Chapman University
CPSC 340 - Spring 2011

Class #3
September 15, 2011
William Wood Harter
Adjunct Professor

Summary

- Next week's class is online
  - I will give details in class. If you miss class and need details I will mail them individually so it's not on a public website or list.
- Components
- Simulating Gravity
- Game States
- Level Building

Components

- Did anyone use components to solve the button problem?
- Screen shot of my game and the Game1.draw method
- Where are the buttons being drawn?


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

    spriteBatch.Begin();

    string eTime = stElapsedSecs + (int) gameTime.TotalGameTime.TotalSeconds;
    spriteBatch.DrawString(myFont,eTime,vTimePos, Color.Tomato);

    spriteBatch.End();

    base.Draw(gameTime);
}
```

Components

- XNA DrawableGameComponent
  - Has Initialize, Update and Draw methods
  - Just like Microsoft.Xna.Framework.Game
- You can create your own class that extends
  - microsoft.xna.framework.DrawableGameComponent
- You create instances of your component and add it to the game like this

```
btnExit = new Button(this, new Vector2(10, 140), "exit-button-norm", "exit-button-over", "exit-button-down");
btnExit.Initialize();

Components.Add(btnLoad);
```

Good Use Of Static Variables

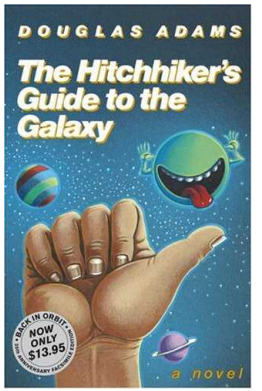
- Only one button down or over at a time
- The buttons need a variable they can all share to see if one of them is down
- I use a static in the Button class to do this.

```
// static so there is one mouse down flag shared by all the buttons
public static bool isMouseDown = false;
```

- Then all the buttons check/set this in update when checking mouse pressed changes

```
if ((isOver) && (!pressed) && (ms.LeftButton == ButtonState.Pressed) && (!Button.isMouseDown))
{
    // new mouse click over this button
    pressed = true;
    Button.isMouseDown = true;
}
```

Delegates



- Sounds fancy and complicated, right?
- Fancy term, simple concept. Don't Panic!
- How does your generic button notify you it is clicked when you create one for New Game and the same button type for Exit?
- We could create a subclass for every button and override a clicked method, but that would be a lot of extra work.
- Wouldn't it be nice if our button could just tell us it was clicked.

Delegates

- Delegates are notification events.
- Basically, you tell the button to call a specific method on a specific event. In this case “Clicked is the event. Easy.
- Don't you feel better already?

- My button only has one event “Clicked” can you think of others we could add?

How to setup a delegate in your class

- The delegate key word identifies what the method looks like that will be called. In this case it returns void and has no parameters.
- The clickedClients is a list that is automatically setup because of the delegate type in ButtonClicked.
- The Add and remove methods simply += the delegete (method) that is passed in.

```
/// <summary>
/// ButtonClicked delegate to receive click events from the button
/// </summary>
public delegate void ButtonClicked();

/// <summary>
/// the list of clicked delegates
/// </summary>
private ButtonClicked clickedClients;

/// <summary>
/// Add a clicked delegate to this button
/// </summary>
/// <param name="newDelegate"></param>
public void AddClickedDelegate(ButtonClicked newDelegate)
{
    clickedClients += newDelegate;
}

/// <summary>
/// Remove a button delegate
/// </summary>
/// <param name="newDelegate"></param>
public void RemoveClickedDelegate(ButtonClicked newDelegate)
{
    clickedClients -= newDelegate;
}
```

How your class calls the delegate list

- Simply calling clickedClients() will automatically iterate the list of delegates that were added to the list with AddClickedDelegate and call each one.

```
else if ((isOver) && (pressed) && (ms.LeftButton == ButtonState.Released))
{
    // pressed and released over the button - cause a click event
    Console.WriteLine("Button clicked");

    // if there are clicked delegates, call them.
    if (clickedClients != null)
    {
        clickedClients();
    }

    // reset flags
    pressed = false;
    Button.isMouseDown = false;
}
```

How to have a delegate call you

- Define a normal everyday method that matches what the delegate expects.
- Call the AddClickedDelegate and pass it that method.
- Now, ExitClicked will be called when the button is clicked.

```
protected void ExitClicked()
{
    Console.WriteLine("Exit button clicked");
}
```

```
btnExit = new Button(this, new Vector2(10, 140), "exi
btnExit.Initialize();
btnExit.AddClickedDelegate(ExitClicked);
Components.Add(btnExit);
```

Complete Solution

- My solution to this week's homework used components and delegates.
- It is uploaded to the course website assignments page.
  - http://www.gamedev360.com
- One of the main reasons you only get 50% if you turn it in late. It's pretty simple to 'tweak' my solution and turn it in.

How To Simulate 2D Gravity

- Create an image that is a circle
- Draw it as other images to the screen.

```
spriteBatch.Draw(texBall, vBallPos, Color.White);
```

- Create vectors for the ball velocity.

```
const float grav = 1.0f;
Vector2 vBallVel = Vector2.Zero;
Vector2 vBallPos = new Vector2(50, 50);
Texture2D texBall;
```

- In the update add gravity to the ball every update.

```
vBallVel.Y = vBallVel.Y + grav;
vBallPos = Vector2.Add(vBallPos, vBallVel);
```

The ball bounces

- In the update, after adding velocity to the current position, check to see if the ball should bounce.

```
if ((vBallPos.Y > GraphicsDevice.PresentationParameters.Bounds.Height - 64) && (vBallVel.Y > 1))
{
    vBallVel.Y = -vBallVel.Y;
}
```

- Check to make sure the ball is moving down before reversing the velocity direction.
- What about mass? This is just a simulation, mass isn't necessary to make it “look” like gravity. The ball bounces doesn't it?

Game State

- What are the different states in a game?
  - Splash
  - Menu
  - Play
  - Pause
  - Game Over

Game State Variables

- The current state of a game is simply a variable. I use integers and constants.

```
const int STATE_MENU = 0;
const int STATE_PLAY = 1;
int currentState = STATE_MENU;
```

- You could also use enumerations. Personal preference.

Game State: Draw

- Notice my draw method now draws differently for each state.
- Shows the ball during play state
- Shows the timer during menu state

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

    if (currentState == STATE_MENU)
    {
        spriteBatch.Begin();
        string eTime = stElapsedSecs + (int)gameTime.TotalGameTime.TotalSeconds;
        spriteBatch.DrawString(myFont, eTime, vTimePos, Color.Tomato);
        spriteBatch.End();
    }
    else if (currentState == STATE_PLAY)
    {
        spriteBatch.Begin();
        spriteBatch.Draw(texBall, vBallPos, Color.White);
        spriteBatch.DrawString(myFont, "Game is playing - Press X to end", new Vector2(10, 10), Color.Tomato);
        spriteBatch.End();            }

    base.Draw(gameTime);
}
```

Changing States

- When do you change states?
- From Menu to Play state when the Start button is pressed.
- Notice that I also remove the button components.

```
protected void StartClicked()
{
    Console.WriteLine("Start button clicked");
    currentState = STATE_PLAY;
    Components.Remove(btnExit);
    Components.Remove(btnLoad);
}
```

Game State: Update

- Change update to do different things for different states. For this case, only STATE_PLAY does anything in update.
- Updating the ball position during play.
- Checking for the X key (to change state back).
- Add button components.

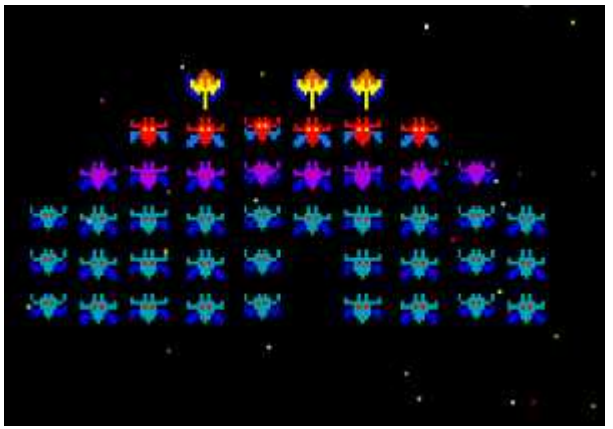
```
if (currentState == STATE_PLAY)
{
    vBallVel.Y = vBallVel.Y + grav;
    vBallPos = Vector2.Add(vBallPos, vBallVel);

    if ((vBallPos.Y > GraphicsDevice.PresentationParameters.Bounds.Height - 64) && (vBallVel.Y > 1))
    {
        vBallVel.Y = -vBallVel.Y;
    }

    if (vBallPos.Y > GraphicsDevice.PresentationParameters.Bounds.Height)
    {
        vBallVel.Y = -32;
    }

    KeyboardState ks = Keyboard.GetState();
    if (ks.IsKeyDown(Keys.X))
    {
        currentState = STATE_MENU;
        Components.Add(btnExit);
        Components.Add(btnLoad);
    }
}
```

Level Building: Galaxian



- Let's discuss some level building techniques
- Complex levels usually require a separate application
- Could you create this Galaxian level using a text file?

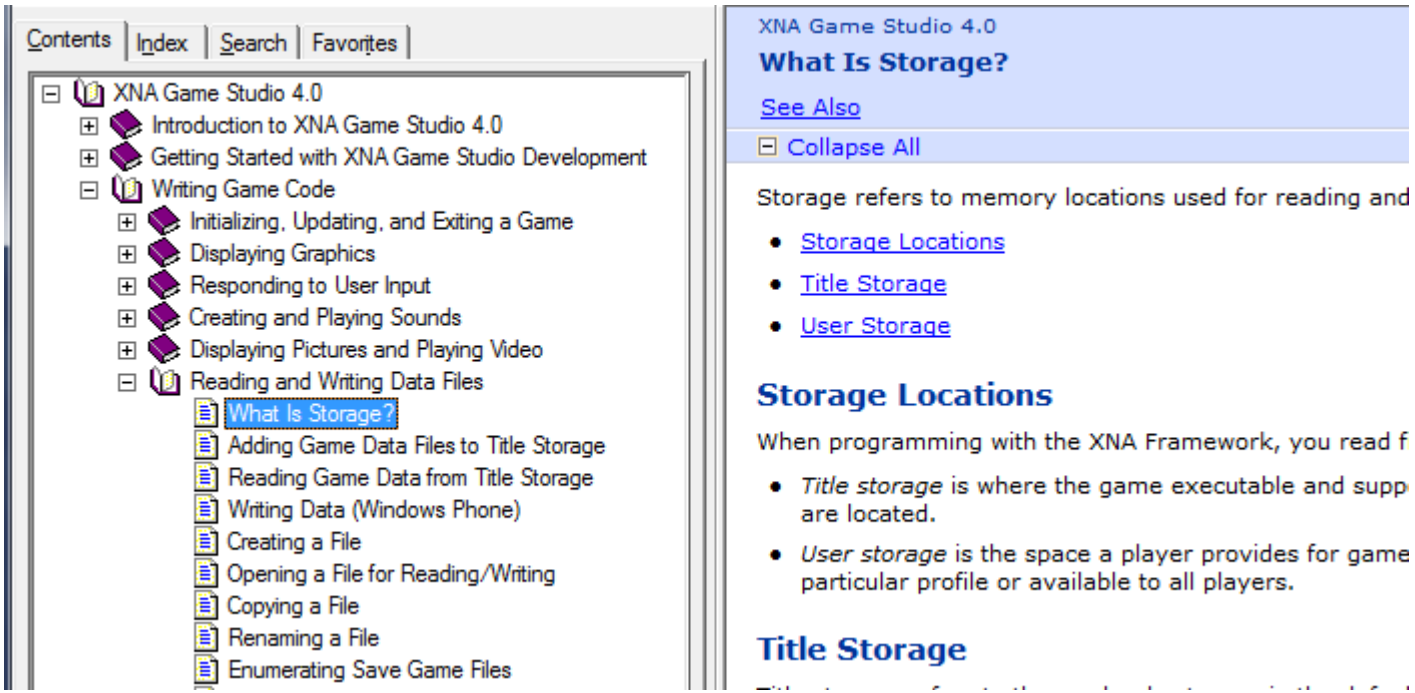
Galaxian Level Text File

- Lines could be rows
- Different characters could represent different types of ships.
- Read each line and find out where the ships are.
- File: galaxian_level_1.txt

```
1     Y YY
2    RRRRRR
3   PPPPPPPP
4  BBBBBBBBBB
5  BBBBB BBBB
6  BBBBB BBBB
```

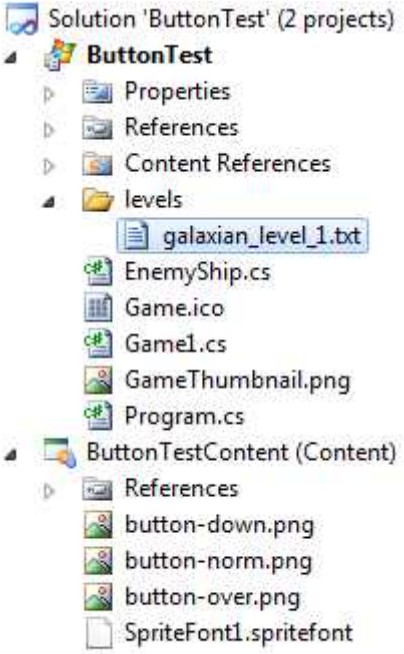
How To Open A File

- Where would I look to find how to read files in XNA?
- How about the help?



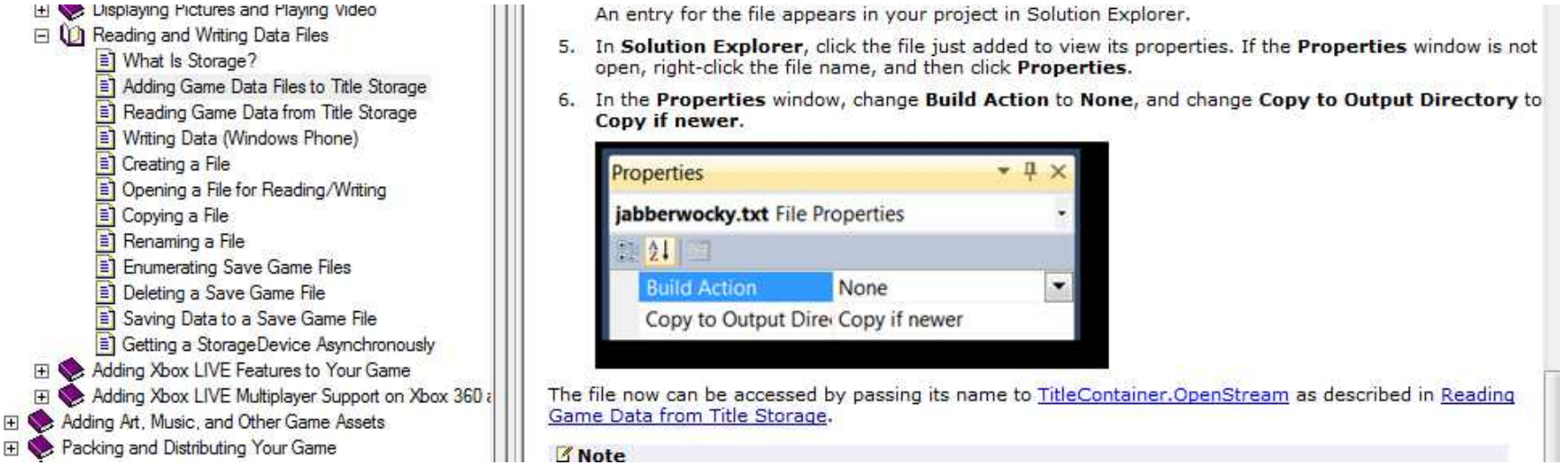
Files can be added to your project

- Download the galaxian_level_1.txt file from the course website.
- Add it to the project
- If you use subdirectories (recommended) you may need to create the subdirectory first.



Change the document properties

- The document properties need to be changed to copy it to the final game build.
- Explained in detail in the XNA help.



- The file now can be accessed by passing its name to TitleContainer.OpenStream as described in Reading Game Data from Title Storage.

What are streams?

- Streams are classes that are used to read and write data to and from files.
- Steams can also read to and from the network.
- Why do we need to use TitleContainer instead of just using the .NET class for FileInputStream

Reading Files Using Streams

```
    LoadLevelFile("levels\\galaxian_level_1.txt");
}

protected void LoadLevelFile(String levelfile)
{
     try
    {
        System.IO.Stream stream = TitleContainer.OpenStream(levelfile);
        System.IO.StreamReader sreader = new System.IO.StreamReader(stream);

        Console.WriteLine("File Size: " + stream.Length);
        string line;
        while ((line = sreader.ReadLine()) != null)
        {
            Console.WriteLine(line);
        }
        stream.Close();
    }
    catch (System.IO.FileNotFoundException e)
    {
        // this will be thrown by OpenStream if gamedata.txt
        // doesn't exist in the title storage location
        Console.WriteLine(e);
    }
}
```

Parsing each line

- We need to parse each line now that it can be printed to the console
- Keep a counter for the row
- Run another iteration that just prints the ship type, the row and the column.

Parsing for Yellow Ships

- This reads each line looking for yellow ships
- Is there a difference between “Y” and 'Y'?
- Why print col,row instead of row,col?

```
int iRow = 0;
while ((line = sreader.ReadLine()) != null)
{
    int iCol = 0;
    for (iCol = 0; iCol < line.Length; iCol++)
    {
        if (line[iCol].Equals('Y'))
        {
            Console.WriteLine("Yellow Ship at " + iCol + "," + iRow);
        }
    }
}
```

Improving EnemyShip

- We created the simple ship types as classes last week.
- We glossed over the location of a ship
- We need a constructor to set the location.
- Since all ships have location, put it in the base class, right?
- The constructor needs to load the image? Or passed the image? Your choice. I like it to load it's own image.

EnemyShip Constructors

- Base Class Constructor

```
public EnemyShip()
{
}

public EnemyShip(int row, int col)
{
    location = new Vector2(col, row);
}
```

- BlueShip Constructor

```
public BlueShip(int row, int col):base(row,col)
{
    image = Game1.game1.Content.Load<Texture2D>("blueship");

}
```

- Uh oh, what is Game1.game1.Content...?

Detour Static Variables

- Another good example of using statics.
- They are like globals. They exist in the class and are accessed from any other class (if they are public)
- In this case, the BlueShip needs the Content class instance to load the image
- The content exists in the game1 instance.
- We create a static variable on Game1 to hold the instance on game1. Does your head hurt yet?

Detour Static Variables

- It's actually pretty simple.
- Create the static variable in Game1

```
public class Game1 : Microsoft.Xna.Framework.Game
{
    public static Game game1 = null;
```

- Then in the Game1 constructor initialize the value of Game1.game1 to 'this'.

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

    game1 = this;
}
```

- 'this' is just a helper variable to get the current instance of the object associated with the method being called.

Detour Static Variables

- Now your BlueShip can get at the game1 instance (there should be only one instance of the game)
- That instance has the Content object which we use to load content.

```
public BlueShip(int row, int col):base(row,col)
{
    image = Game1.game1.Content.Load<Texture2D>("blueship");

}
```

What data structure to store ships

- What data structure will you use to store ships?
- EnemyShip[] ships;
- EnemyShip[][] ships;
- List<EnemyShip> ships;
- The choices are endless.
- Since these are fixed sizes, the single dimensional array is probably most efficient. My opinion.
- Since this is an early programming exercise, the two dimensional array is probably the most human to use. I would use a single dimensional array.

Two Dimensional Arrays

- Yes, behind the curtain I used google to see the syntax for C# arrays.
  - http://msdn.microsoft.com/en-us/library/aa288453%28v=vs.71%29.aspx
- Declare the array in Game1

```
EnemyShip[][] ships;

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

    game1 = this;
}
```

Populate the ships array

- Create the ships array
- Read each line, parse and add the instances of ships to it.

```
ships = new EnemyShip[12][];
int i;
for (i = 0; i < ships.Length; i++)
{
    ships[i] = new EnemyShip[12];
}
while ((line = sreader.ReadLine()) != null)
{
    int iCol = 0;
    for (iCol = 0; iCol < line.Length; iCol++)
    {
        if (line[iCol].Equals('Y'))
        {
            Console.WriteLine("Yellow Ship at " + iCol + "," + iRow);
            ships[iCol][iRow] = new YellowShip(iRow,iCol);
        }
    }
}
```

What about empty array elements

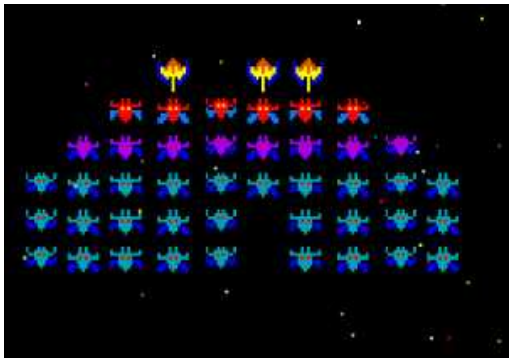
- Notice that array element [0][0] is empty.
- What happens?
- How would we draw these ships?
- A for loop in the Game1.Draw method?
  - Don't miss the fact that Draw is not done an the ships have a specific amount of space between them. This is the part I want you to think through on your own.

```
int i;
int j;
for (i = 0; i < ships.Length; i++)
{
    for (j = 0; j < ships[i].Length; j++)
    {
        ships[i][j].Draw(gameTime);
    }
}
```

Yes, there is a bug here!

Assignment #5

- Finish the code to populate the array of ships with the level text file. Add the ship images (can be blobs for all I care 64x64 or 32x32 pixels might be a good size. You decide)
- Once you press the start game button stop drawing the buttons (game state) and draw the list of ships.
- The easiest way to start is probably to use the GameState sample code from the lecture and copy in the EnemyShip.cs file from the other lecture sample, create the ships and go from there.
- The ships do not have to move. They only have to be drawn based on the level given and that is the starting Galaxian level layout. Much more detail on the course website.



```
1    Y YY
2   RRRRRR
3  PPPPPPPP
4 BBBBBBBBBB
5 BBBBB BBBB
6 BBBBB BBBB
```

Open Time

- Feel free to ask hard questions
- Feel free to work on your home work and ask questions while I'm still here
- Feel free to use this time for your own benefit