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

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

Georgia Institute of Technology

Prof. Lee's game: DawgShower

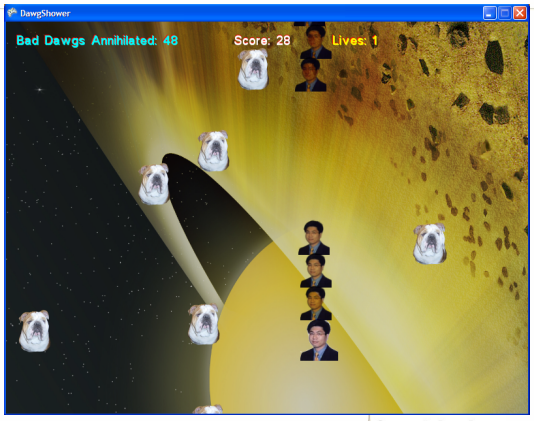
- Shoot the bad dawgs!
- Consist of four main objects
  - The shooter (ship.cs)
  - The bad dawgs (meteros.cs)
  - The missile (missile.cs)
  - Music (AudioComponent.cs)

- 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"

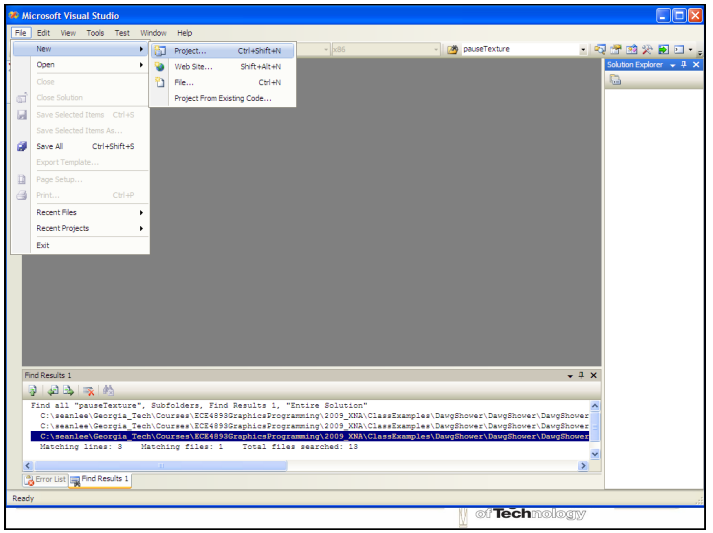
Georgia Institute of Technology

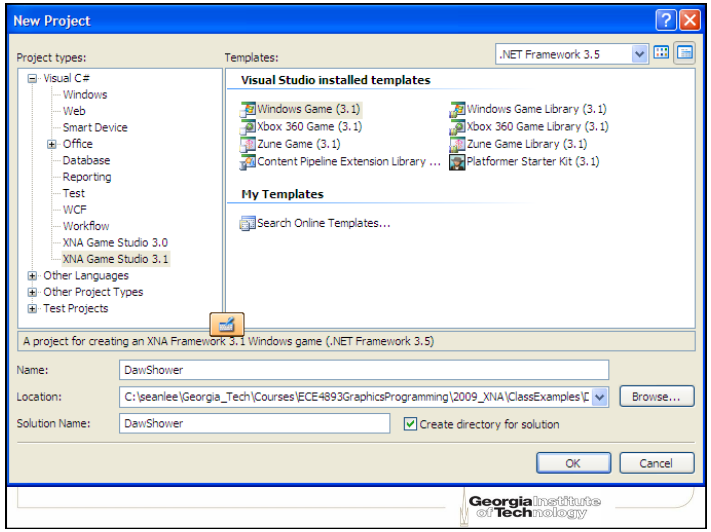
Screenshot

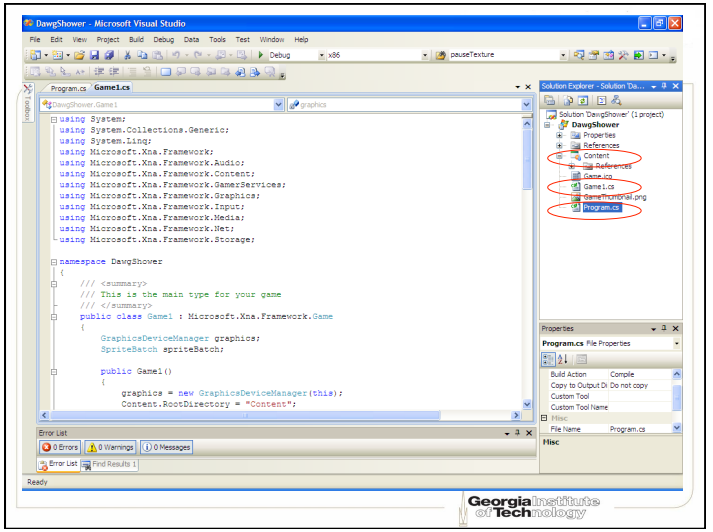


Georgia Institute of Technology

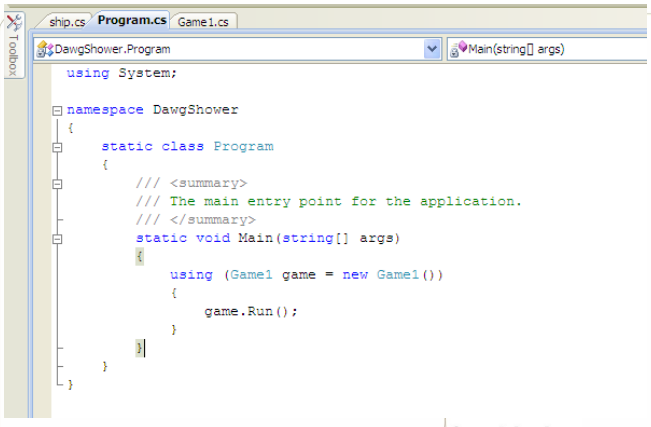
Demo DawgShower Game Example



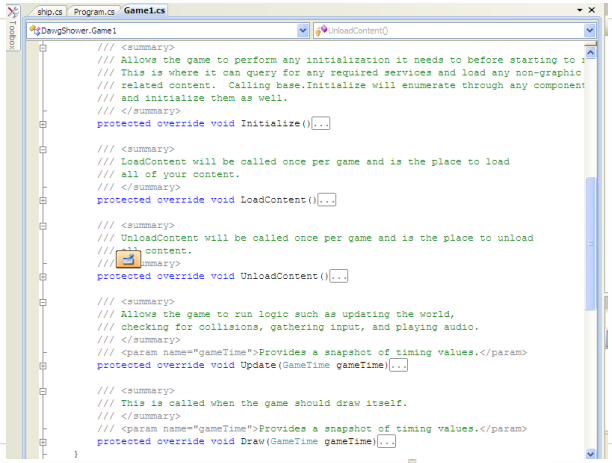




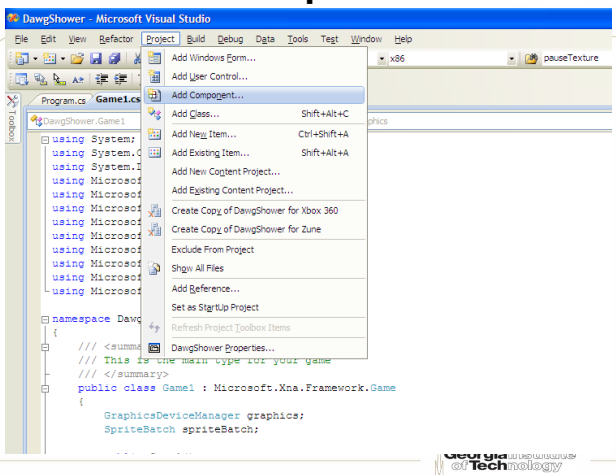
Top level program (Program.cs)



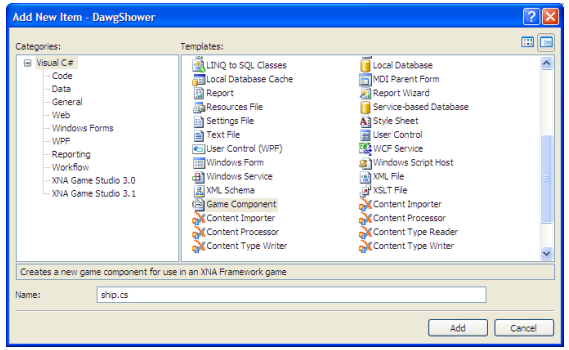
Main game loop (Game1.cs)



Add Game Component

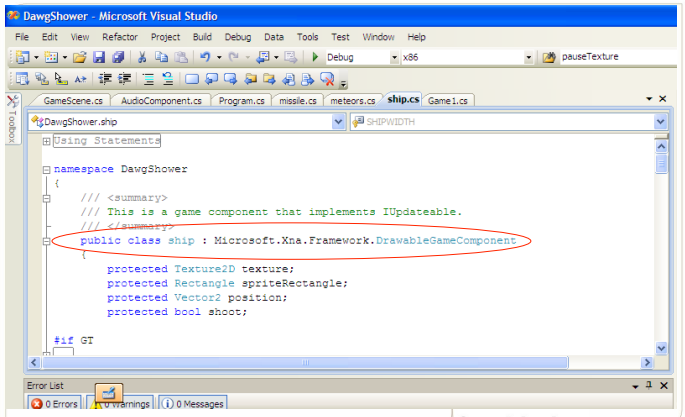


Adding Game Component (C# src file)



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

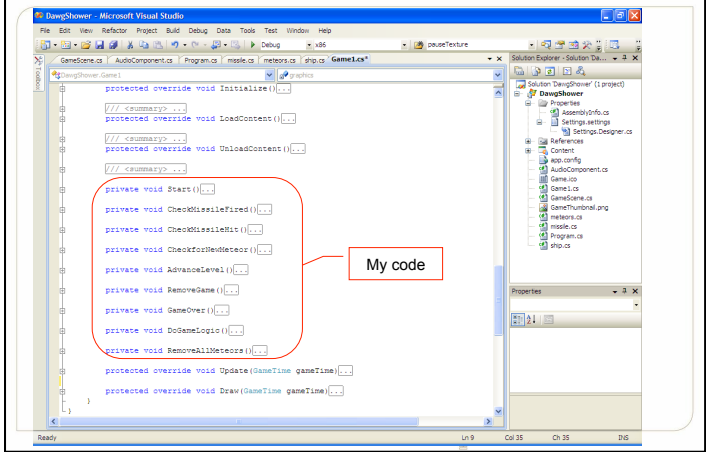
DrawableGameComponent class (1)



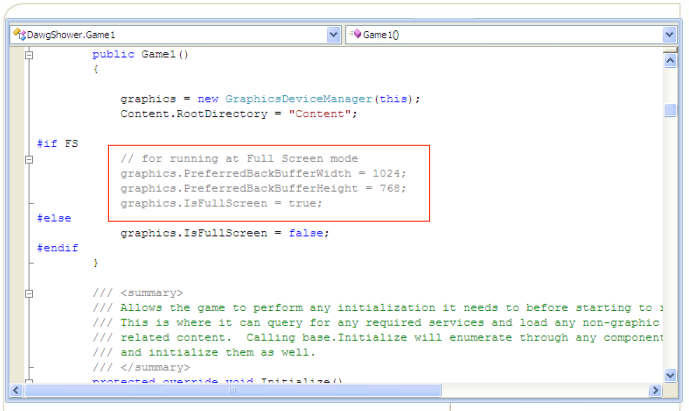
DrawableGameComponent class (2)

- 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



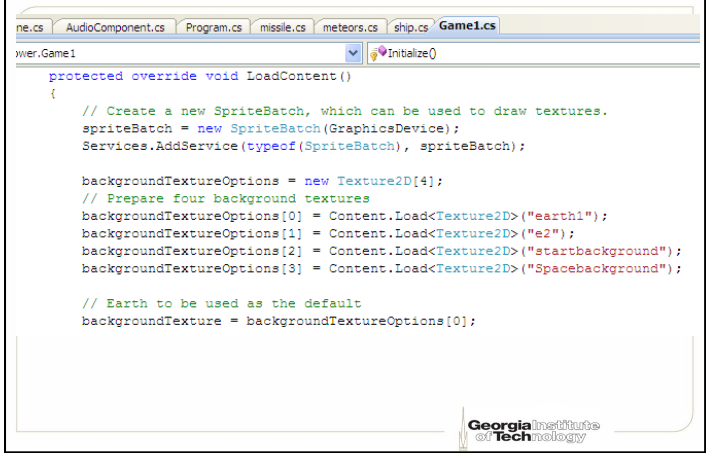
Full screen mode



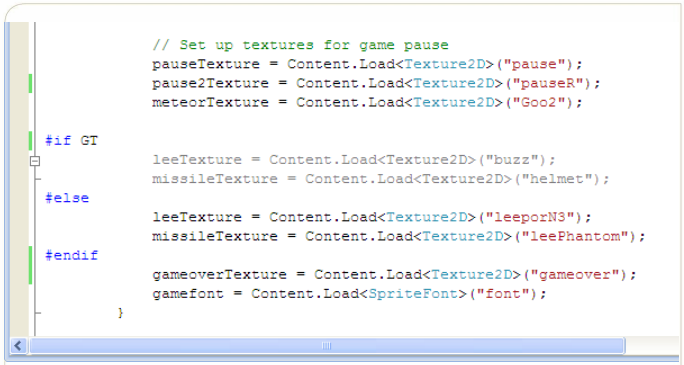
Drawing sprites using XNA

- 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



Small sprites for game assets



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()

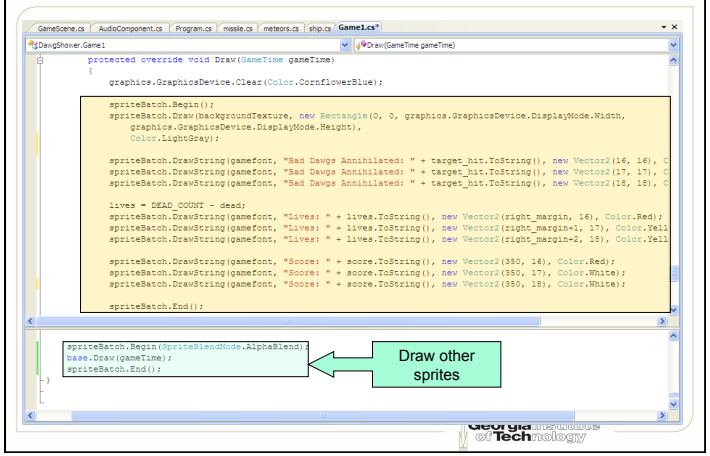
```
public override void Draw(GameTime gameTime)
{
    SpriteBatch sBatch =
        (SpriteBatch)Game.Services.GetService(typeof(SpriteBatch));

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

    base.Draw(gameTime);
}
```

Use GetService to acquire service for each DrawableGameComponent

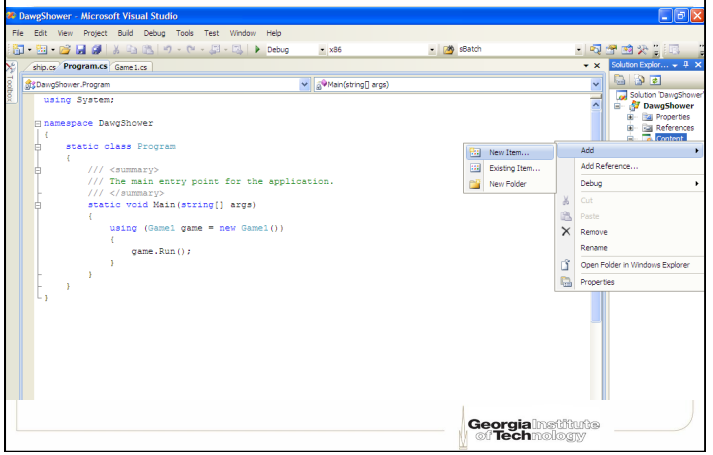
Drawing the background



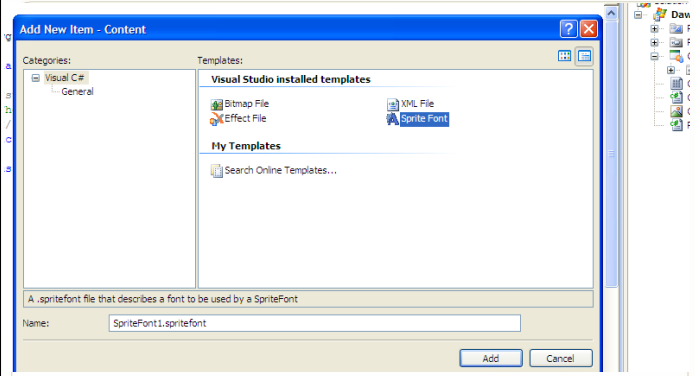
Printing text

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

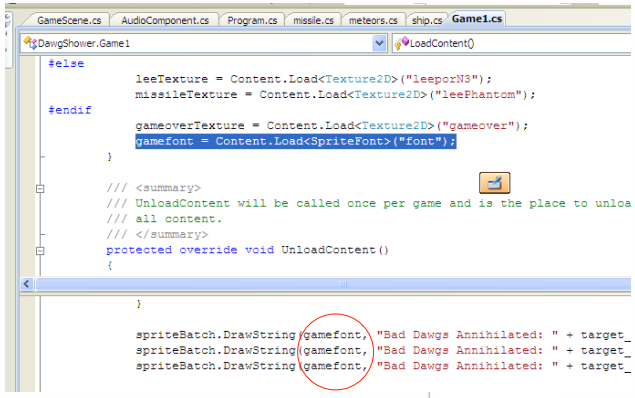
Create a SpriteFont (1)



Create a SpriteFont (2)



DrawString



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 && ![illegible]ver)
{
    DoGameLogic();
    base.Update(gameTime);
}
```

Built-in test for collision detection

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

Missile's position

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

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)

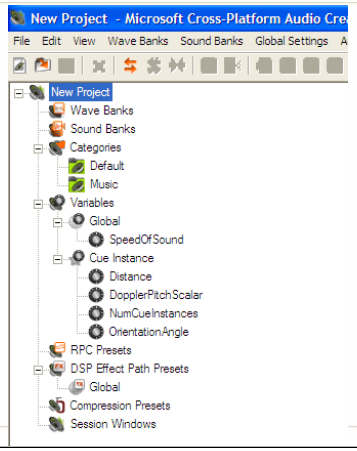
- Drag sound files into Content folder
- Load through Content Pipeline

```
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

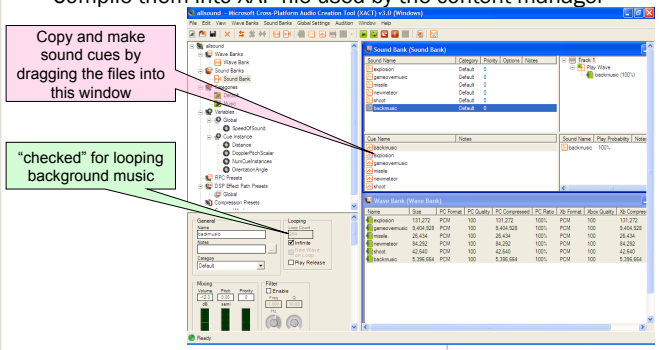
Sound – powerful, complex way



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

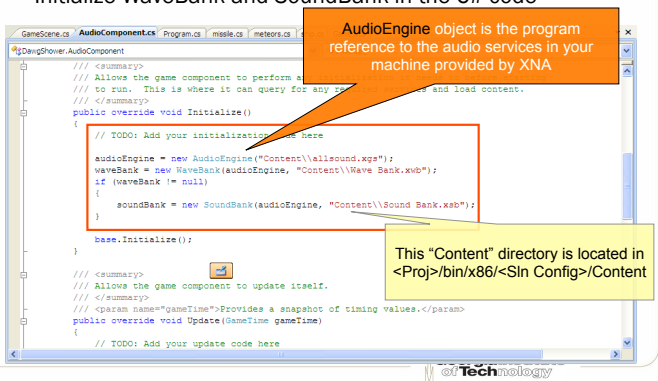
Using XACT

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



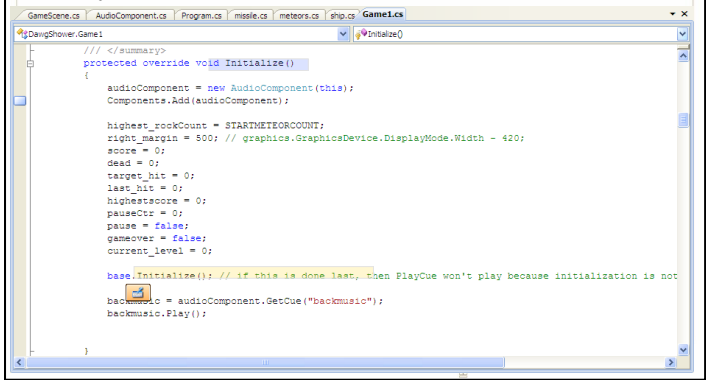
Using files created by XACT in XNA

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



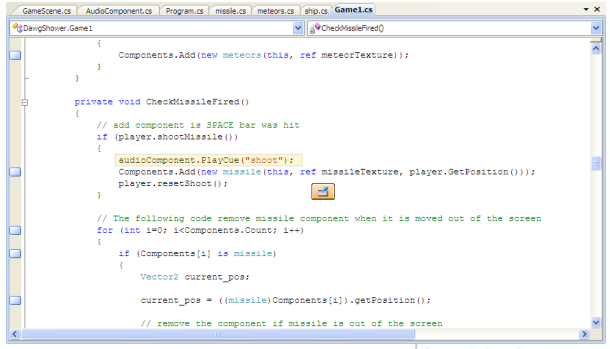
XACT – background looping music

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



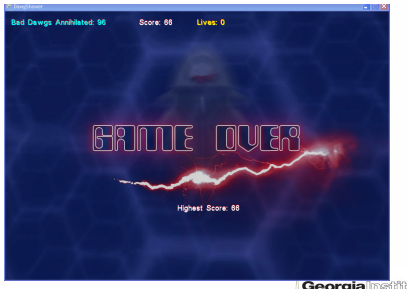
XACT – play sound on event

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



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