

ECE4893A/CS4803MPG: MULTICORE AND GPU PROGRAMMING FOR VIDEO GAMES



Walkthrough of an XNA 2D Game

Prof. Aaron Lanterman

(Based on slides by Prof. Hsien-Hsin Sean Lee)

School of Electrical and Computer Engineering

Georgia Institute of Technology



2D games

- Using “Sprites”
 - All textures
 - Simple to make or obtain
- Early games before the 3D revolution
 - Space Invaders, Lode Runner, Donkey Kong, Pac-Man
- Do not require high performance accelerators
- Simple enough for your grandparents to enjoy
- Easy to do using XNA framework
- No “effect” (.fx) used

Prof. Lee's game: DawgShower

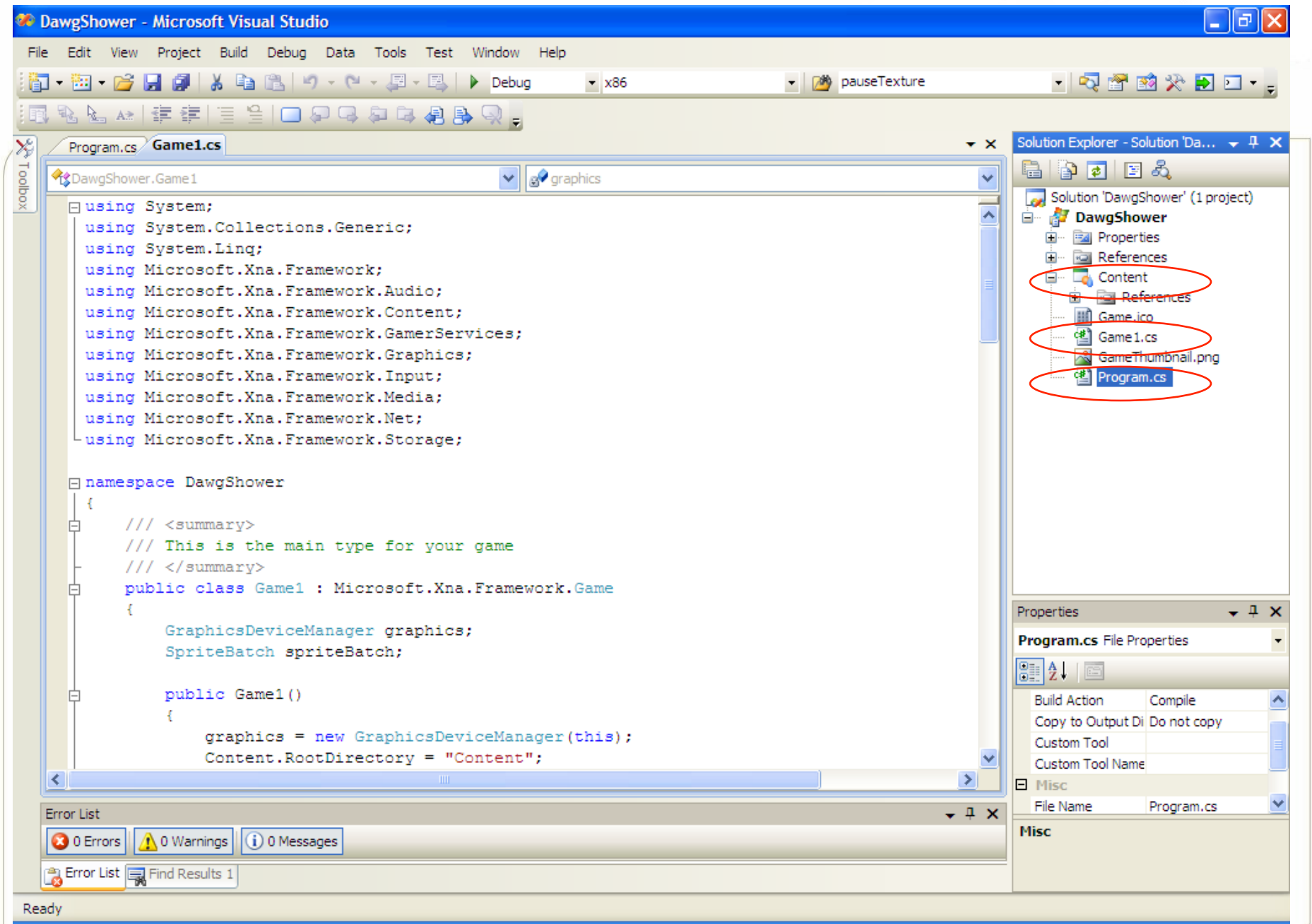
- Shoot the bad dawgs!
- Consist of four main objects
 - The shooter (ship.cs)
 - The bad dawgs (meteros.cs)
 - The missile (missile.cs)
 - Music (AudioComponent.cs)
- The moving objects are all made of sprites

Adapted from Chapter 3 of A. Lobao, B. Evangelista, and J.A. Leal de Farias,
“XNA 2.0 Game Programming: From Novice to Professional”

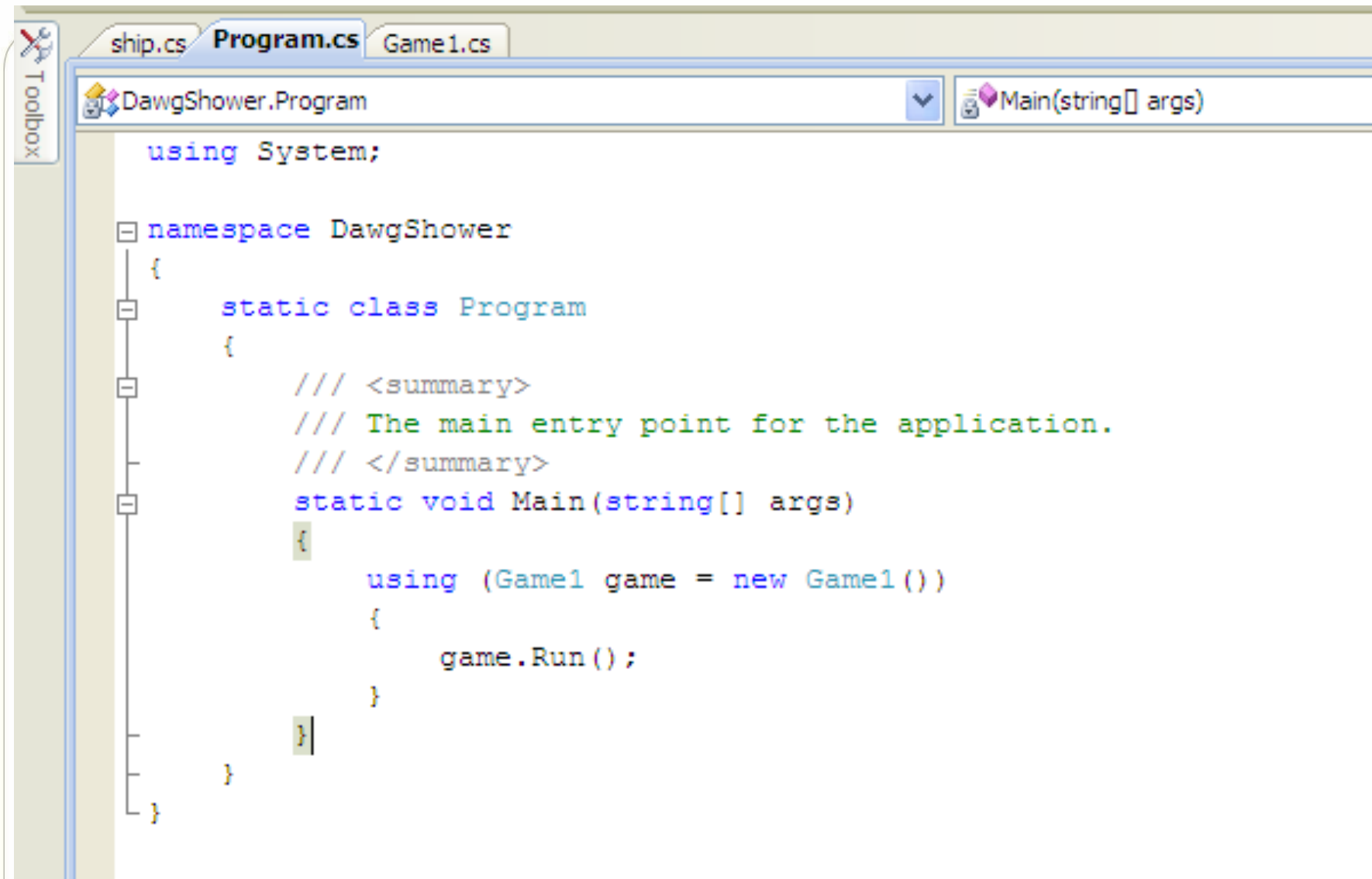
Screenshot



Demo DawgShower Game Example



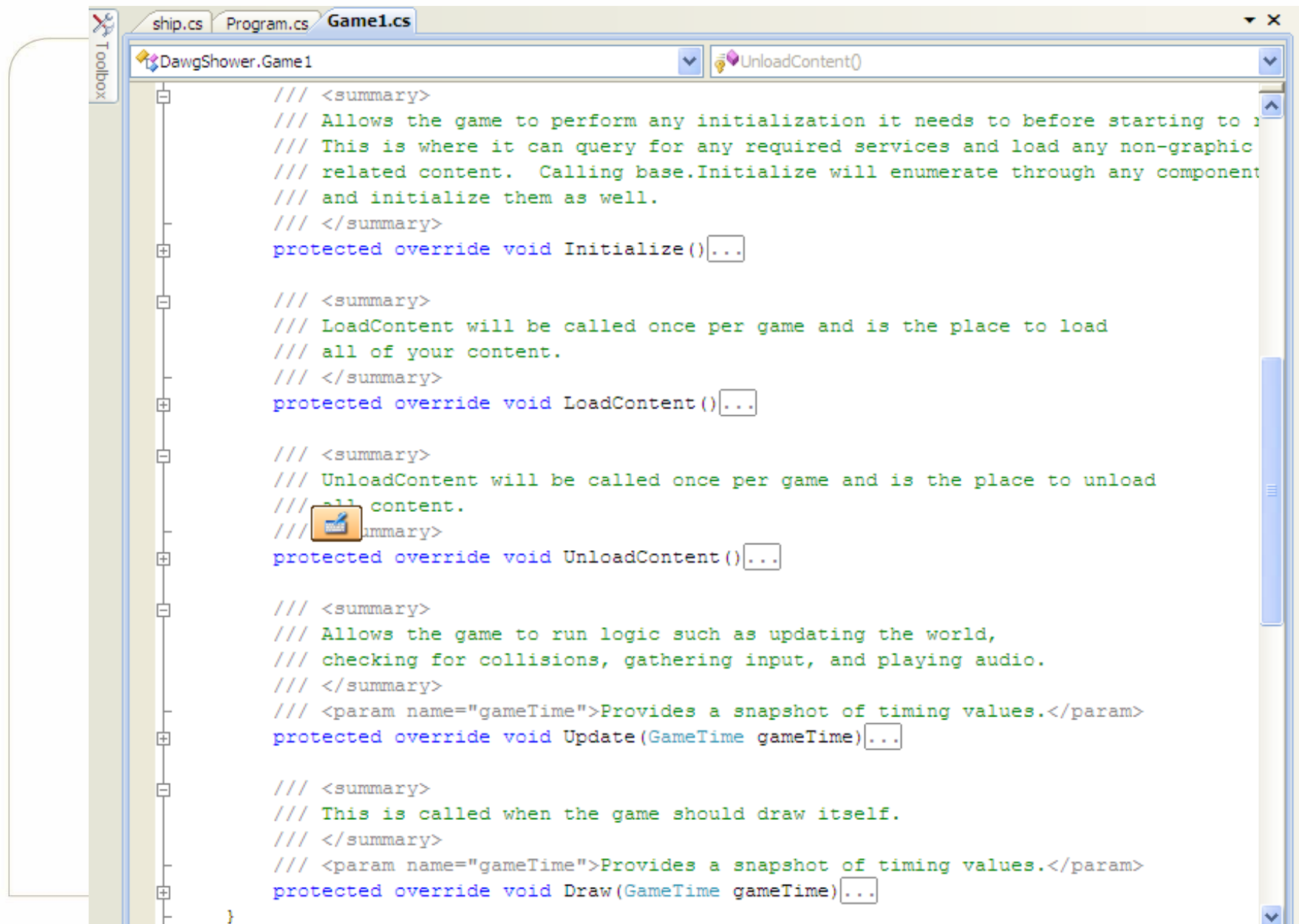
Top level program (Program.cs)



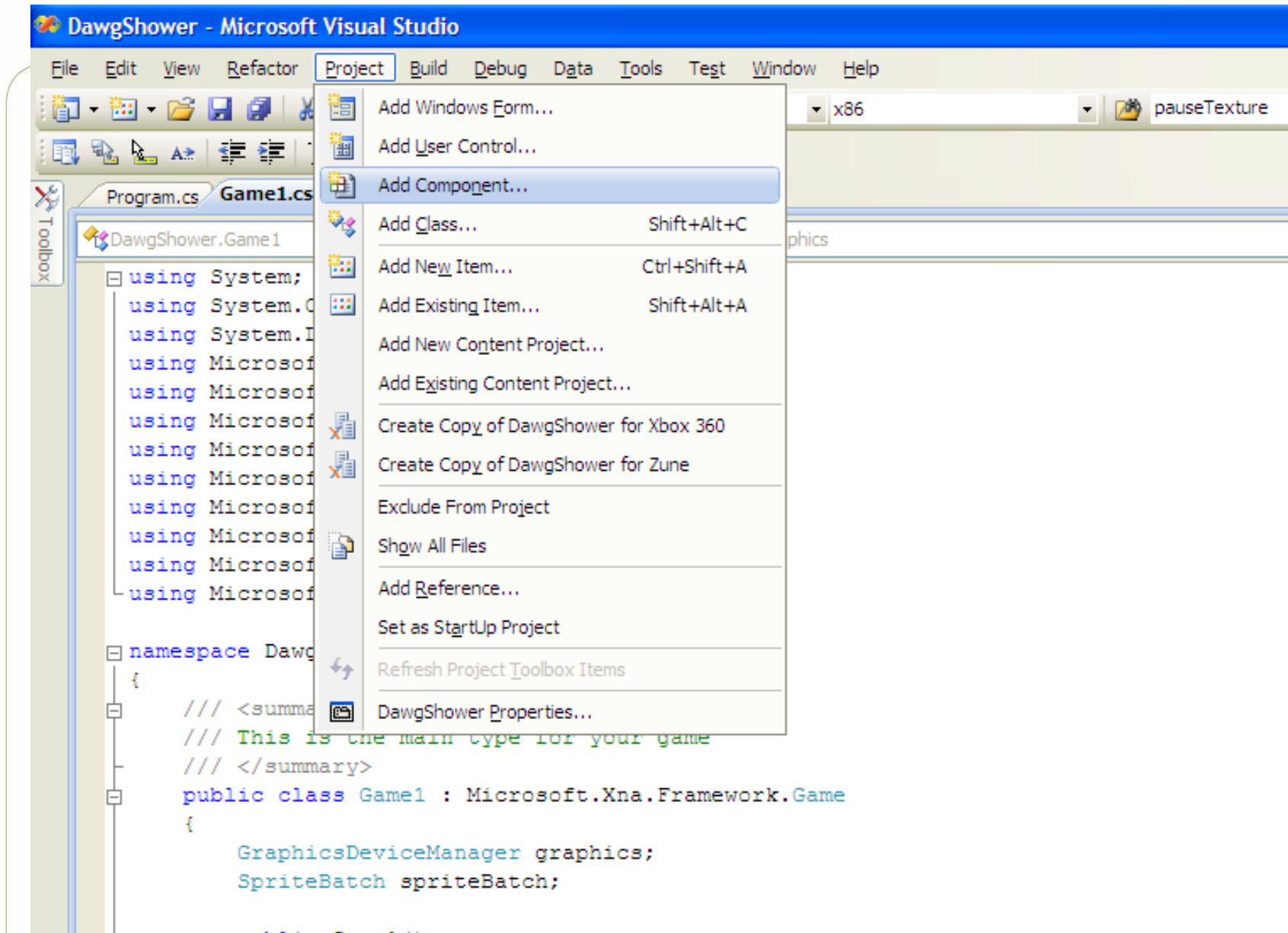
```
using System;

namespace DawgShower
{
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        static void Main(string[] args)
        {
            using (Game1 game = new Game1())
            {
                game.Run();
            }
        }
    }
}
```

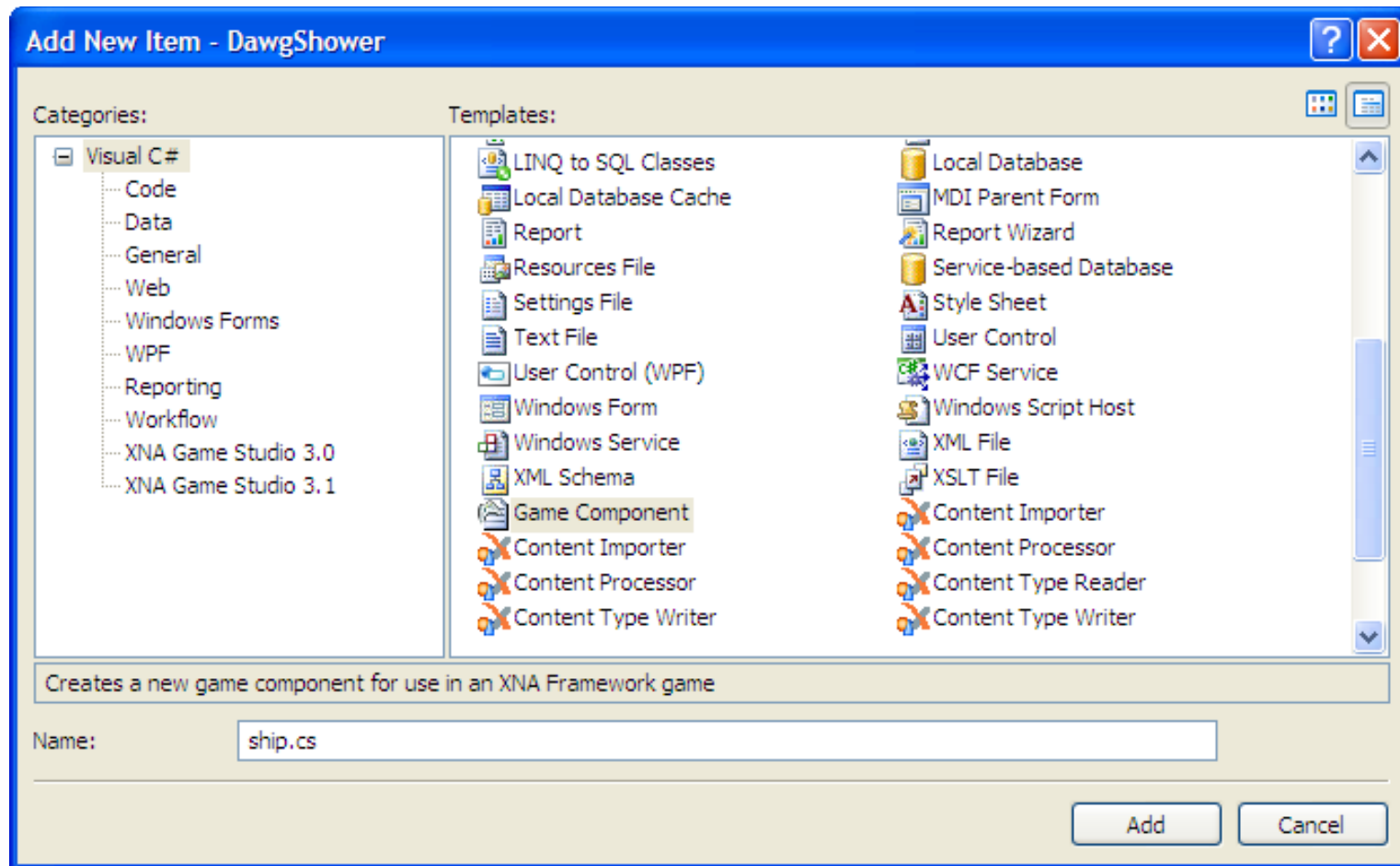

Main game loop (Game1.cs)



Add Game Component

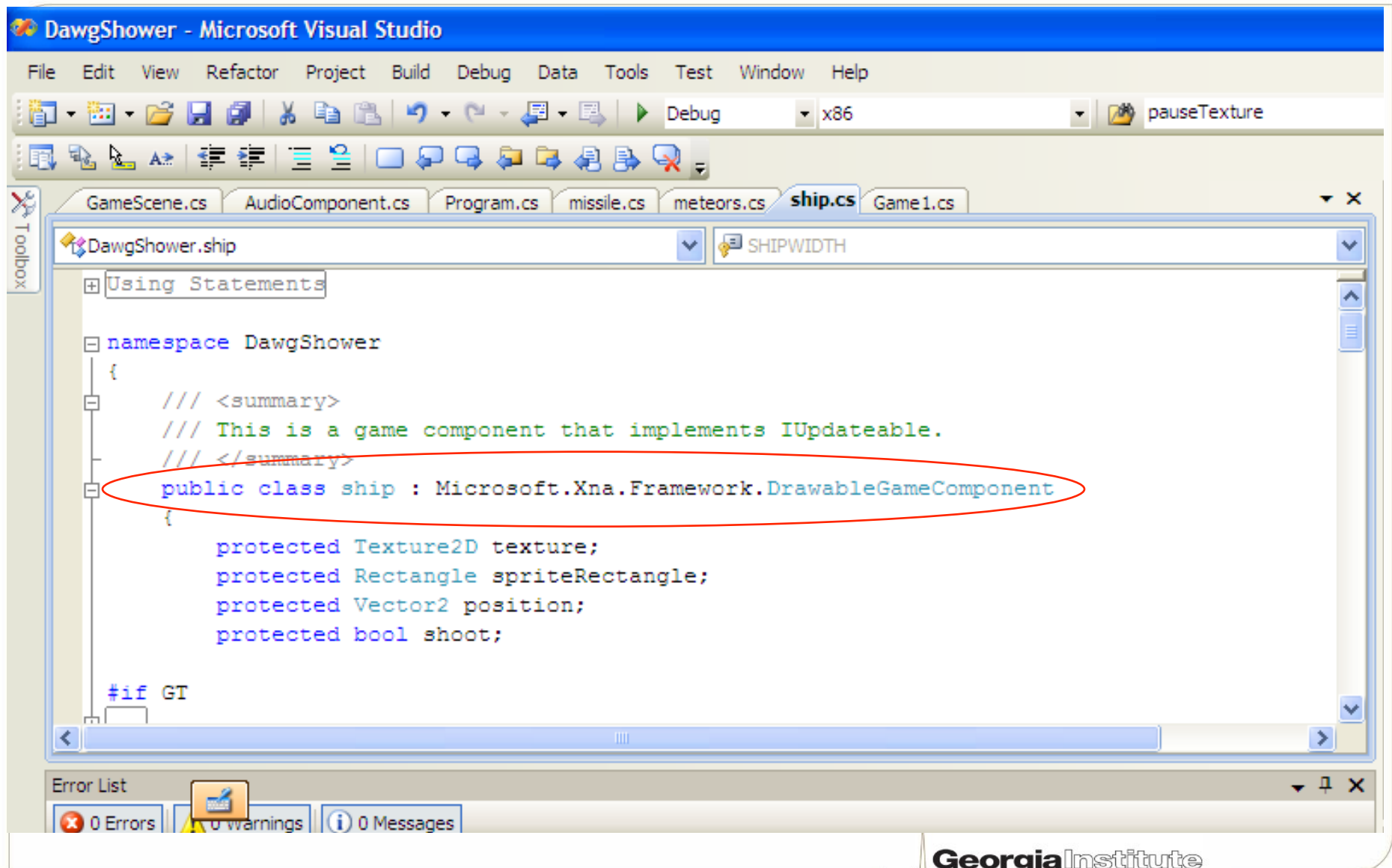


Adding Game Component (C# src file)



- Adding new **DrawableGameComponent** class of objects in the game
 - This class loads and draws graphics content
- Project → Add Components

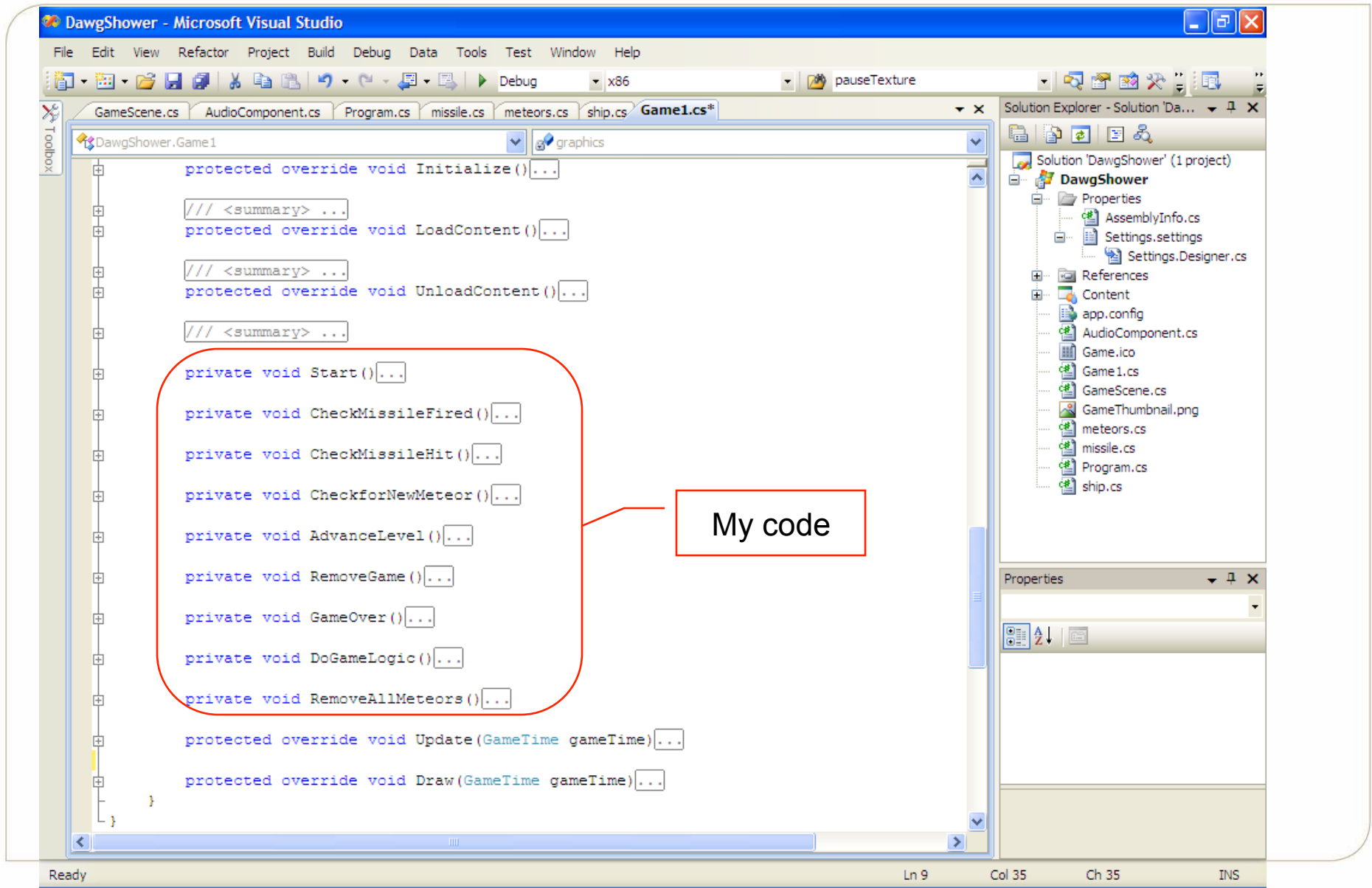
DrawableGameComponent class (1)



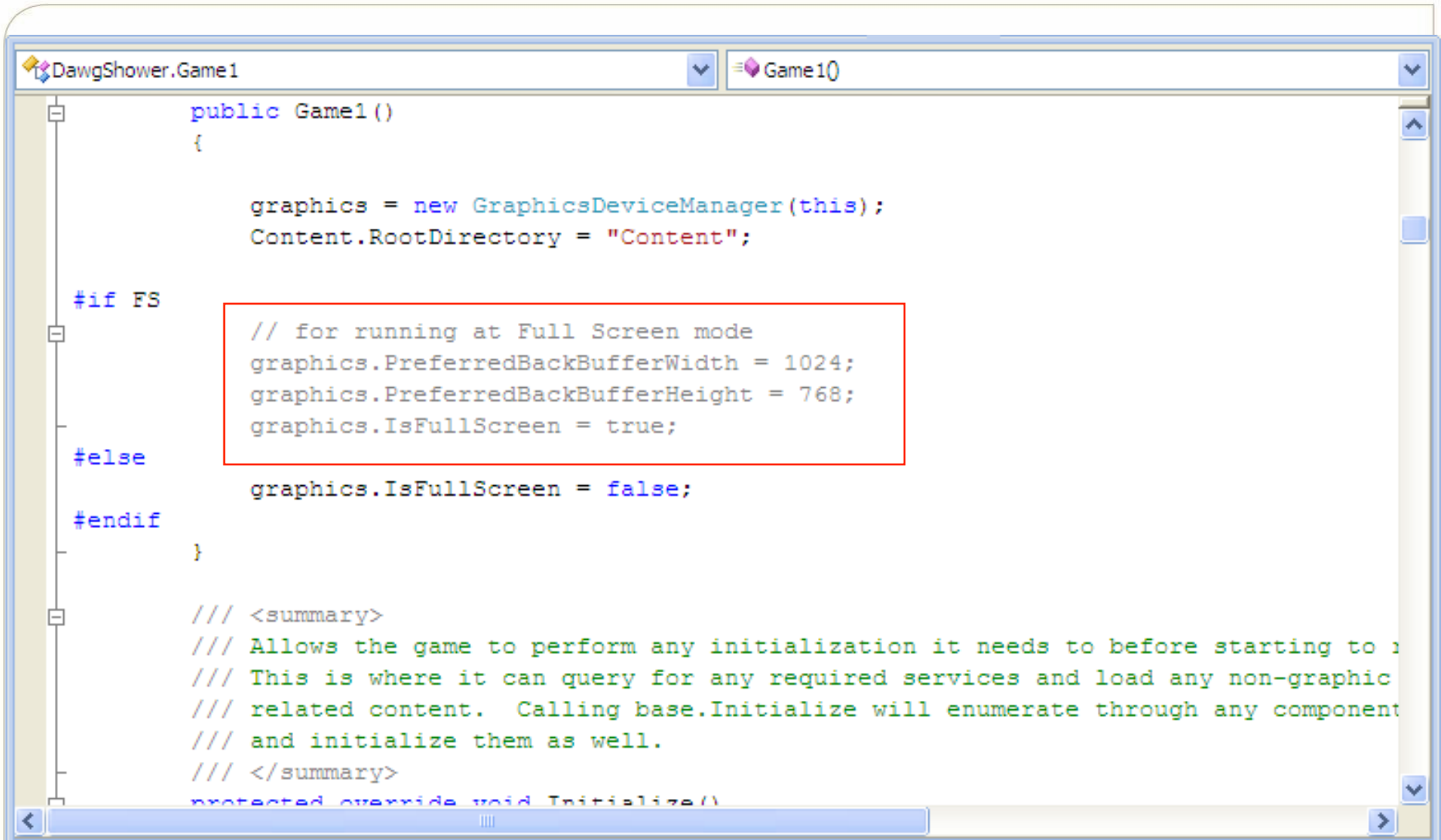
DrawableGameComponent class (2)

- Inherits:
 - GameComponent.Initialize
 - GameComponent.Update
 - DrawableGameComponent.Draw
- Base.draw() method will go through all DrawableGameComponents to execute their respective draw() call

Main program in Game1.cs



Full screen mode (3.1)



```
public Game1()
{
    graphics = new GraphicsDeviceManager(this);
    Content.RootDirectory = "Content";

    #if FS
        // for running at Full Screen mode
        graphics.PreferredBackBufferWidth = 1024;
        graphics.PreferredBackBufferHeight = 768;
        graphics.IsFullScreen = true;
    #else
        graphics.IsFullScreen = false;
    #endif
}

/// <summary>
/// Allows the game to perform any initialization it needs to before starting to
/// This is where it can query for any required services and load any non-graphic
/// related content. Calling base.Initialize will enumerate through any component
/// and initialize them as well.
/// </summary>
protected override void Initialize()
```

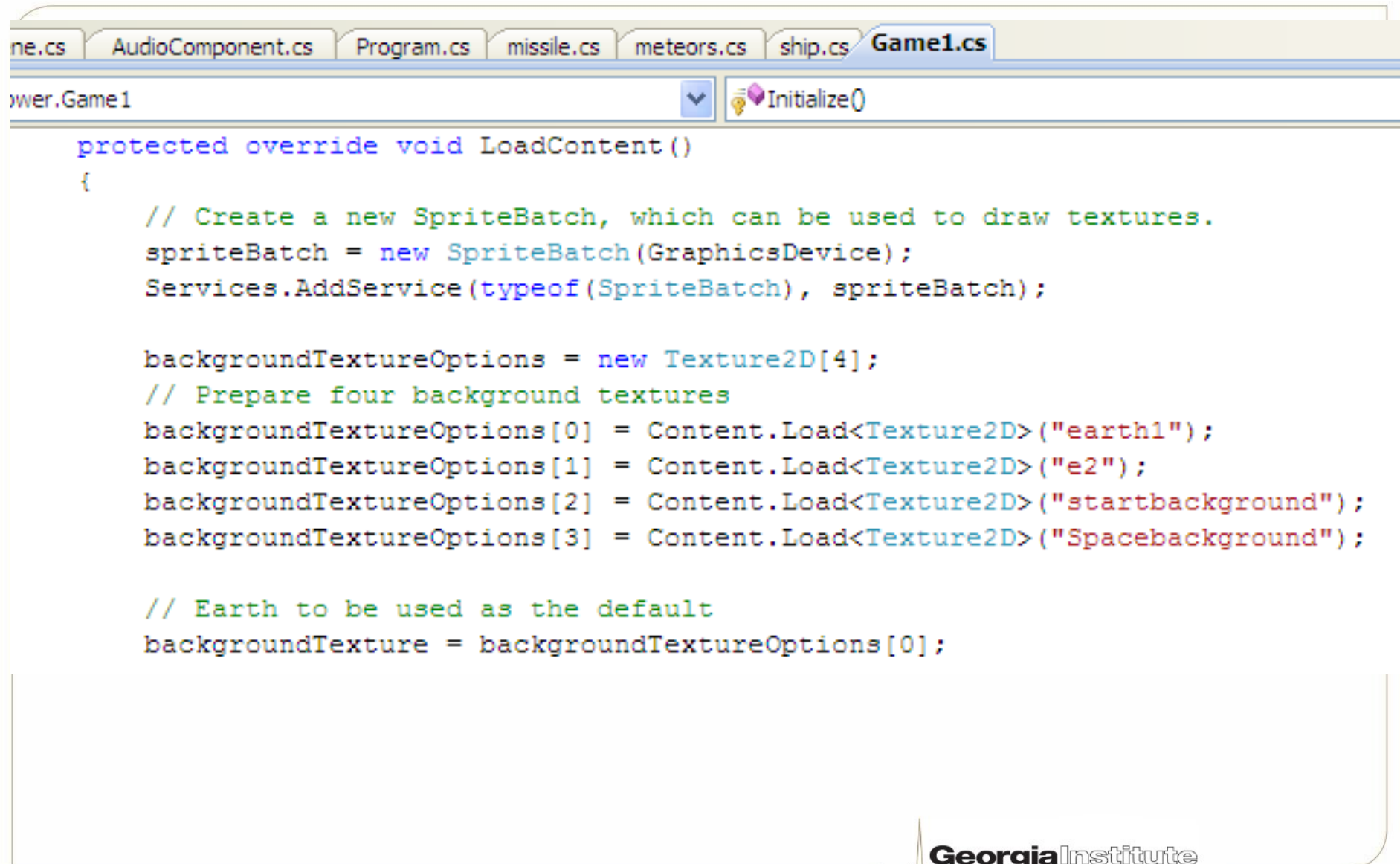
Full screen mode (4.0)

- In porting to XNA 4.0, had to change `DisplayMode.Height` to `PresentationParameters.BackBufferHeight` to get backgrounds to scale correctly
- Similarly for `Width`

Drawing sprites using XNA

- Use “textures”
- Store with XNA’s `Texture2D` class
- Include new texture images into the Content Pipeline
- Use “`Content.Load`” to associate texture variables

Big sprites for the background



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

    backgroundTextureOptions = new Texture2D[4];
    // Prepare four background textures
    backgroundTextureOptions[0] = Content.Load<Texture2D>("earth1");
    backgroundTextureOptions[1] = Content.Load<Texture2D>("e2");
    backgroundTextureOptions[2] = Content.Load<Texture2D>("startbackground");
    backgroundTextureOptions[3] = Content.Load<Texture2D>("Spacebackground");

    // Earth to be used as the default
    backgroundTexture = backgroundTextureOptions[0];
}
```

Small sprites for game assets

```
// Set up textures for game pause
pauseTexture = Content.Load<Texture2D>("pause");
pause2Texture = Content.Load<Texture2D>("pauseR");
meteorTexture = Content.Load<Texture2D>("Goo2");

#if GT
    leeTexture = Content.Load<Texture2D>("buzz");
    missileTexture = Content.Load<Texture2D>("helmet");
#else
    leeTexture = Content.Load<Texture2D>("leeporN3");
    missileTexture = Content.Load<Texture2D>("leePhantom");
#endif

gameoverTexture = Content.Load<Texture2D>("gameover");
gamefont = Content.Load<SpriteFont>("font");
}
```

Game Services

- Game services maintain loose coupling between objects that need to interact with each other
- Register a “global” **SpriteBatch** for drawing all sprites
- **Draw()** method will look for an active **SpriteBatch** in GameServices
- All **GameComponents** will use this **SpriteBatch**

Game Services – setup & use

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

Services.AddService(typeof(SpriteBatch), spriteBatch);
```

Registering a Game Service in LoadContent()

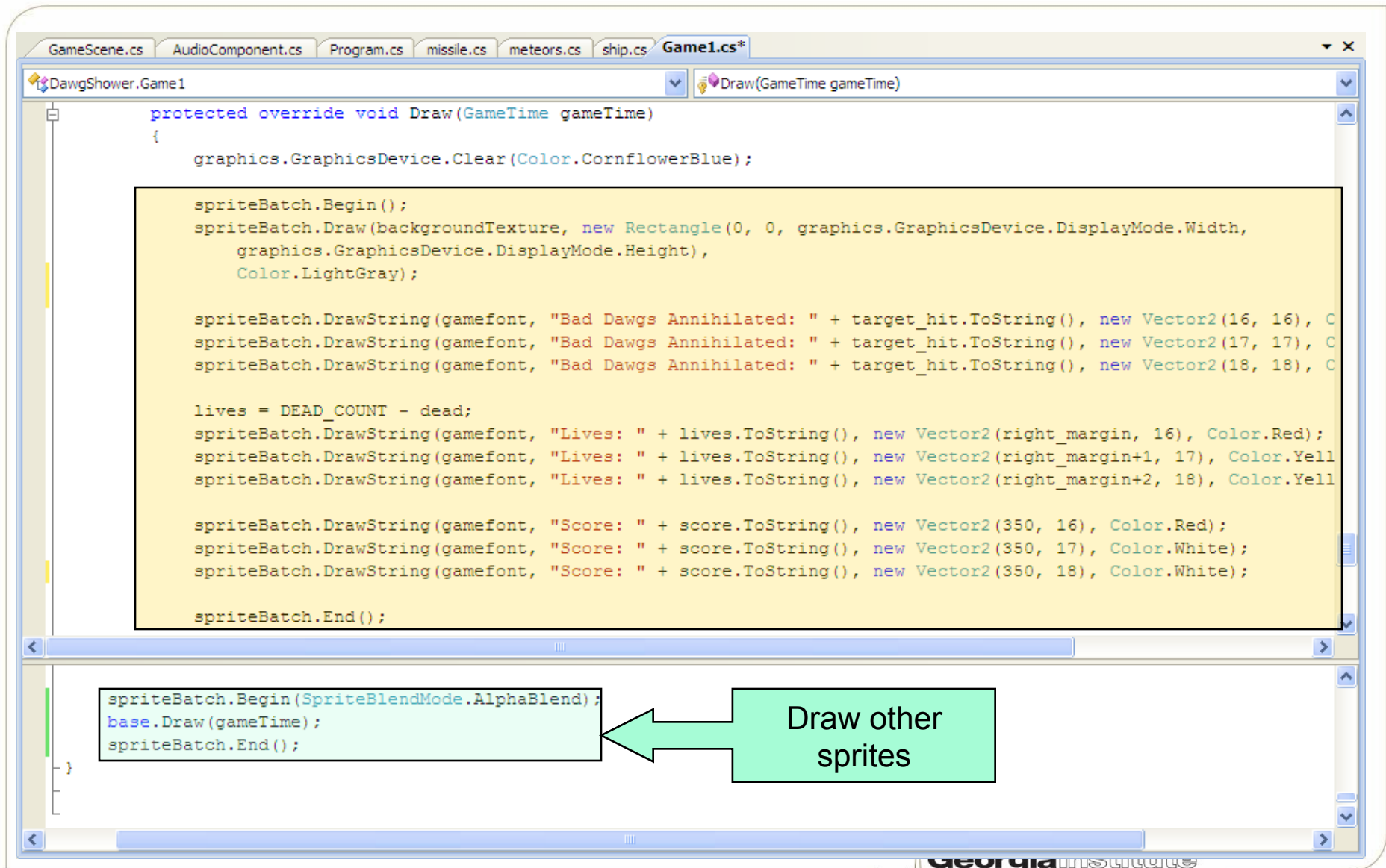
```
public override void Draw(GameTime gameTime)
{
    SpriteBatch sBatch =
        (SpriteBatch)Game.Services.GetService(typeof(SpriteBatch));

    sBatch.Draw(texture, position, spriteRectangle, Color.White);

    base.Draw(gameTime);
}
```

Use GetService to acquire service for each DrawableGameComponent

Drawing the background (3.1)



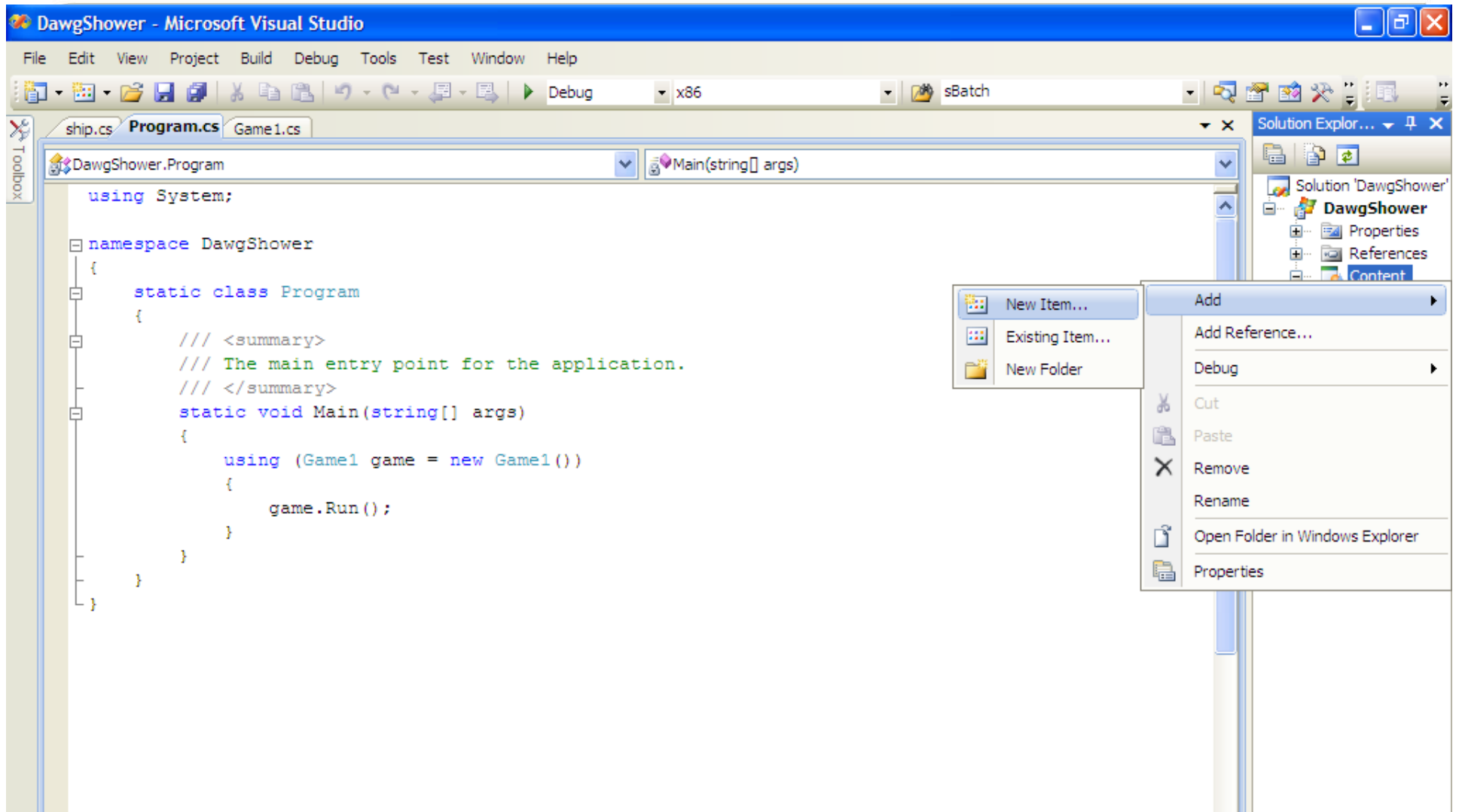
Drawing the background (4.0)

- In porting to XNA 4.0, had to change
`SpriteBlendMode.AlphaBlend`
to
`BlendState.AlphaBlend`

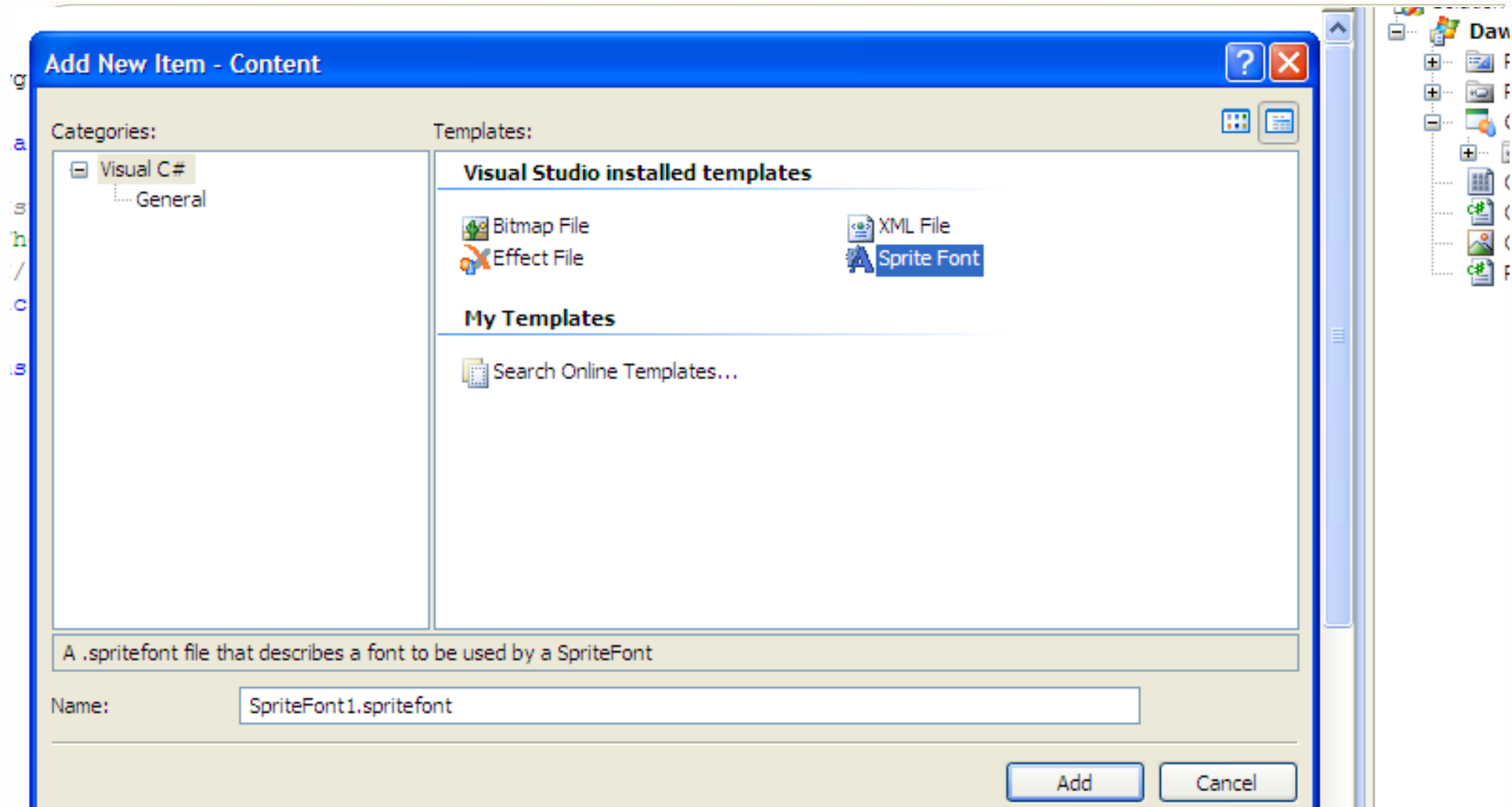
Printing text

- Draw using **SpriteBatch**
- Create a “font sprite”
- Based on available Fonts in the system
- Add font in **LoadContent()**

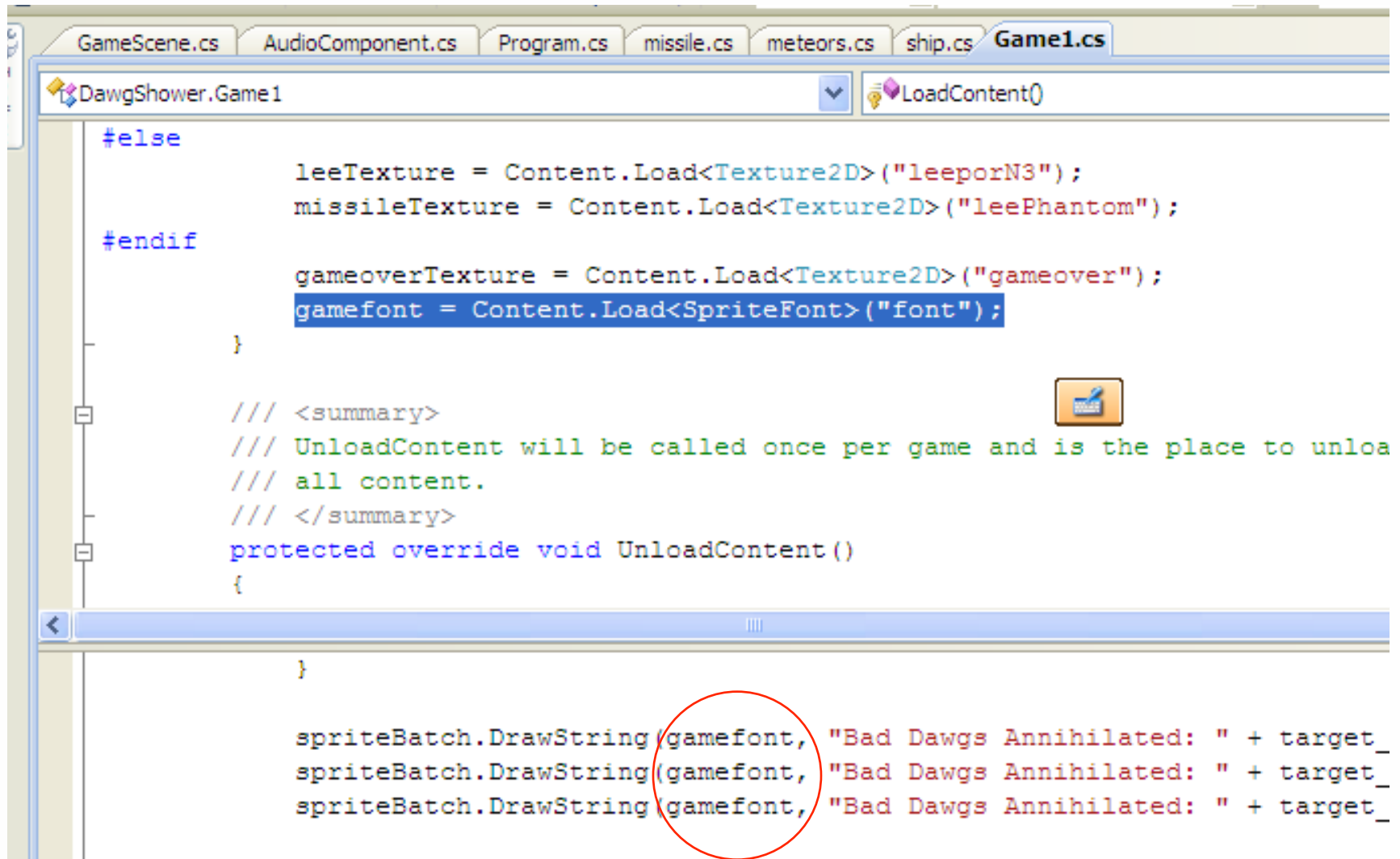
Create a SpriteFont (1)



Create a SpriteFont (2)



DrawString



```
GameScene.cs AudioComponent.cs Program.cs missile.cs meteors.cs ship.cs Game1.cs
DawgShower.Game1 LoadContent()

#else
    leeTexture = Content.Load<Texture2D>("leeporN3");
    missileTexture = Content.Load<Texture2D>("leePhantom");
#endif

    gameoverTexture = Content.Load<Texture2D>("gameover");
    gamefont = Content.Load<SpriteFont>("font");
}

/// <summary>
/// UnloadContent will be called once per game and is the place to unload
/// all content.
/// </summary>
protected override void UnloadContent()
{

}

spriteBatch.DrawString(gamefont, "Bad Dawgs Annihilated: " + target_
spriteBatch.DrawString(gamefont, "Bad Dawgs Annihilated: " + target_
spriteBatch.DrawString(gamefont, "Bad Dawgs Annihilated: " + target_
```

Manage components

- Components: member of `GameComponentCollection` (i.e., `Microsoft.Xna.Framework.Game`)
- Use `Components.Add()` to add a new component to the list
- `Components.RemoveAt(j)` removes j^{th} component

Add sound and the player's ship

```
protected override void Initialize()
{
    audioComponent = new AudioComponent(this);
    Components.Add(audioComponent);
}
```

```
private void Start()
{
    if (player == null)
    {
        player = new ship(this, ref leeTexture);
        Components.Add(player);
    }
}
```

Add missiles, remove meteors

```
private void CheckMissileFired()
{
    // add component is SPACE bar was hit
    if (player.shootMissile())
    {
        audioComponent.PlayCue("shoot");
        Components.Add(new missile(this, ref missileTexture, player.GetPosition()));
        player.resetShoot();
    }
}
```

```
if (Components[j] is meteors)
{
    bool hasCollision = false;
    hasCollision = ((meteors)Components[j]).CheckCollision(((missile)Components[i]).GetBounds());
    if (hasCollision)
    {
        audioComponent.PlayCue("explosion");
        // remove collided meteros (BadDawgs)
        score++;
        Components.RemoveAt(j);
    }
}
```


Game logic

- Embedded inside `Update()` function


```
private void DoGameLogic()
{
    bool hasCollision = false;
    Rectangle shipRectangle = player.GetBounds();
    foreach (GameComponent gc in Components)
    {
        if (gc is meteors)
        {
            hasCollision = ((meteors)gc).CheckCollision(shipRectangle);
            if (hasCollision)
            {
                audioComponent.PlayCue("missile");
                score -= PENALTY;
                dead++;
                RemoveAllMeteors();
                Start();
                break;
            }
        }
    }

    CheckforNewMeteor();
    CheckMissileFired();
    CheckMissileHit();
    AdvanceLevel();
}
```

Pause the game

```
if (!pause && keyboard.IsKeyDown(Keys.P))
{
    pause = true;
}
else if (pause && keyboard.IsKeyDown(Keys.Tab))
{
    pause = false;
}

// Check if Game is over
GameOver();

if (pause == false && ! over)
{
    DoGameLogic();
    base.Update(gameTime);
}
```

Built-in test for collision detection

- Test bounding boxes of given rectangular sprites
- Return a Boolean result

```
public bool CheckCollision(Rectangle rect)
{
    Rectangle spriterect = new Rectangle((int)position.X, (int)position.Y,
                                         MISSILEWIDTH, MISSILEHEIGHT);
    return spriterect.Intersects(rect);
}
```

Missile's
position



BoundingSphere

- Alternative method (more often used for 3-D games)

```
public BoundingSphere (  
    Vector3 center,  
    float radius  
)
```

```
public bool Intersects (  
    BoundingSphere sphere  
)
```

Sound – simple, easy way (1)

- Can use .wav, .mp3, and .wma
- SoundEffect Class
 - Quick vocalizations, door knocks, etc.
- Song Class
 - Background music
- MediaPlayer Class
 - Can pause, resume, skip around, change volume, etc.
 - Access user's music library on WMP, Xbox 360, and Zune

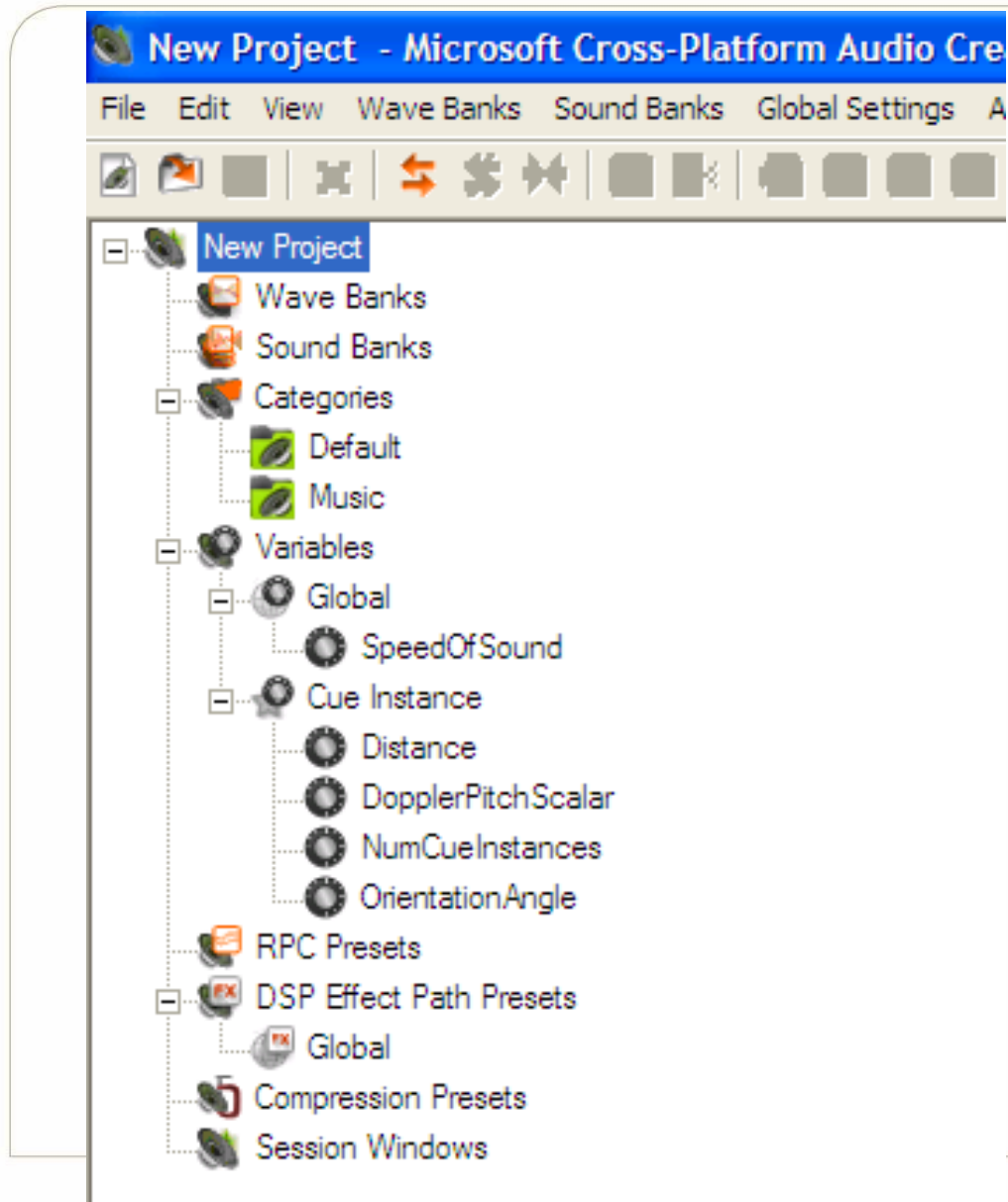
Sound – simple, easy way (2)

- Drag sound files into Content folder
- Load through Content Pipeline

```
SoundEffect soundEffect =  
    Content.Load<SoundEffect> (@"Content\Audio\LaserShot");  
  
soundEffect.Play();
```

Example from <http://xna-uk.net/blogs/offbyone/archive/2010/01/21/sound-in-xna-3-1-part-i.aspx>

Sound – powerful, complex way



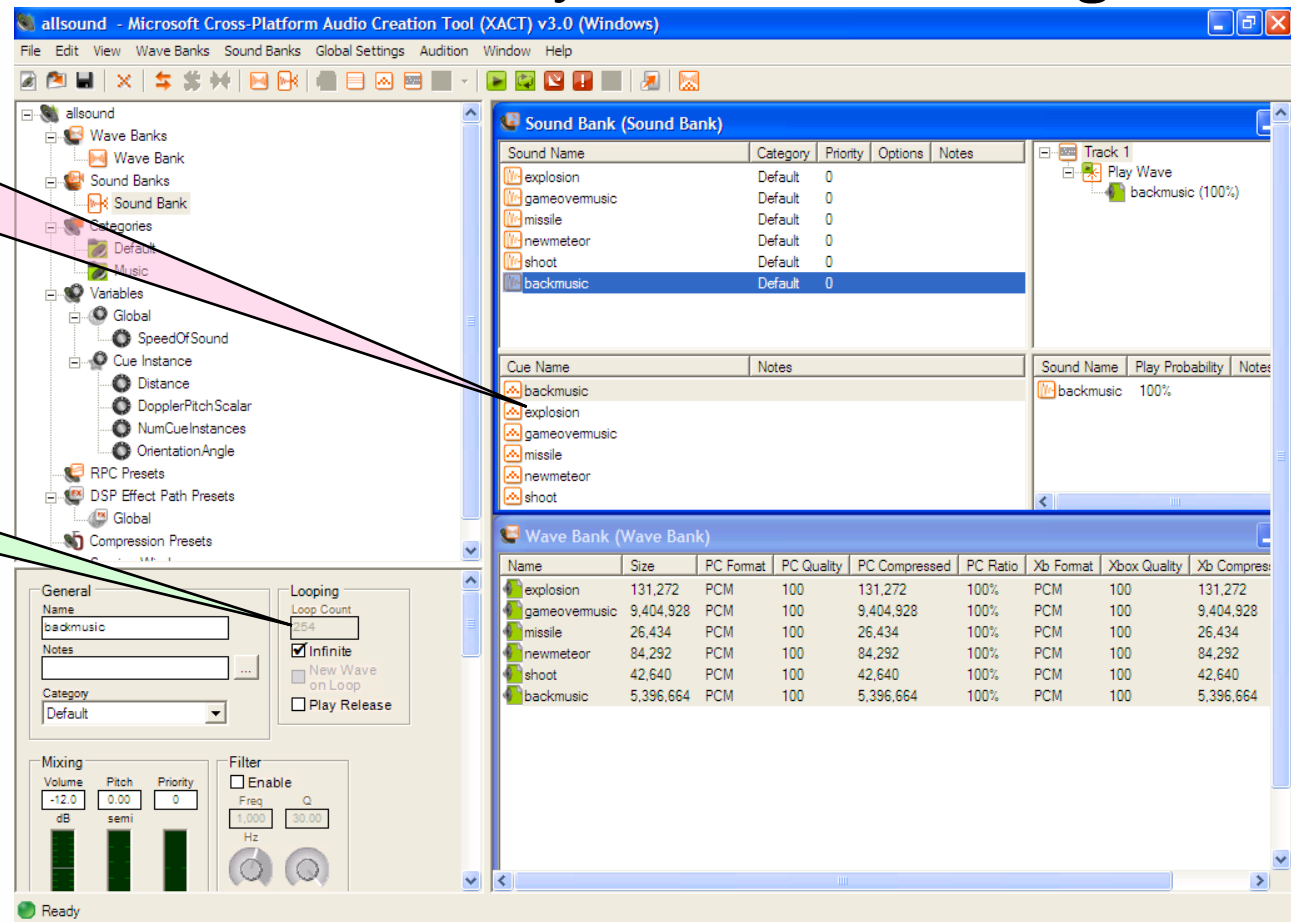
- XACT:
Microsoft
Cross-Platform
Audio Creation
Tool
- Only way to do
sound before
XNA 3.0!

Using XACT

- Create wave banks and sound banks
- Compile them into XAP file used by the content manager

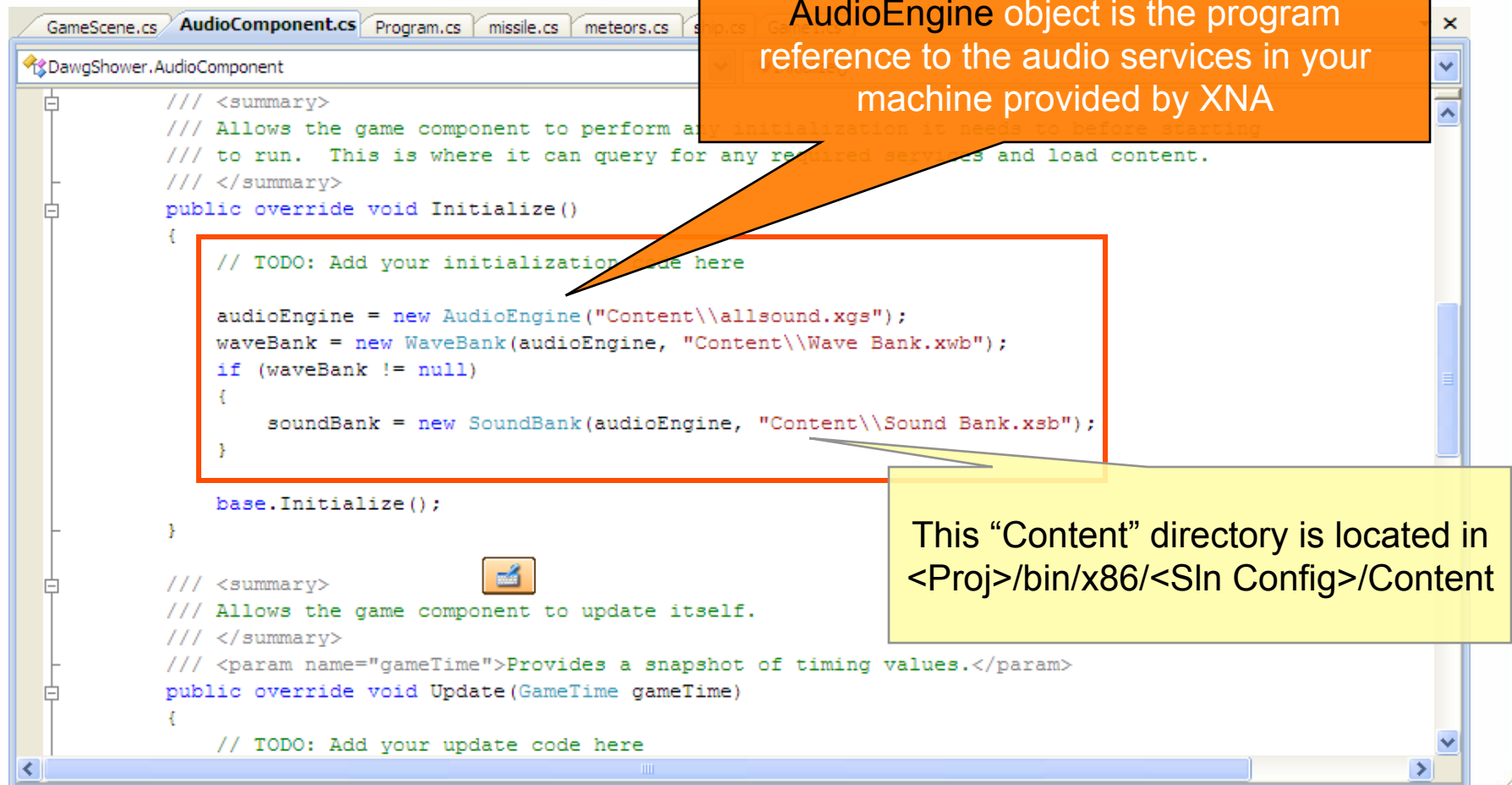
Copy and make sound cues by dragging the files into this window

“checked” for looping background music



Using files created by XACT in XNA

- Create a new **GameComponent** for audio
- Initialize WaveBank and SoundBank in the C# code



The screenshot shows the Visual Studio IDE with the `AudioComponent.cs` file open. The file is part of the `DawgShower.AudioComponent` project. The code includes XML documentation comments and two methods: `Initialize()` and `Update()`. The `Initialize()` method is highlighted with a red box, and the `Update()` method is highlighted with a yellow box. An orange callout box points to the `audioEngine` variable, and a yellow callout box points to the `Content` directory path in the code.

```
/// <summary>
/// Allows the game component to perform any initialization it needs to before starting
/// to run. This is where it can query for any required services and load content.
/// </summary>
public override void Initialize()
{
    // TODO: Add your initialization code here

    audioEngine = new AudioEngine("Content\\allsound.xgs");
    waveBank = new WaveBank(audioEngine, "Content\\Wave Bank.xwb");
    if (waveBank != null)
    {
        soundBank = new SoundBank(audioEngine, "Content\\Sound Bank.xsb");
    }

    base.Initialize();
}

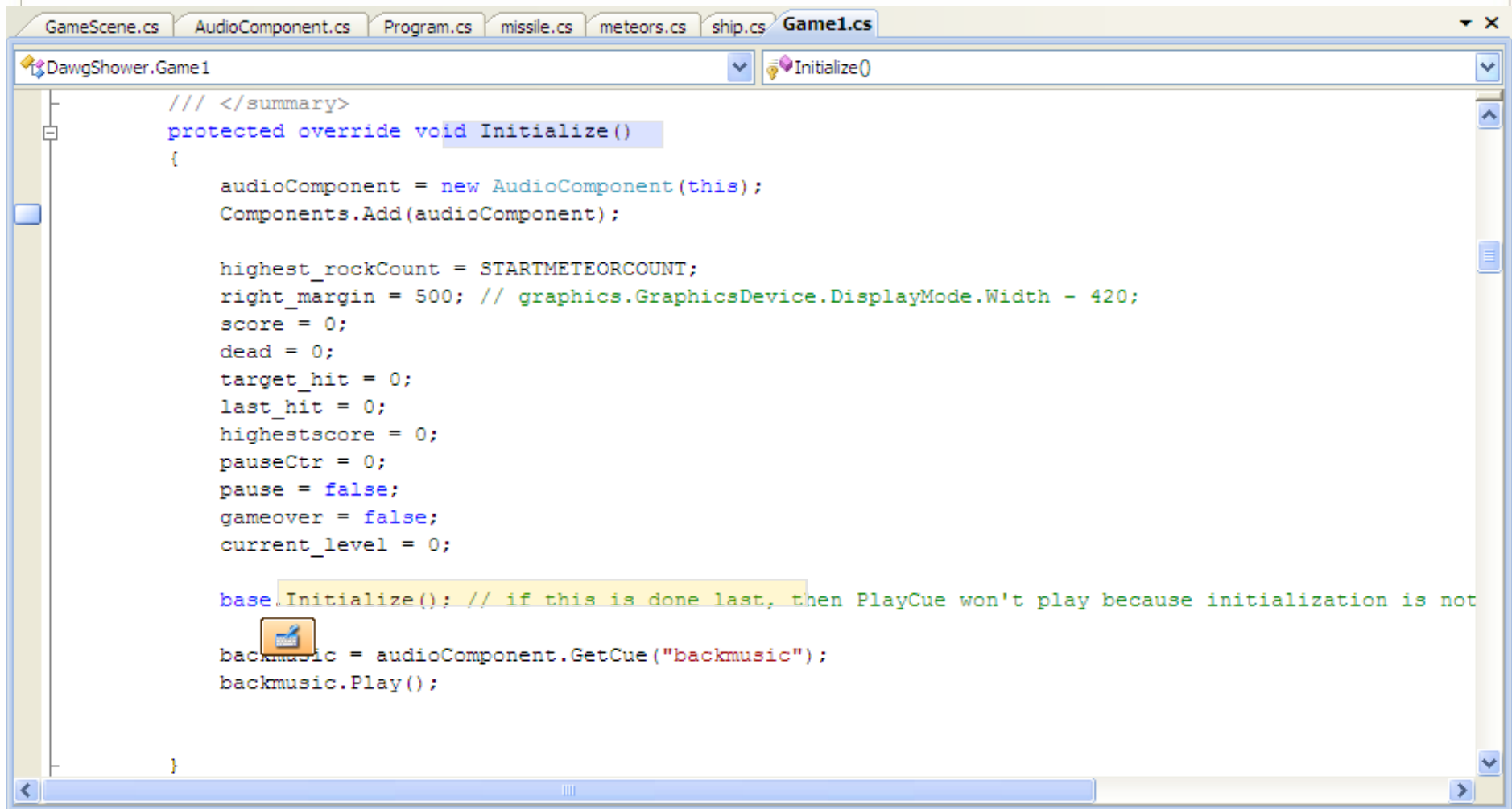
/// <summary>
/// Allows the game component to update itself.
/// </summary>
/// <param name="gameTime">Provides a snapshot of timing values.</param>
public override void Update(GameTime gameTime)
{
    // TODO: Add your update code here
}
```

AudioEngine object is the program reference to the audio services in your machine provided by XNA

This "Content" directory is located in <Proj>/bin/x86/<Sln Config>/Content

XACT – background looping music

- Add in Initialize code of the Game
- PlayCue is a method of SoundBank



```
GameScene.cs  AudioComponent.cs  Program.cs  missile.cs  meteors.cs  ship.cs  Game1.cs
DawgShower.Game1  Initialize()

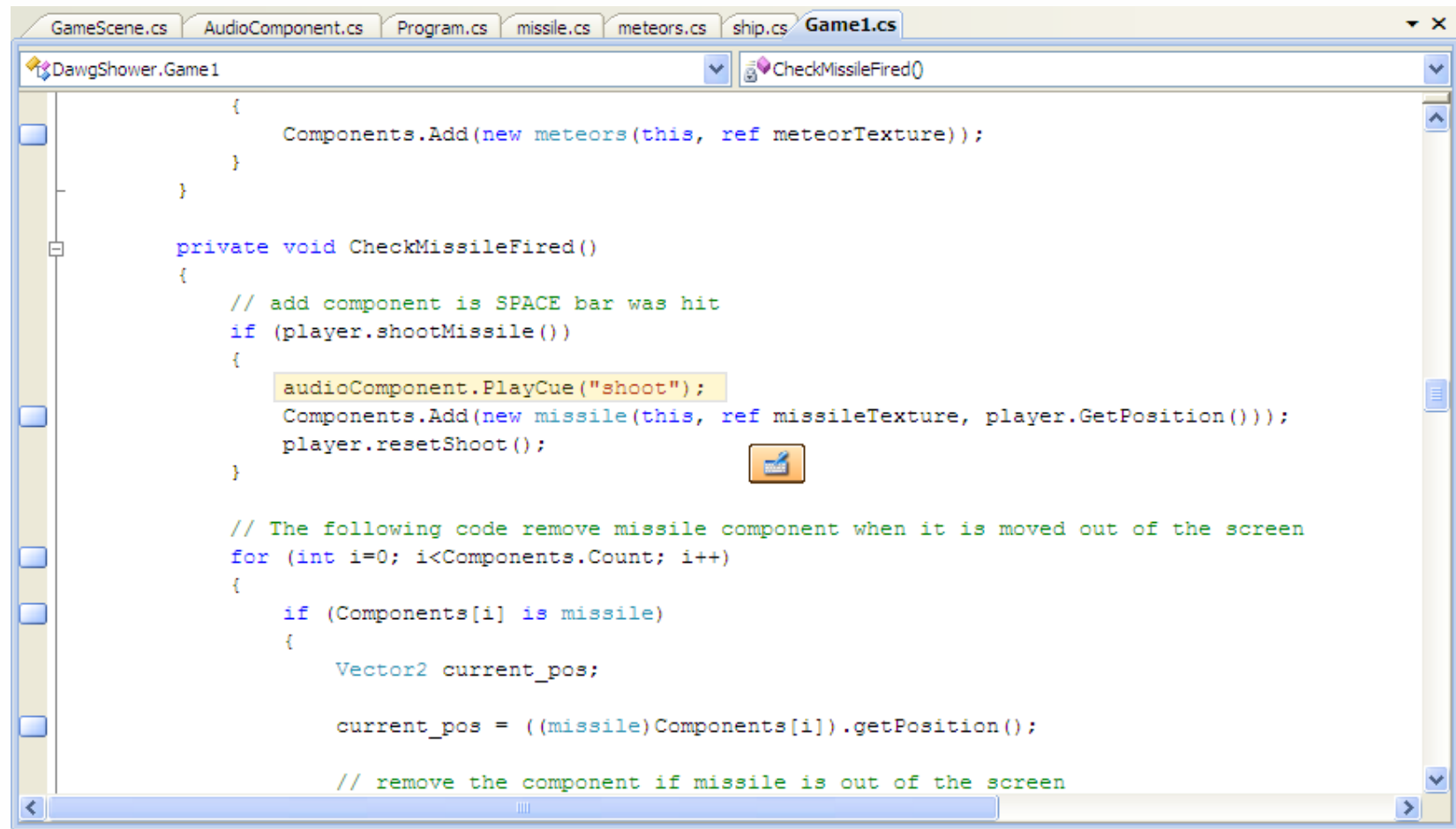
/// </summary>
protected override void Initialize()
{
    audioComponent = new AudioComponent(this);
    Components.Add(audioComponent);

    highest_rockCount = STARTMETEORCOUNT;
    right_margin = 500; // graphics.GraphicsDevice.DisplayMode.Width - 420;
    score = 0;
    dead = 0;
    target_hit = 0;
    last_hit = 0;
    highestscore = 0;
    pauseCtr = 0;
    pause = false;
    gameover = false;
    current_level = 0;

    base.Initialize(); // if this is done last, then PlayCue won't play because initialization is not
    backmusic = audioComponent.GetCue("backmusic");
    backmusic.Play();
}
```

XACT – play sound on event

- Shoot.wav in SoundBank was not set to “infinite”, thus will only be played once



```
GameScene.cs  AudioComponent.cs  Program.cs  missile.cs  meteors.cs  ship.cs  Game1.cs
DawgShower.Game1
CheckMissileFired()

{
    Components.Add(new meteors(this, ref meteorTexture));
}

private void CheckMissileFired()
{
    // add component is SPACE bar was hit
    if (player.shootMissile())
    {
        audioComponent.PlayCue("shoot");
        Components.Add(new missile(this, ref missileTexture, player.GetPosition()));
        player.resetShoot();
    }

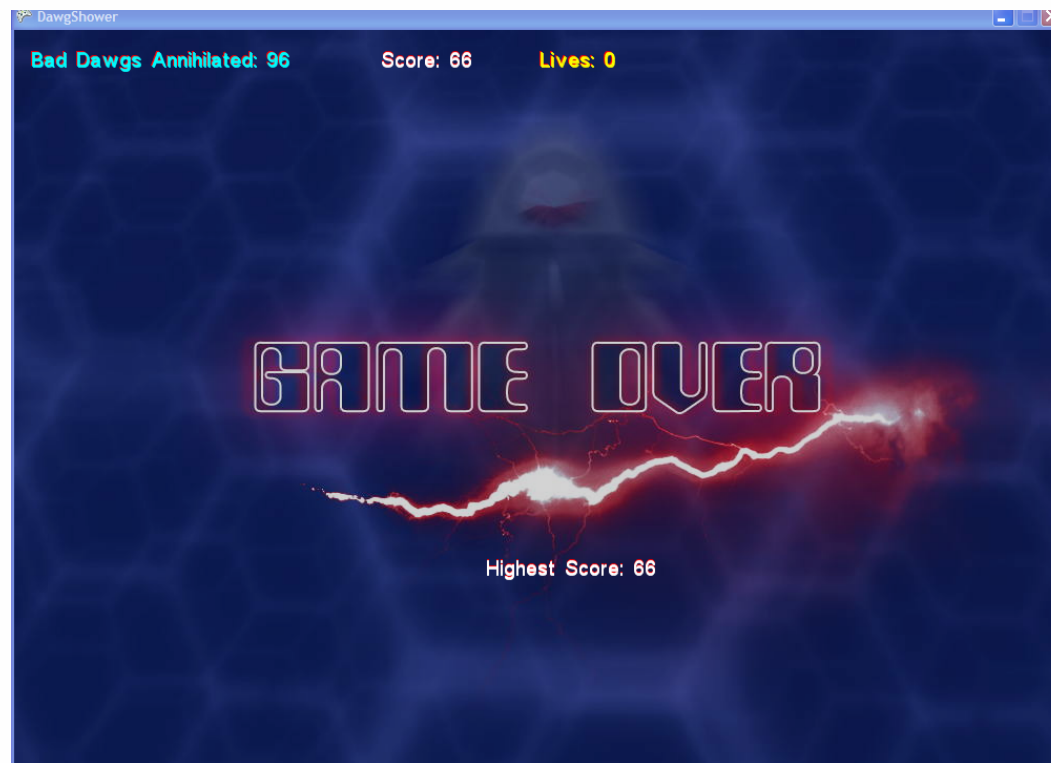
    // The following code remove missile component when it is moved out of the screen
    for (int i=0; i<Components.Count; i++)
    {
        if (Components[i] is missile)
        {
            Vector2 current_pos;

            current_pos = ((missile)Components[i]).getPosition();

            // remove the component if missile is out of the screen
        }
    }
}
```


Game Over

- Remove all components
- Replace background canvas
- Pause the music



Game Over - Code

```
private void RemoveGame()
{
    for (int i = 0; i < Components.Count; i++)
    {
        Components.RemoveAt(i);
        i--;
    }
}

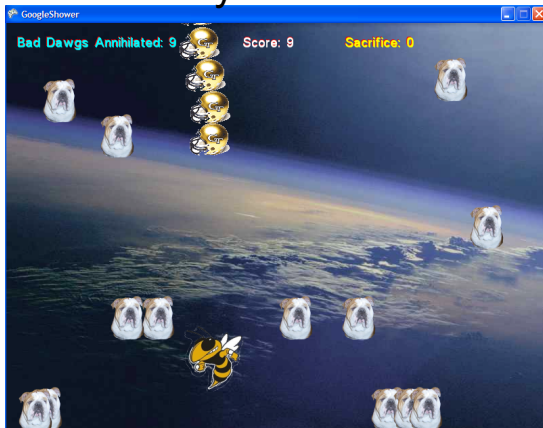
private void GameOver()
{
    if (dead >= DEAD_COUNT)
    {
 if (highestscore < score)
        highestscore = score;

        RemoveGame();
        backgroundTexture = gameoverTexture;
        gameover = true;
        player = null;
        backmusic.Pause();
    }
}
```

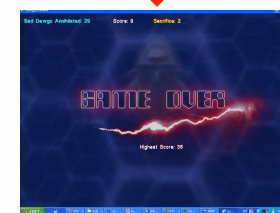
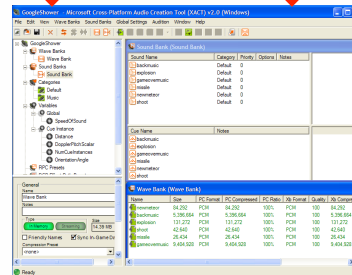
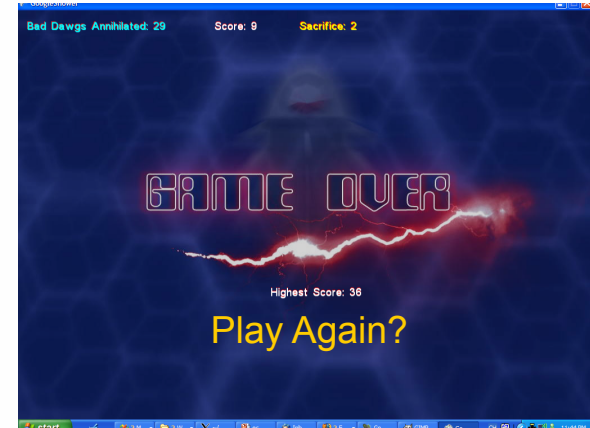
Organized game structure

Class GameScene : DrawableGameComponent

PlayScene



EndScene



Play Again?

SoundBank