
```

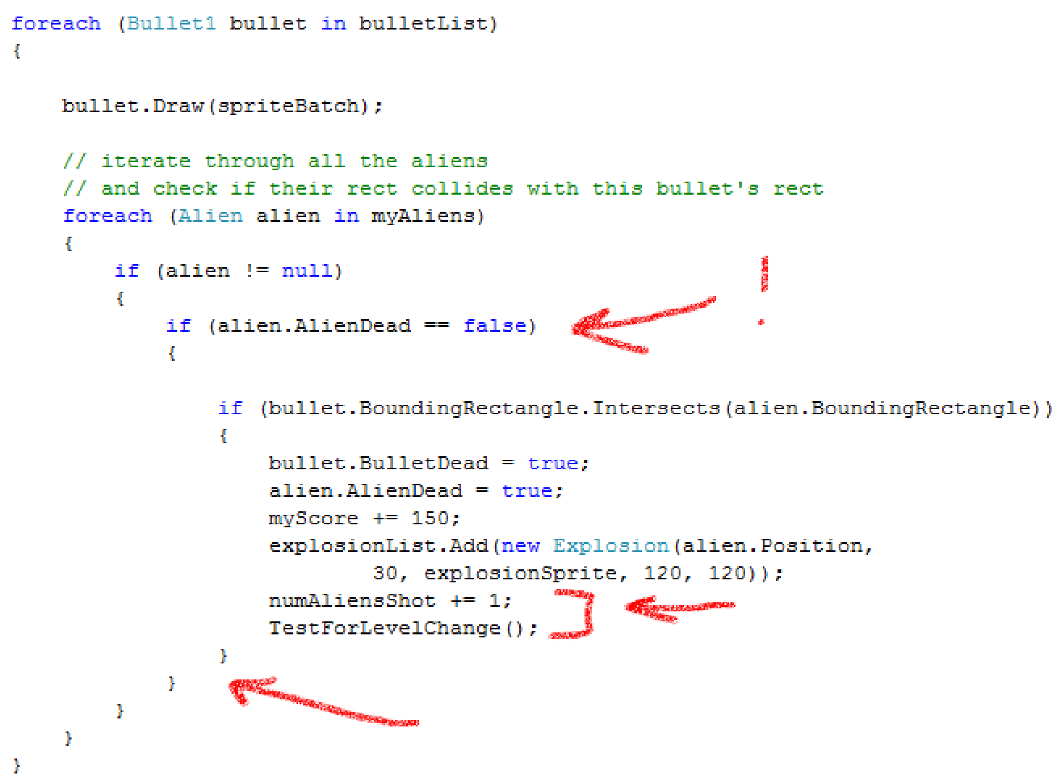
Scroll down to the Draw method, and find where we cycle through the aliens and then change the code...



```
foreach (Alien alien in myAliens)
{
    if (alien != null)
    {
        alien.Move();
        alien.Draw(spriteBatch);
    }
}
```

We do this because there might not be any aliens right now, even though space exists for them.

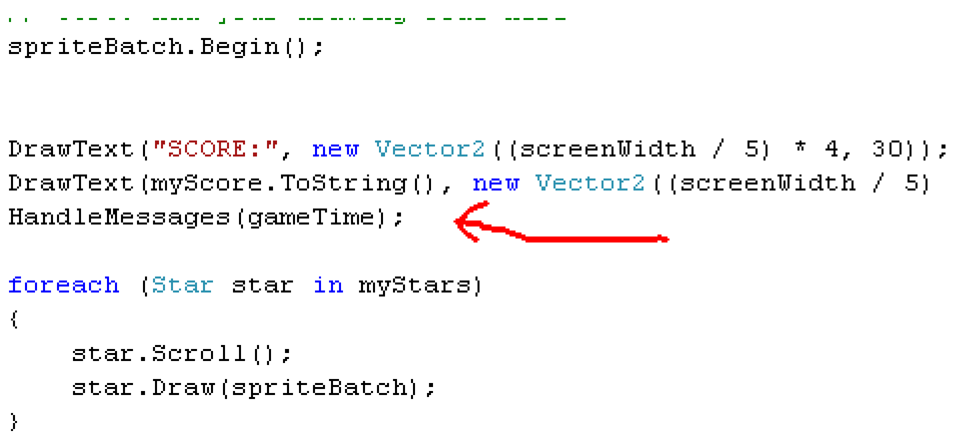
Scroll a bit further down and in the bullet checking code modify the code to look like this...



```
foreach (Bullet1 bullet in bulletList)
{

    bullet.Draw(spriteBatch);

    // iterate through all the aliens
    // and check if their rect collides with this bullet's rect
    foreach (Alien alien in myAliens)
    {
        if (alien != null)
        {
            if (alien.AlienDead == false)
            {

                if (bullet.BoundingRectangle.Intersects(alien.BoundingRectangle))
                {
                    bullet.BulletDead = true;
                    alien.AlienDead = true;
                    myScore += 150;
                    explosionList.Add(new Explosion(alien.Position,
                            30, explosionSprite, 120, 120));
                    numAliensShot += 1;
                    TestForLevelChange();
                }
            }
        }
    }
}
```

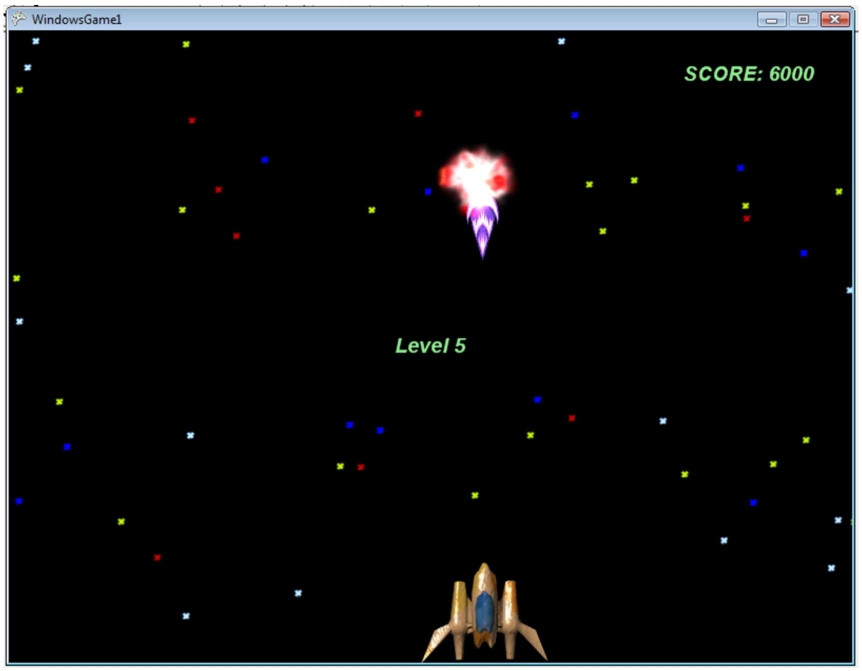
Again we need to check the aliens exists before testing bullets against them. Make sure you add the containing brace at the end as pointed out. Also add the increment of numAliensShot and the call to TestForLevelChange to see if it's the end of the level.

Go to the Draw method and below the two DrawText calls add the following code...



```
spriteBatch.Begin();


DrawText("SCORE:", new Vector2((screenWidth / 5) * 4, 30));
DrawText(myScore.ToString(), new Vector2((screenWidth / 5)
HandleMessages(gameTime);

foreach (Star star in myStars)
{
    star.Scroll();
    star.Draw(spriteBatch);
}
```

F6 to compile, CTRL+F5 to run. You will be able to shoot levels and levels of aliens. Yes the waves are the same, but it's the core functionality for a scrolling shooter. Well done – it's time to play – well actually go back to work on the 2D tutorials please.


WindowsGame1
SCORE: 6000
Level 5