

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

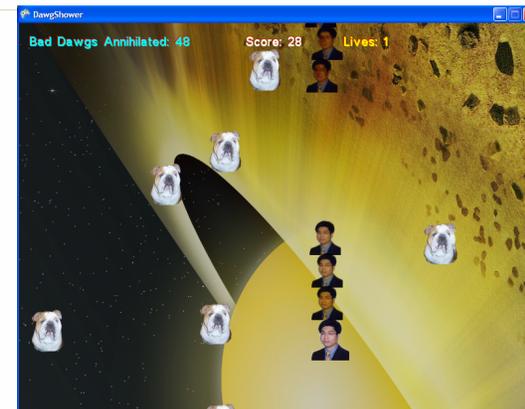
- Shoot the bad dawgs!
- Consist of four main objects
 - The shooter (ship.cs)
 - The bad dawgs (meteros.cs)
 - The missile (missile.cs)
 - Music (AudioComponent.cs)

- The moving objects are all made of sprites

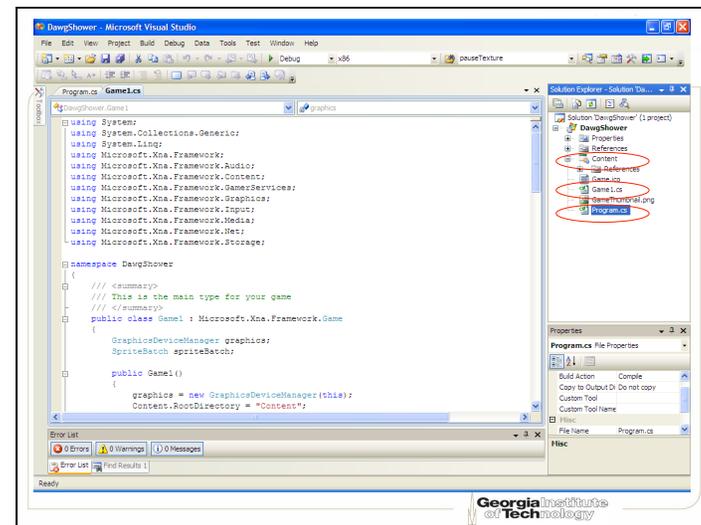
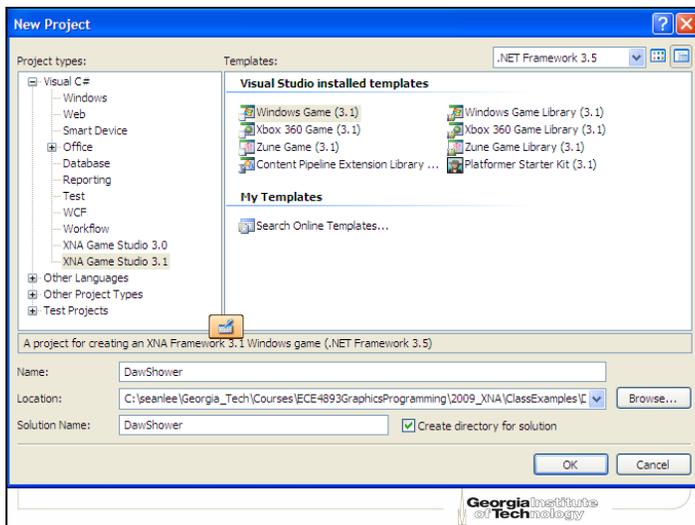
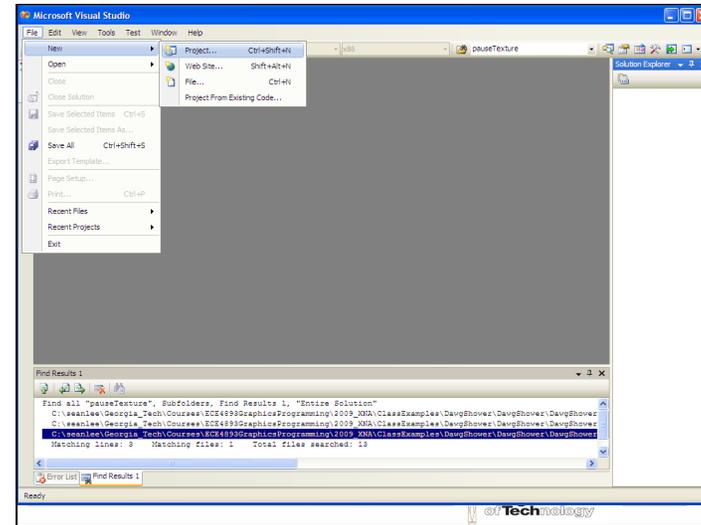
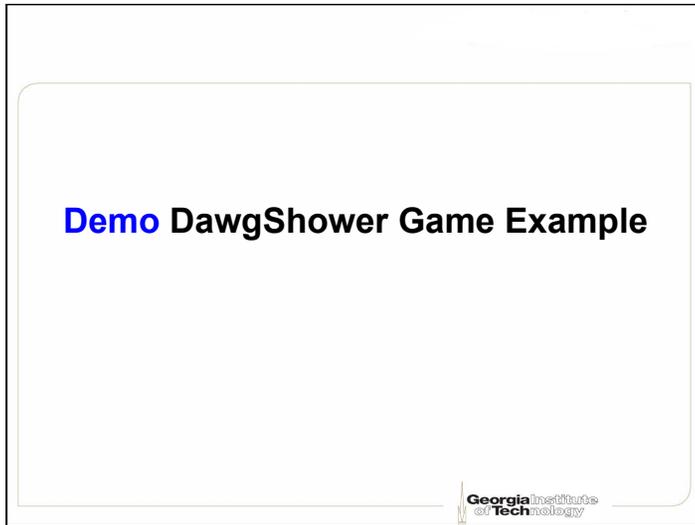
Adapted from Chapter 3 of A. Lobao, B. Evangelista, and J.A. Leal de Farias,
 “XNA 2.0 Game Programming: From Novice to Professional”

Georgia Institute of Technology

Screenshot



Georgia Institute of Technology



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)

```

/// <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(){}

/// <summary>
/// LoadContent will be called once per game and is the place to load
/// all of your content.
/// </summary>
protected override void LoadContent(){}

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

/// <summary>
/// Allows the game to run logic such as updating the world,
/// checking for collisions, gathering input, and playing audio.
/// </summary>
/// <param name="gameTime">Provides a snapshot of timing values.</param>
protected override void Update(GameTime gameTime){}

/// <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){}
    
```

Add Game Component

```

using System;
using System.Collections.Generic;
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;

namespace DawgShower
{
    /// <summary>
    /// This is the main type for your game
    /// </summary>
    public class Game1 : Microsoft.Xna.Framework.Game
    {
        GraphicsDeviceManager graphics;
        SpriteBatch spriteBatch;
    }
}
    
```

Adding Game Component (C# src file)

Creates a new game component for use in an XNA Framework game

Name:

Add Cancel

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

DrawableGameComponent class (1)

```

namespace DawgShower
{
    /// <summary>
    /// This is a game component that implements IUpdateable.
    /// </summary>
    public class ship : Microsoft.Xna.Framework.DrawableGameComponent
    {
        protected Texture2D texture;
        protected Rectangle spriteRectangle;
        protected Vector2 position;
        protected bool shoot;

        #if GT
    }
    #endif
}
    
```

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

```

protected override void Initialize()
{
}

/// <summary> ...
protected override void LoadContent()
{
}

/// <summary> ...
protected override void UnloadContent()
{
}

/// <summary> ...
private void Start()
{
}

private void CheckMissileFired()
{
}

private void CheckMissileHit()
{
}

private void CheckForNewMeteor()
{
}

private void AdvanceLevel()
{
}

private void ResumeGame()
{
}

private void GameOver()
{
}

private void DoGameLogic()
{
}

private void RemoveAllMeteoors()
{
}

protected override void Update(GameTime gameTime)
{
}

protected override void Draw(GameTime gameTime)
{
}
    
```

Full screen mode

```

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()
{
}
    
```

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

```

ne.cs  AudioComponent.cs  Program.cs  missile.cs  meteors.cs  ship.cs  Game1.cs
wvr.Game1  Initialize()
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 spriteBatch =
        (SpriteBatch)Game.Services.GetService(typeof(SpriteBatch));

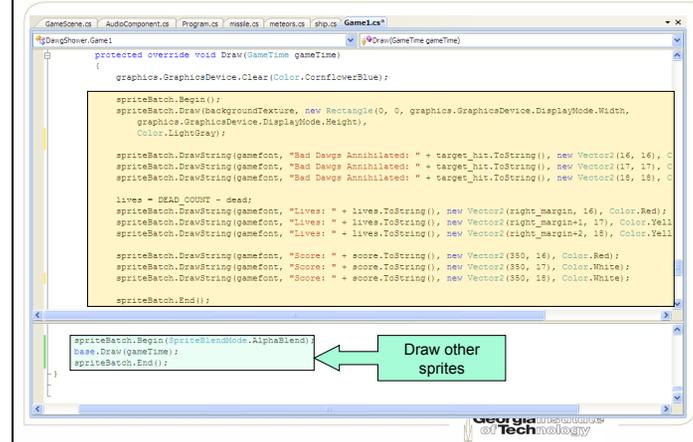
    spriteBatch.Draw(texture, position, spriteRectangle, Color.White);

    base.Draw(gameTime);
}
```

Use GetService to acquire service for each DrawableGameComponent



Drawing the background

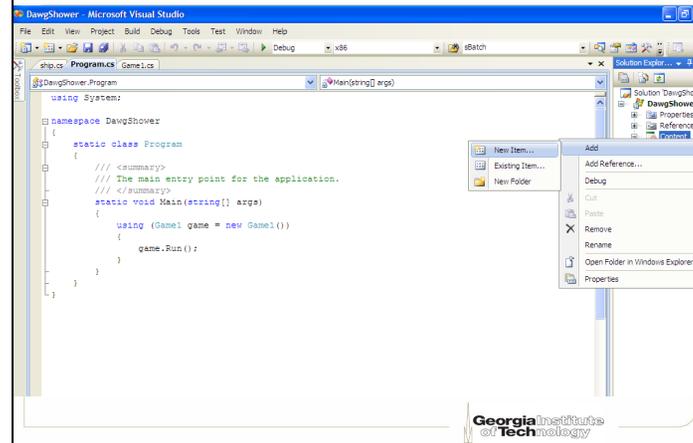


Printing text

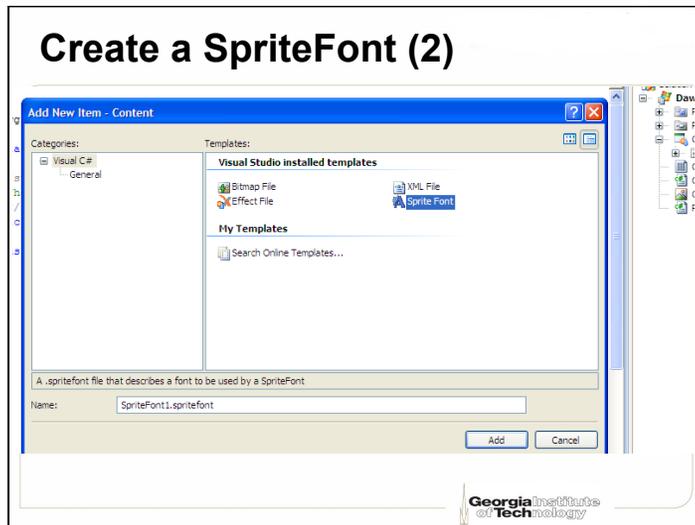
- Draw using **SpriteBatch**
- Create a “font sprite”
- Based on available Fonts in the system
- Add font in **LoadContent()**



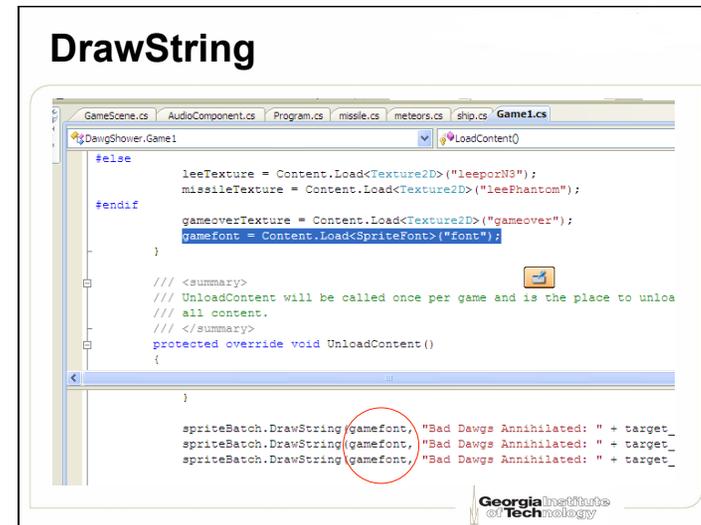
Create a SpriteFont (1)



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[0] is meteors)
{
    bool hasCollision = false;
    hasCollision = ((meteors)Components[0]).CheckCollision(((missile)Components[1]).GetBounds());
    if (hasCollision)
    {
        audioComponent.PlayCue("explosion");
        // remove collided meteors (BadDawgs)
        score++;
        Components.RemoveAt(0);
    }
}
```

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 && !GameOver())
{
    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

BoundingBox

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

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

```
public bool Intersects (
    BoundingBox sphere
)
```

Georgia Institute
of Technology

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

Georgia Institute
of Technology

Sound – simple, easy way (2)

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

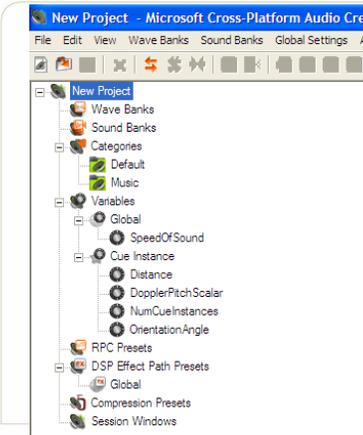
```
SoundEffect soundEffect =
    Content.Load<SoundEffect>(@"Content\Audio\LaserShot");

soundEffect.Play();
```

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

Georgia Institute
of Technology

Sound – powerful, complex way



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

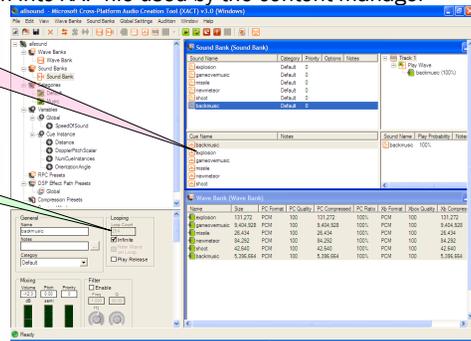
Georgia Institute
of Technology

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

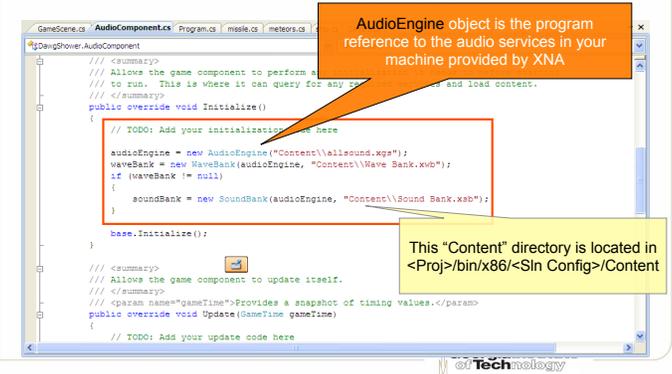


Using files created by XACT in XNA

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

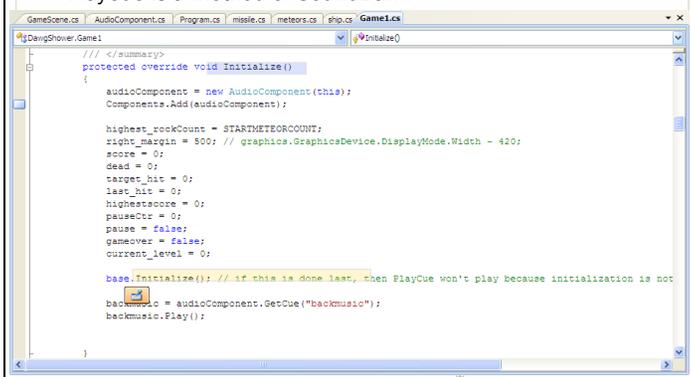
AudioEngine object is the program reference to the audio services in your machine provided by XNA

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



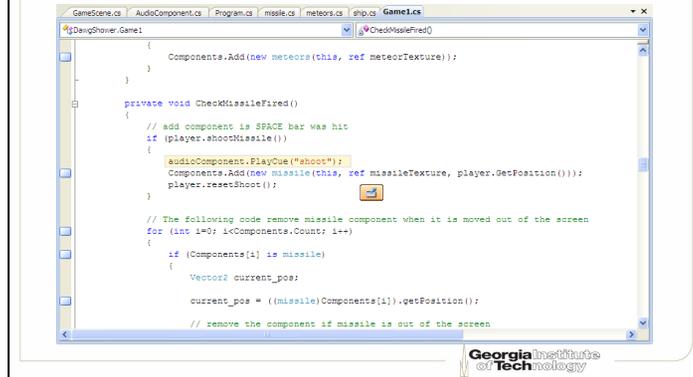
XACT – background looping music

- 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



Georgia Institute of Technology

Game Over - Code

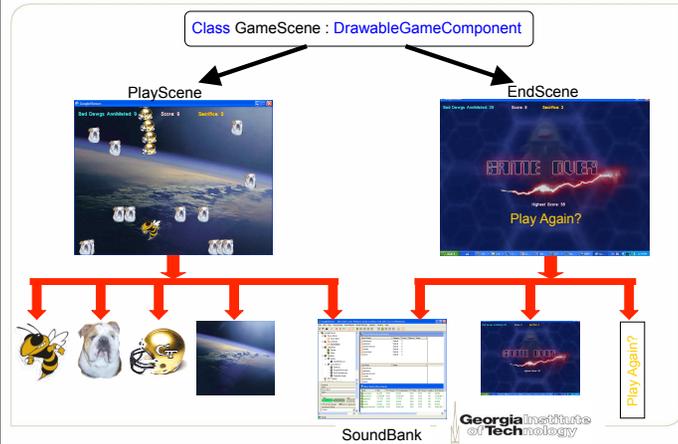
```
private void RemoveGame ()
{
    for (int i = 0; i < Components.Count; i++)
    {
        Components.RemoveAt(i);
        i--;
    }
}

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

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

Georgia Institute of Technology

Organized game structure



Georgia Institute of Technology