CMPS 20: Game Design Experience

[Image of XNA logo]
## Foreach Statement

```csharp
foreach ( type identifier in array-or-collection )
{
    ...
}
```

- Iterates through all elements in an array, or collection type
- Creates identifier just for the scope of the statements inside the `foreach`
  - Holds the current element within the array/collection
- Very convenient when it can be used

```csharp
string[] aStringArray = { “Cherry”, “Apple”, “Banana”, “Peach” }; // Sort elements
Array.Sort( aStringArray );

foreach (string s in aStringArray)
    System.Console.Write (“{0} : “, s);

// Output: “Apple : Banana : Cherry : Peach : ”
```
List

• Arrays have problem that you must know how many elements you want in advance
  – This is not always known
• List class is collection with variable size
  – Dynamically increases in size if needed
  – When an array reaches its capacity, need to create new array, and copy all elements from old array to new array
• Ugh!
Creating a List

List<type> listname

Example:

List<string> stringList = new List<string>(); // Create list of string.
    // Don’t forget ()

stringList.Add(“Quick”);
stringList.Add(“Brown”);
stringList.Add(“Fox”);

foreach (string s in myStringList) // Lists work with
    // System.Console.Write(“{0} “, s);
• Add elements with Add() method
• Clear() removes all elements from list
• Remove() removes first element from list
• Sort() sorts the list
• Count property: number of elements in list
Queue, Stack, Dictionary

- C# provides queue, stack, and dictionary
- Queue: first-in, first-out
  - Enqueue(), Dequeue(), Peek()
- Stack: last-in, first-out
  - Push(), Pop(), Peek()
- Dictionary
  - Holds set of key, value pairs
  - Permits lookup of a value given the key
  - Example use: extensible character attribute system
    - Keys: strings, names of attribute
    - Value: int, value of specific attribute
Chapter 4 (Classes and Objects)
Chapter 9 (Arrays, Indexers, and Collections)
from pp. 155-176
in *Programming C# 3.0*

If you are behind in the reading, you need to catch up fast.

– You are already 5-10 hours behind.
– It is hard to catch up after this point
Scaffolding for a simple XNA GSE game is created when you select a new game project in Visual C# Express

- File ... New Project ... Windows Game (3.0)
- Or File ... New Project ... Xbox 360 Game (3.0)

Can fill-in this scaffolding to create your own game

Creates a class (myGameClass) that includes
- Constructor
- Initialization
  - Initialize(), LoadContent()
- Update
  - Update game state every clock tick
- Draw
  - Create display every clock tick

**Demonstration of XNA GSE scaffolding in Visual C# 2008 Express**
XNA GSE Game Scaffolding

Microsoft.Xna.Framework.Game

- # Initialize()
- # Run()
- # Tick()

Update(gameTime);
Draw(gameTime);

myGame

- graphics: GraphicsDeviceManager
- content: ContentManager

+ myGame()
  # Initialize()
  # LoadContent(loadAllContent: bool)
  # UnloadContent(unloadAllContent: bool)
  # Update(gameTime: GameTime)
  # Draw(gameTime: GameTime)

graphics = new GraphicsDeviceManager(this);
Content.RootDirectory = "Content";
base.Initialize()
XNA GSE Game Initialization

• **Create new myGame**
  – Call to constructor, myGame()
  – myGame.run()
    1. Initializes game, then,
    2. Runs the main game loop & processes events

• **Initialization phase of run()**,
  – The following methods are called on myGame
  – Initialize()
    1. call Initialize() on parent class
    2. Initialize your game state
      1. Create player object, create enemy objects, create object to hold main game state, etc.

  – LoadContent()
    • Method used to load textures, create SpriteBatches
XNA GSE Main Game Loop

• Time elapsed between each clock tick:
  – Fixed:
    • 1/60th of a second (16.6667 milliseconds per tick)
    • myGame.IsFixedTimeStep = true
    • The default value
  – Variable:
    • Adjusts based on the time required to perform previous tick
    • myGame.IsFixedTimeStep = false

• Each clock tick
  – Run() calls Tick()
  – Tick() calls Update() then Draw()
    • You supply Update() and Draw()
Update() and Draw()

• **Update()**
  – Update the state of all objects
  – Receive input, move player avatar
  – Compute opponent AI, move opponent objects
  – Collision detection & consequences
  – Detect end-of-game or end-of-level condition

• **Draw()**
  – Re-create the on-screen scene using the up-to-date positions of player, opponent

• **Advice**
  – Avoid stuffing your entire game into the definition of these two methods
    • Methods become too big!
  – Have these methods call out to your player object, opponent objects, etc.
    • foreach (Opponent o in opponentList) o.update();
Getting a 2D Image to Appear on Screen

**LoadContent()**
1. Create a Texture
   - A bitmap image
2. Create a SpriteBatch
   - Collects all textures being drawn to screen

**Draw()**
3. Begin the SpriteBatch
4. Draw texture
   - Draw() is defined on a SpriteBatch
   - Adds texture to the SpriteBatch
5. End the SpriteBatch
   - Causes textures in SpriteBatch to be drawn to screen
Creating a Texture

• Create an instance of ContentManager
  – XNA GSE scaffolding does this for you
  – Content = new ContentManager(Services) in constructor

• Call Load<T> on ContentManager
  – For 2D sprites, type T is “Texture2D”
  – This loads an art asset that has been created by the Content Pipeline
    • In our case, conversion of a 2D bitmap image in PNG or JPG into XNA internal bitmap format
  – Give the pathname of the bitmap image (e.g., in PNG or JPG) to load
    • Path is relative to the “Content” directory of the Visual C# project
    • Note: normally need to escape slash in a string “\” → \\
    • Can put “@” at beginning of string to make string “verbatim”
      – No need to escape slashes in this case
      – “\images\\” is the same as @“\images\”
Example of creating a texture

• Create new bitmap image
  – In GIMP, Photoshop, etc.
  – Save to disk, then copy over to Visual C# project
    • Copy to
      Visual Studio 2008\Projects\{your project}\{your project}\Content
    • Go to Solution Explorer in Visual C# Express
    • Right click on **Bolded Project Name**
    • Add → Add Existing Item
    • Pick filename of new bitmap image file
    • Will now appear in the project file list
    • Verify that Content Pipeline processed file by building solution (F6)
      – Build > Build Solution

• Create a Texture2D, then load the bitmap image via the content manager:

```
Protected Texture2D m_bullet = null;
m_bullet = Content.Load<Texture2D>("mushi-bullet");
```
Once a texture has been made, how does this get displayed?

- Create a SpriteBatch
- Within a clock tick, begin() the batch
  - Prepares the graphics device for drawing sprites
- Draw() the texture as part of the batch
- End() the batch
  - Causes textures to be drawn to the screen
  - Restores device to how it was before the batch
- Typically this is performed in your game’s Draw() method
protected override void LoadGraphicsContent(bool loadAllContent)
{
    if (loadAllContent)
    {
        m_batch = new SpriteBatch(graphics.GraphicsDevice);  // Initialize the sprite batch
        m_bullet = content.Load<Texture2D>("mushi-bullet");  // Create Texture2D
    }
}

protected override void Draw(GameTime gameTime)
{
    Vector2 loc = new Vector2(120, 120);  // Create Vector2 to give location of Texture2D
    m_batch.Begin();  // Start the batch
    m_batch.Draw(m_bullet, loc, Color.White);  // Add Texture2D to batch. Not yet on screen.
    m_batch.End();  // Now Texture2D is drawn to screen.
}

• Draw() inside SpriteBatch is heavily overloaded
  – 7 different choices
Tinting Sprites

• On previous slide, used Color.White in the Draw() method
  – This gives the tint of the sprite
  – White indicates use of the original colors
  – Can choose any other color to tint the sprite
    • Visual C# Express gives list of predefined colors
    • Can also defined a Vector3 to give RGB values

protected override void Draw(GameTime gameTime)
{
    Vector2 loc = new Vector2(120, 120);              // Create Vector2 to give location of
    Texture2D
    m_batch.Begin();                                         // Start the batch
    m_batch.Draw(m_bullet, loc, Color.Red);       // Add Texture2D to batch. Has red
tint.
    m_batch.End();                                           // Now Texture2D is drawn to screen.
Transparent Sprites

- It is possible to make a sprite partially opaque
  - Colors have RGB, and Alpha (opacity)
  - Use Vector4 to represent this
  - Create color by passing Vector4 into constructor

```csharp
protected override void Draw(GameTime gameTime)
{
    Vector2 loc = new Vector2(120, 120);  // Create Vector2 to give location of Texture2D
    Vector4 v4Color = new Vector4(1.0f, 1.0f, 1.0f, 0.5f);  // Create Vector4 to create color w/opacity
    Color color = new Color(v4Color);  // Create color from v4Color
    m_batch.Begin();  // Start the batch
    m_batch.Draw(m_bullet, loc, color);  // Add Texture2D to batch. Is partially opaque
    m_batch.End();  // Now Texture2D is drawn to screen.
}
```
Other Sprite features

• Depth ordering
  – Draw some sprites in front of (behind) others to give depth of field effect
• Rotation
  – Can rotate sprite image to a specific angle
• Scaling
  – Can make sprites larger or smaller
• Animated sprites
  – Need to write code that cycles the animation yourself
  – Variant of batch.Draw() where you specify a specific rectangle within a Texture2D to draw
• Warp effects
  – Deform the Texture2D before placing on screen