Chapman University
CPSC 340 - Spring 2011

Class #2
September 8, 2011
William Wood Harter
Adjunct Professor

Second Exercise

- We left of last week in the middle of this exercise. Some review.
- We are going to create our own button
- We will use images to draw the button in the various states.
- We will capture mouse location and understand when the button was pressed.

Properties of buttons

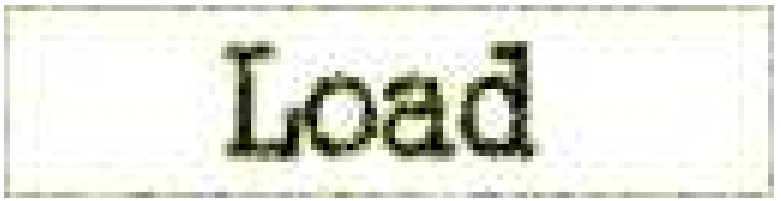
- Location
- A set of images
- Different states
  - Normal
  - Up
  - Down
  - Down but not over – can just use normal

Vector

- In XNA, a vector is 2 or three floating point numbers
- Vector2 is X,Y
- Vector3 is X,Y,Z
- Vector2 myvect = new Vector2(10.0f, 20.0f);
- It is simply a location on the screen.
- Vector3 is simply a location in space.

Drawing Images

- XNA is a 2D and 3D platform
- Loading an image is an important skill
- Book Chapter 3
  - Recipe 3-1 Display 2D Images
- What is a SpriteBatch?
- Create a button image with Photoshop
  - Save it as a png image



Iterative development

- Always take small iterative steps
  - Make small changes
  - Test the small changes
  - Move onto the next small change
- Why?

Texture Variable and LoadContent

- Declare the texture variable
- Make changes to the LoadContent method to load the image.

```
/// </summary>
public class Game1 : Microsoft.Xna.Framework.Game
{
    GraphicsDeviceManager graphics;
    SpriteBatch spriteBatch;

    Texture2D ButtonLoadNorm;
```

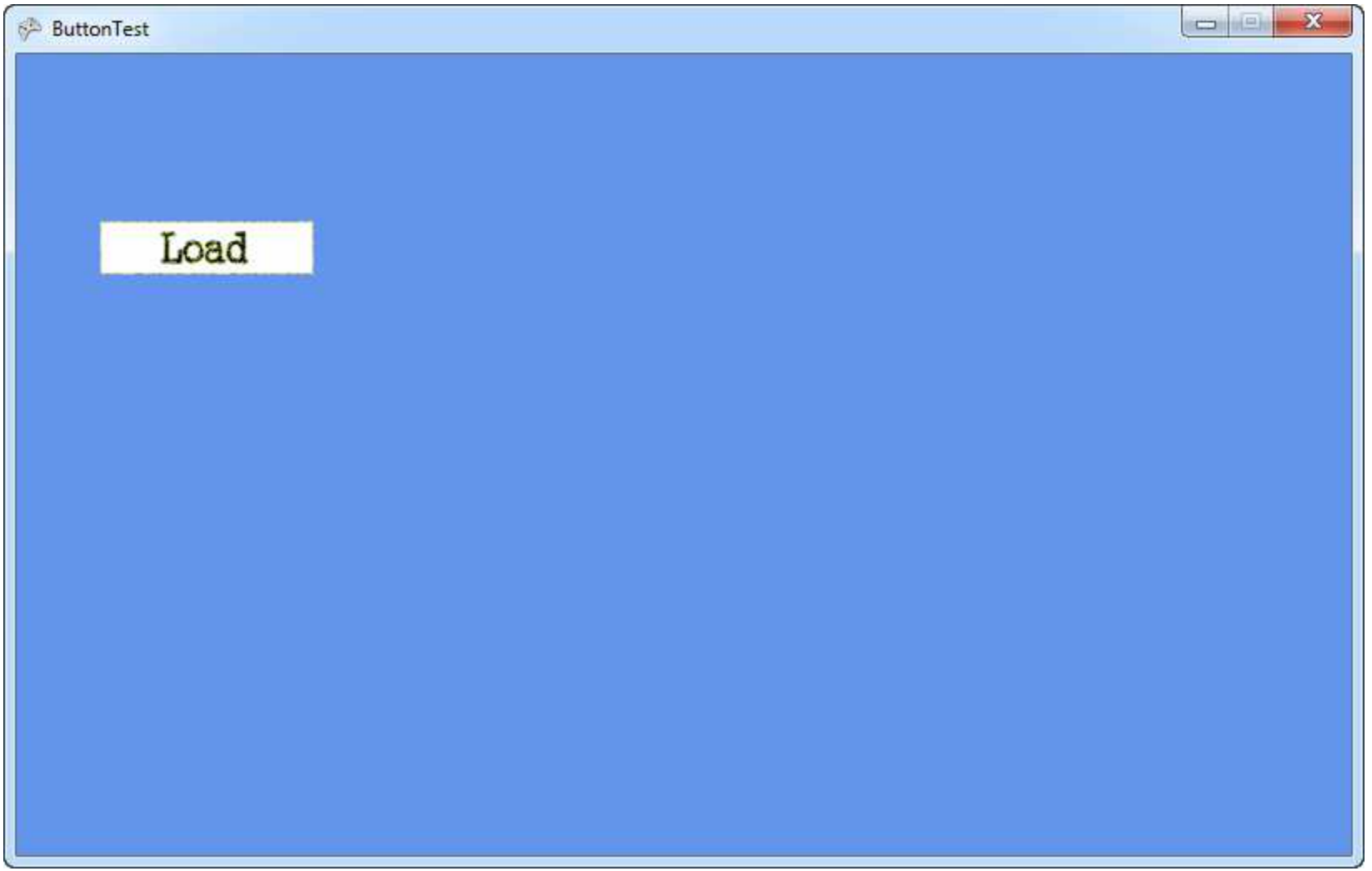
```
/// all of your content.
/// </summary>
protected override void LoadContent()
{
    // Create a new SpriteBatch, which can be used to draw textures.
    spriteBatch = new SpriteBatch(GraphicsDevice);

    ButtonLoadNorm = Content.Load<Texture2D>("button-norm");
}
```

Drawing the image

- You draw the image like this.

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

    spriteBatch.Begin();

    spriteBatch.Draw(ButtonLoadNorm, new Vector2(10.0f,20.0f), Color.White);

    spriteBatch.End();

    base.Draw(gameTime);
}
```

Your second first game



Improvements

- What is wrong with the new Vector(50,100) in the context of the Draw method?

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

    spriteBatch.Begin();

    // draw the button
    spriteBatch.Draw(ButtonLoadNorm, new Vector2(50, 100), Color.White);

    spriteBatch.End();

    base.Draw(gameTime);
}
```

Fix the overactive Vector

- It's time to move the location of the image to a variable.
- Add a variable for the location

```
Texture2D texButtonLoadNorm;
Vector2 vButtonLoadLoc = new Vector2(50, 100);
```

- Then use that variable when drawing the button.

```
spriteBatch.Draw(texButtonLoadNorm, vButtonLoadLoc, Color.White);
```

Variable scope

- All variables have scope
- Scope tells where a variable is active and available
- The variables you have been adding have had object scope. Meaning they are part of an object. One for each object of that type.
- Some variables have a method scope and disappear when the method/function returns, ie. passed parameters.

More variable scoping

- Variables that are scoped to an object can only be accessed when an instance of that object has been created.

```
class MyClass
{
	public int myvar = 4;
}

function xyz()
{
	MyClass mc = new MyClass();
	Console.WriteLine(mc.myvar);
	mc.myvar = 8;
}
```

More variable scoping

- Some variables can be scoped to the class. Meaning they are accessible even when no objects are created. They are like global variables.
- These are called static variables

```
public class MyClass
{
	public static int currentState = 0;
}
…

MyClass.currentState = 4;
```

Some Coding Conventions

- I like to use a Hungarian notation where the variable type is identified in the name.
  - Vector vDirection;
  - int iScore;
  - Texture2D texButtonImage;
- Lots of comments.
  - Leave comments not only for others, but for yourself, you wont remember what it did a month from now.
- Descriptive file and method headers.
  - User /** to create tags for auto documentation at the beginning of functions.

Some Coding Conventions

- Align those brackets.
  - Brackets on the end of lines are only done to save space in books. Always put your brackets on their own line and make sure they line up.
  - Current editors do a lot of this work for you. Lucky you.

Yes

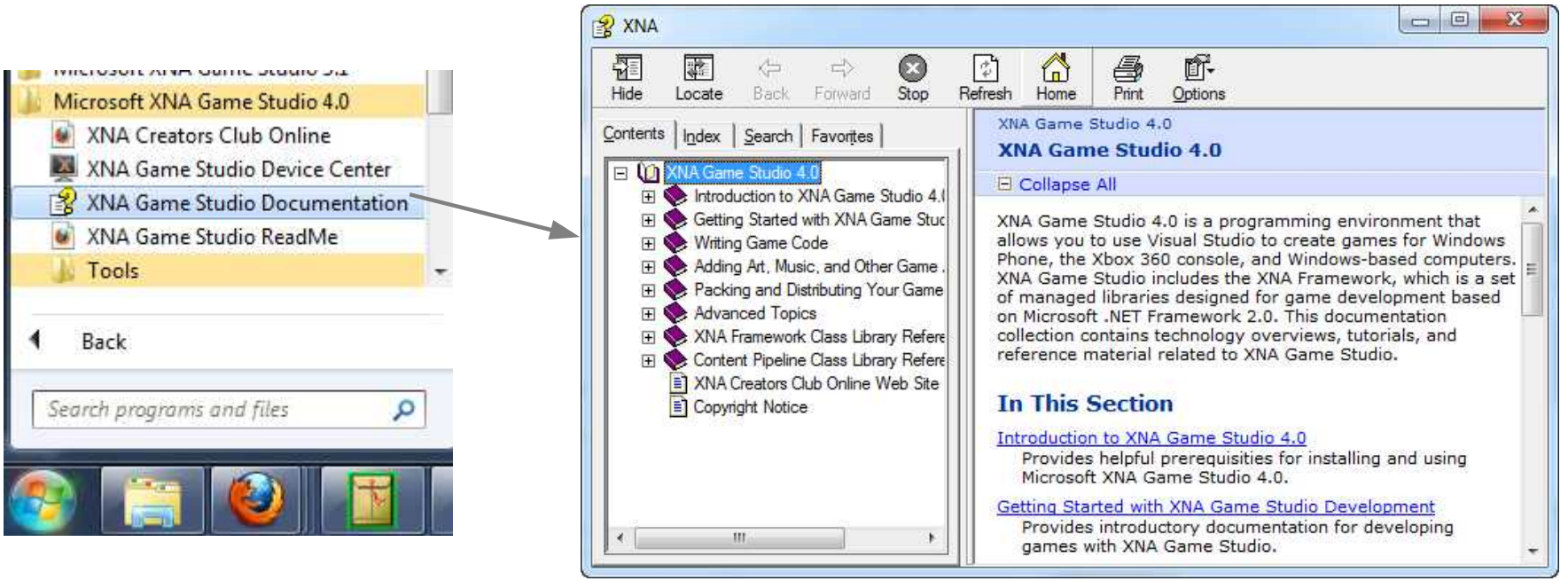
```
void MyFunction()
{
   Xyz = 40;
}
```

No

```
Void MyFunction() {
   Xyz = 40;
}
```

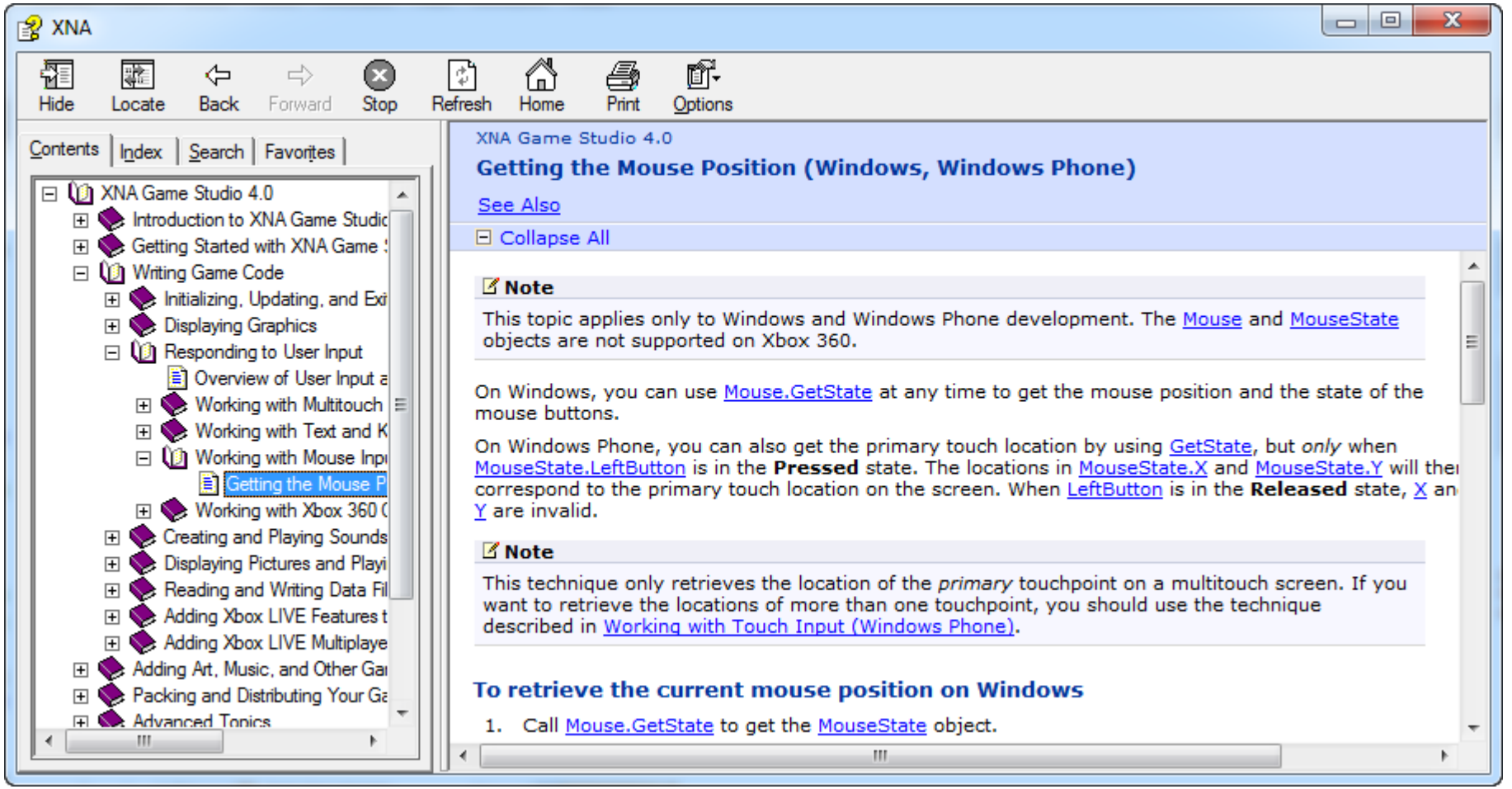
Detecting a mouse over

- How do you detect a mouse over?
- Update method?
- How about using the XNA help system. What? No one really uses 'help' do they?



Detecting a mouse over

- It's all there.



Detecting a mouse over

- In the update method, you can get mouse state which includes the X,Y location and compare it to the location of your button.

```
            MouseState ms = Mouse.GetState();

            // on Windows, the current state of the mouse cursor can be obtained at any time.
#if WINDOWS
            curMousePos.X = ms.X;
            curMousePos.Y = ms.Y;
#endif

            // if the mouse button is held down (or the touchscreen is pressed for phone), drop
            // a piece of food at the location given.
            if (canDrop && ms.LeftButton == ButtonState.Pressed)
            {
                foodLocations.Add(new Vector2(ms.X, ms.Y));
                canDrop = false;
            }
            else if (ms.LeftButton == ButtonState.Released)
            {
                // wait until the button is released before allowing the player to drop another
                // piece of food.
                canDrop = true;
            }
```

Comparison?

- How do I compare?
- All languages include a handy thing called an if statement. Try something like this.

```
if ((ms.X > vButtonLoadLoc.X) &&
   (ms.X < vButtonLoadLoc.X+132) &&
   (ms.Y > vButtonLoadLoc.Y) &&
   (ms.Y < vButtonLoadLoc.Y+32))
{
      Console.WriteLine("I'm over!!!");
}
else
{
      Console.WriteLine("I'm not over!!");
}
```

Performance and Console.Write

- If you have performance issues, first make sure your code isn't using Console.Write.
- Even a few calls to Console.Write can bring an application to a crawl.

Comparison

- If statement operations.

Symbol	Use
&&	AND
||	OR
>	Greater Than
<	Less Than
!=	Not Equal
>=	Greater Than and Equal
<=	Less Than and Equal

Changing the button image

- You need a way to change which image is showing when the mouse is over. A boolean value is a great way to switch between two images.
  - bool isOverLoad = false
  - bool isLoadPressed = false;

```
if ((ms.X > vButtonLoadLoc.X) &&(ms.X < vButtonLoadLoc.X+132) &&
   (ms.Y > vButtonLoadLoc.Y) && (ms.Y < vButtonLoadLoc.Y+32))
{
      isOverNorm = true;
}
else
{
      isOverNorm = false;
}
```

Changing the button image

- In the draw method you have to draw one image or the other

```
// draw the button
if (isOverLoad)
{
    spriteBatch.Draw(texButtonLoadOver, vButtonLoadLoc, Color.White);
}
else
{
    spriteBatch.Draw(texButtonLoadNorm, vButtonLoadLoc, Color.White);
}
```

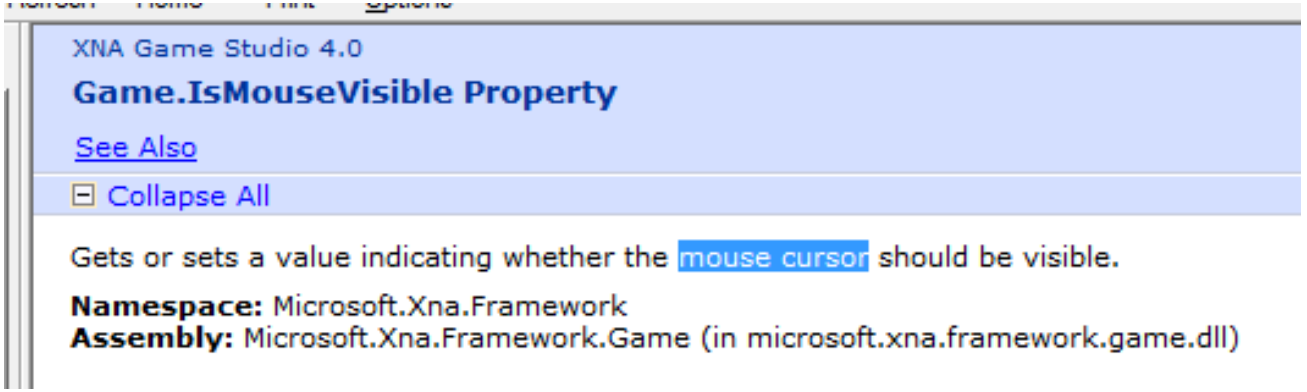
Texture Size

- In the previous example we hard coded the texture size.
- Did you know you can get that value programatically?

```
if ((ms.X > vButtonLoadLoc.X) &&
    (ms.X < vButtonLoadLoc.X + textureButtonLoadNorm.Width) &&
    (ms.Y > vButtonLoadLoc.Y) &&
    (ms.Y < vButtonLoadLoc.Y+ textureButtonLoadNorm.Height))
{
      isOverNorm = true;
}
else
{
      isOverNorm = false;
}
```

Cursor

- Did you notice the cursor doesn't show. Me too and I had to look it up in the XNA help



- In Initialize I now call
- this.IsMouseVisible = true.
- Why this? Did you notice that the Game1 class for this game you are creating inherits type Game?

Logic for button down

- Brain pain here. When the mouse is down over the button, you have to keep the mouse down state until the mouse is then released.
- You only show the button down image when the mouse is still over the image and you don't want other buttons to show their down state until the after the mouse is released again.

Quick Detour – Keyboard Input

- This might seem simpler to deal with keyboard input.
- I want to know when the a key is pressed and let go. Put the following code in your Update method.

```
KeyboardState ks = Keyboard.GetState();
if (ks.IsKeyDown(Keys.A))
{
    Console.WriteLine("A key is down");
}
```

- What is wrong with this?

Keyboard Input

- We know when the key is down, but we only want to know when it went down, right?
- We use a boolean (method variable)

```
bool isAKeyDown = false;
```

```
KeyboardState ks = Keyboard.GetState();
if ((ks.IsKeyDown(Keys.A)) && (!isAKeyDown))
{
    Console.WriteLine("A key pressed");
    isAKeyDown = true;
}
else if ((!ks.IsKeyDown(Keys.A)) && (isAKeyDown))
{
    Console.WriteLine("A key released");
    isAKeyDown = false;
}
```

Variable Naming Thought

- When naming boolean variables, make sure to use positive names.
- ItIsDown
- ItIsOn
- Negative names will drive you insane.
- ~~ButtonIsOff = true~~
- ~~PlayerNotJumping = true~~

Button clicked

- How do I tell if the button is clicked?
- If the button state is down.
  - Meaning the mouse was pressed while over the button.
  - And the mouse is now up and still over the button.
  - Then the mouse was clicked.
  - You'll then also need to reset all the button state flags for that button.

Good place for a function

- Agree that the following code is ugly.
- Agree we are going to do this a couple of times.

```
if ((ms.X > vButtonLoadLoc.X) &&
    (ms.X < vButtonLoadLoc.X + texButtonLoadNorm.Width) &&
    (ms.Y > vButtonLoadLoc.Y) &&
    (ms.Y < vButtonLoadLoc.Y + texButtonLoadNorm.Height))
```

Create a function called OverArea

```
/// <summary>
/// Returns whether a position is witin a given area
/// </summary>
/// <param name="pos"></param>
/// <param name="origin"></param>
/// <param name="width"></param>
/// <param name="height"></param>
/// <returns></returns>
private bool OverArea(int posX, int posY, Vector2 origin, int width, int height)
{
    if ((posX > origin.X) &&
      (posX < origin.X + width) &&
      (posY > origin.Y) &&
      (posY < origin.Y + height))
    {
        return true;
    }
    return false;
}
```

```
MouseState ms = Mouse.GetState();
if (OverArea(ms.X, ms.Y, vButtonLoadLoc, texButtonLoadNorm.Width, texButtonLoadNorm.Height))
{
    isOverLoad = true;
}
else
{
    isOverLoad = false;
}
```

Check for mouse down

```
if ((!isMouseDown) && (ms.LeftButton == ButtonState.Pressed))
{
    if (isOverLoad)
    {
        isLoadPressed = true;
    }
    isMouseDown = true;
}
```

Check for mouse release

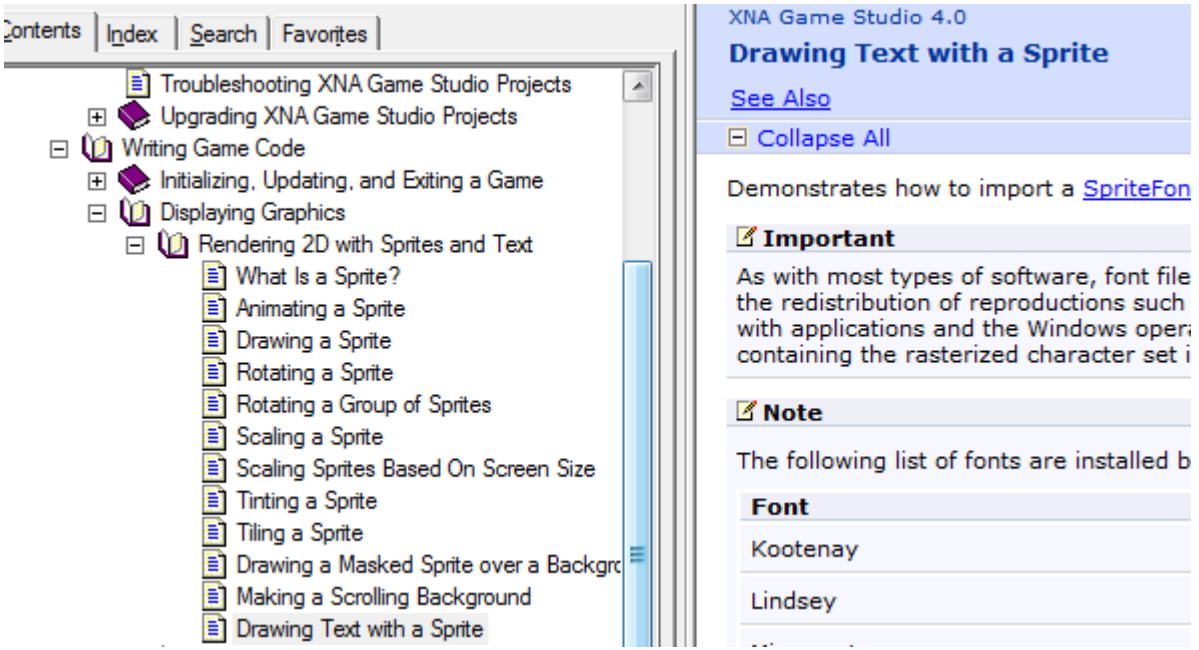
```
else if ((isMouseDown) && (ms.LeftButton == ButtonState.Released))
{
    if ((isOverLoad) && OverArea(ms.X, ms.Y, vButtonLoadLoc, texButtonLoadNorm.Width, texBu
    {
        Console.WriteLine("Load button pressed");
    }

    isMouseDown = false;
}
```

Homework #3

- Continuing #2 finish the two buttons
- New Game
- Exit
- Each buttons should print something to the console when clicked.
- Each button should have an up, over and down button and function properly.

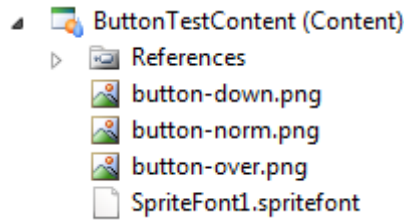
Drawing Text

- How do you figure out how to draw text?
- Try looking at the XNA help!



Drawing Text To The Screen

- Or look at your book. Exercise 3-5 p#190
- Add a font to the content project



- Get a method variable reference to that font

```
myFont = Content.Load<SpriteFont>("SpriteFont1");
```

- Create and draw the string

```
string eTime = "Elapsed Seconds: " + gameTime.TotalGameTime.Seconds.ToString();
spriteBatch.DrawString(myFont, eTime, new Vector2(50, 20), Color.Tomato);
spriteBatch.End();
```

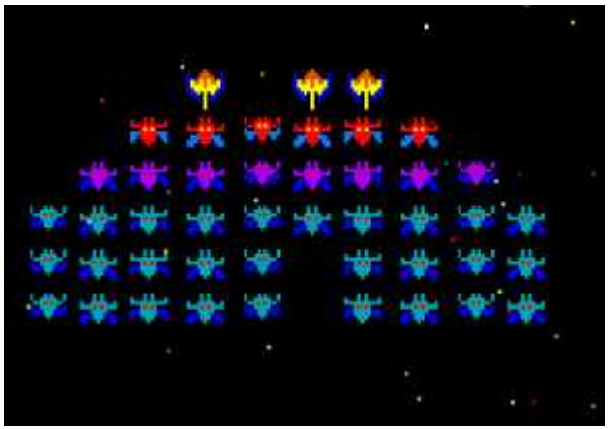
What Will The Width Be?

- You can find the pixel width of a given string
  - myFont.MeasureString(“Hello World”);
- How could you use this?

Homework #4

- Use the code from book section 3-5
- Show the amount of time elapsed since the application started.
- Show it right justified in the upper right corner.
- Don't have it move when the values go from 9-10, 99-100 or 99-100. Leave 4 digits space using Font.MeasureString.
- For more details, see the course website.

Classes: Galaxian



- Let's discuss some more about classes and inheritance.
- What are some of the classes involved in describing the enemies in Galaxian?
- BlueShip, PurpleShip, RedShip, YellowShip?

Galaxian Ship Properties

- What are some of the properties of the ships?
  - Location
  - Image
  - Image Orientation
  - Moving?
  - Size?
  - Exploding?
  - Formation – with another ship.

Galaxian Ship Behaviors

- What are some of the behaviors of the ships?
  - Draw
  - Move
  - Collision

Galaxian Ship Base Class

- Can you see that all the ships share many of the properties and behaviors, but may handled some of them differently?
- You would have an EnemyShip base class.

```
public class EnemyShip
{
    protected Vector2 location;
    protected Texture2D image;
    protected bool flying;

    public virtual void Draw(GameTime gameTime)
    {
    }

    public virtual void Update(GameTime gameTime)
    {
    }
}
```

Ship Types

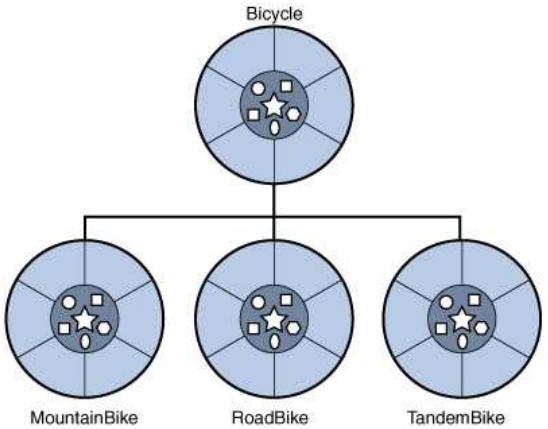
```
public class BlueShip : EnemyShip
{
    public override void Update(GameTime gameTime)
    {
        // I fly straight down
        if (flying)
        {
            location.Y = location.Y - 1;
        }
    }
}

public class RedShip : EnemyShip
{
    public override void Update(GameTime gameTime)
    {
        // I fly straight down
        if (flying)
        {
            location.Y = location.Y - 1;
        }
    }
}
```

Ships At Runtime

```
public class MyGame
{
    // hypothetical game init
    EnemyShip[] ships = new EnemyShip[42];
    private void GameInit()
    {
        ships[0] = new BlueShip();
        ships[1] = new RedShip();
    }

    private void Update(GameTime gameTime)
    {
        int i;
        for (i = 0; i < ships.Length; i++)
        {
            ships[i].Update(gameTime);
        }
    }
}
```

Class inheritance

- Class inheritance is something that can't be explained. You have to use it and figure it out. We'll continue to talk more about it as we continue, but I wanted to tell you it exists and it is very powerful and important.
  - http://download.oracle.com/javase/tutorial/java/concepts/inheritance.html
- This type of object would allow you to have an array of Bicycles not caring what sub-types actually exist. This is handy once you start to feel the pain of working with objects of different types. Remember that thing about “Strongly Typed”?



Create your own class

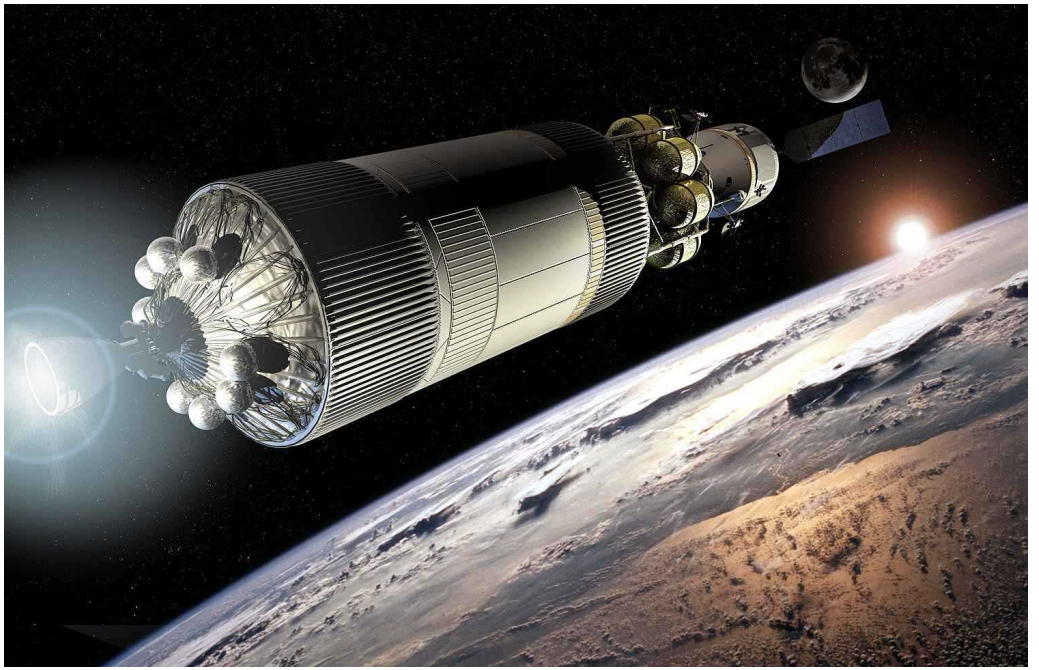
- Classes are easy to create. They simply encapsulate both data and functions. The functions associated with a class are called methods.

```
public class MyClass
{
        public int xyz;

        Void printXYZ()
        {
                Console.WriteLine(xyz);
        }
}
…

MyClass mc = new MyClass()
mc.printXYZ();
```

When to create new classes

- Create new classes for functionality that you will use over and over again.
  - A Car in a game
  - A Button
- Create new classes for functionality that you might use in your next game as well as the game you are working on.
  - An animation system
  - A game saving system
  - A button system

X Prize Homework



- If you feel compelled to work on a some really hard C# problems to help advance the PantherEngine, let me know. I have some bugs that need looking into. Lua interpreter debugging anyone?

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