UNIVERSITY OF Hull

Getting Started with XNA

Rob Miles

Department of Computer Science

XNA

- XNA is a framework for writing games
- Includes a set of professional tools for game production and content management
- It works within Visual Studio
  - There are XNA project types in the same way we have Windows Forms project types

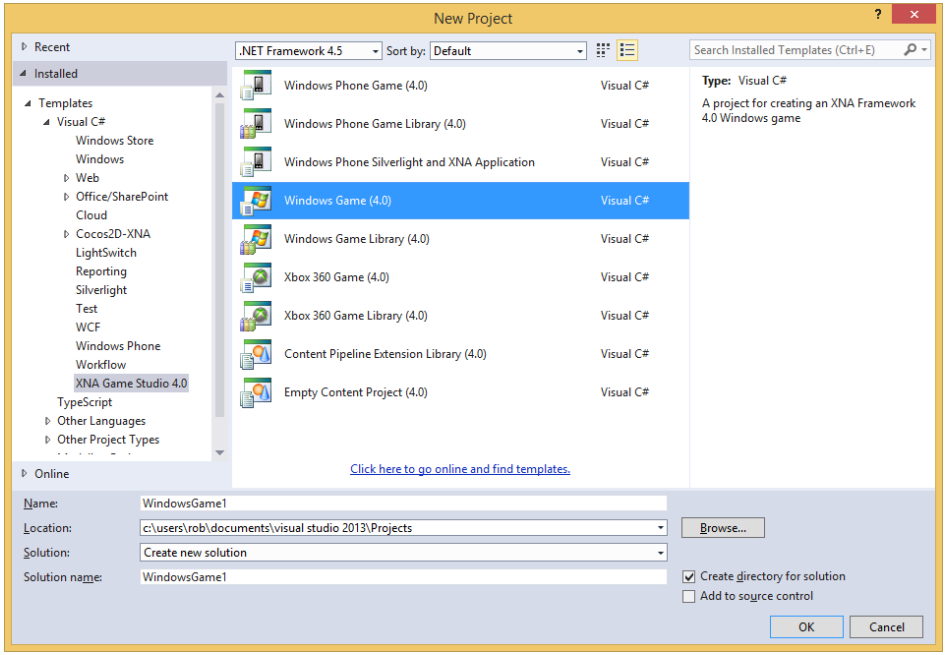
Running Games

- Games can be run on the PC, Xbox 360 or Windows Phone device
- To run games on the Xbox 360 or sell Windows Phone games you need to be a member of the Creators Club
  - You can get free membership from DeamSpark, there is a link on the Sharepoint site
  - www.dreamspark.com
- For the practical work you will be running code for the PC
  - It is easy to retarget the work to Xbox or Windows Phone

XNA Versions

- The latest version of XNA is 4.0
  - This works with Visual Studio 2010, 2012 and 2013
- You can obtain this from:
  https://msxna.codeplex.com/releases
  - Follow the instructions in the Readme.txt to install it
- Not all installations of Visual Studio on campus have the XNA components installed
- The Fenner Computer Suite and the labs in the Robert Blackburn Building have XNA

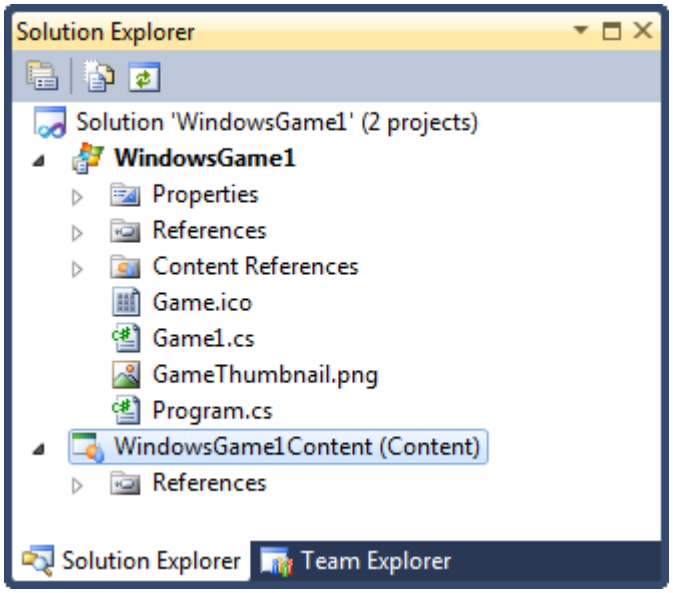
Creating a Game



- Visual Studio 2012 with XNA installed provides Windows Game project types for Windows Phone, Windows PC and Xbox 360
- We are going to use the Windows Game project
  - The others may not work

The Game Project

- The solution explorer shows the items that make up our game project
- At the moment there are a couple of class files which are created automatically
- The solution will also contain any content that we add to the game project



XNA and Programs

- XNA works by using classes and objects
  - You create instances of objects
  - You use objects built into the framework
- This lets you create games
- It is also a very good example of how you construct a set of software resources and use them
  - This is a good way to learn how to do this

Running the Game

- We can build and run our game now if we like
- The initial application contains code which starts our game:

```
static void Main(string[] args)
{
    using (Game1 game = new Game1())
    {
        game.Run();
    }
}
```

First Game Display

- At the moment all our game does is display a blue screen
- This is because the behaviour of the `Draw` method in a brand new project is to clear the screen to blue

What a Game Does When it Runs

1. Initialise all the resources at the start
   - fetch all textures, models, scripts etc
2. Repeatedly run the game:
   1. Update the game engine
      - read the controllers, update the state and position of game elements
   2. Draw the game environment
      - render the game elements on the viewing device

XNA Game Class Methods

```
partial class PongGame: Microsoft.Xna.Framework.Game
{

  protected override void LoadContent
                              (bool loadAllContent)
  {
  }

  protected override void Update(GameTime gameTime)
  {
  }

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

Some C# Details....

- `partial`
  - There may be other C# source files which contain other code for this class
- `protected`
  - This member is visible in child classes, but not ones external to this class hierarchy
- `using`
  - When you leave this block, garbage collect the thing I'm using

Loading Game Content

```
protected override void LoadContent()
{
    // Create a new SpriteBatch, which can be used to draw
    spriteBatch = new SpriteBatch(GraphicsDevice);

    // TODO: use this.Content to load your game content here
}
```

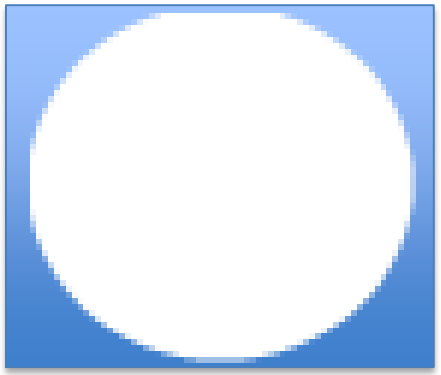
- `LoadContent` is called when our game starts
- It is where we put the code that loads the content into our game
- Content includes images, sounds, models etc.

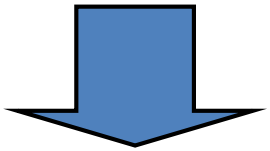
Game Content

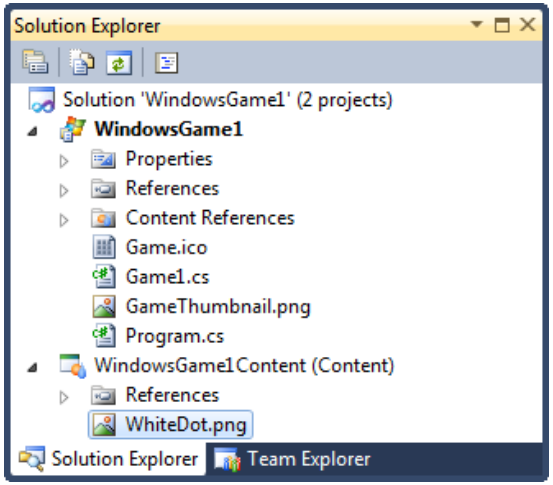
- Games are not just programs, they also contain other content:
  - Images for textures and backgrounds
  - Sound Effects
  - 3D Object Meshes
  - Scripts
  - Videos
- We need to add this content to our project
- The XNA framework provides a content management system which is integrated into Visual Studio 2008

Image Resources

- This is my Ball image
- I have saved it as a PNG file
  - This allows me to use transparency
- You can use any image resource you like
- The resources are added to the Visual Studio project
- They are held in the Content directory as part of your project







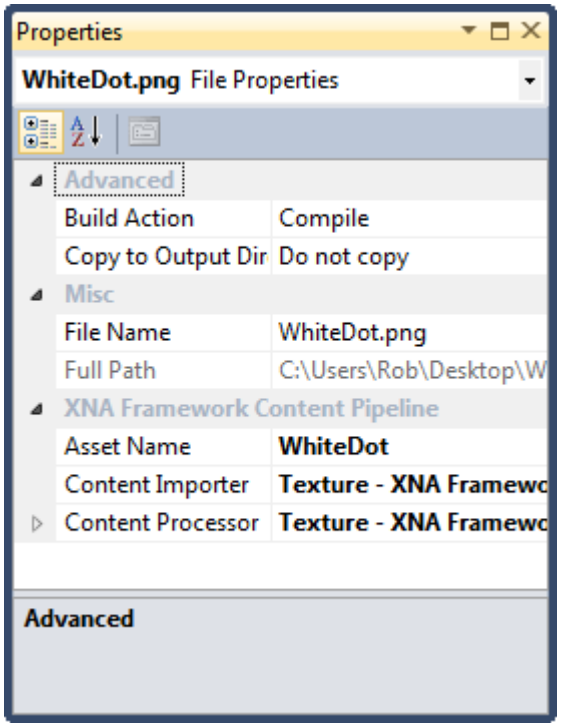
Storing the Ball Texture in the game

```
Texture2D ballTexture;
```

- XNA provides a `Texture2D` type which holds a 2D (flat) texture to be drawn on the display
- The game class needs to contain a member variable to hold the ball texture that is to be drawn when the game runs
- This variable will be shared by all the methods in the game

Using the Content Pipeline

- Each resource is given an asset name
- The `Load` method from the Content Manager provides access to the resource using the Asset Name that we gave it




```
ballTexture = Content.Load<Texture2D>("WhiteDot");
```

Loading Content into the Ball Texture

```
protected override void LoadContent()
{
    // Create a new SpriteBatch, which can be used to draw
    spriteBatch = new SpriteBatch(GraphicsDevice);

    ballTexture = Content.Load<Texture2D>("WhiteDot");
}
```

- `LoadContent` is called when our game starts
- It loads an image into the ball texture
- The content manager fetches the images which are automatically sent to the target device

XNA Game Draw Method

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

    base.Draw(gameTime);
}
```

- The `Draw` method is called repeatedly when an XNA game is running
  - It has the job of drawing the display on the screen
- A brand new XNA game project contains a `Draw` method that clears the screen to `CornflowerBlue`
- We must add our own code to the method to draw the ball

Sprite Batching

- 2D Graphics drawing is handled by a set of "sprite" drawing methods provided by XNA
- These create commands that are passed to the graphics device
- The graphics device will not want to draw everything on a piecemeal basis
- Ideally all the drawing information, textures and transformations should be provided as a single item
- The `SpriteBatch` class looks after this for us

SpriteBatch Begin and End

```
spriteBatch.Begin();

// Code that uses spriteBatch to draw the display

spriteBatch.End();
```

- The call to the `Begin` method tells `SpriteBatch` to begin a assembling a new set of drawing operations
- The call to the `End` method tells `SpriteBatch` that the there are no more operations and causes the rendering to take place
- Between the two we put our drawing code

Using SpriteBatch.Draw

```
spriteBatch.Draw(ballTexture,
    new Rectangle(
        0, 0,
        ballTexture.Width, ballTexture.Height),
        Color.White);
```

- The `SpriteBatch` class provides a `Draw` method to do the sprite drawing
- It is given parameters to tell it what to do:
  - Texture to draw
  - Position (expressed as a `Rectangle`)
  - Draw colour (expressed as an XNA `Color`)

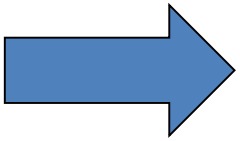
The Rectangle structure

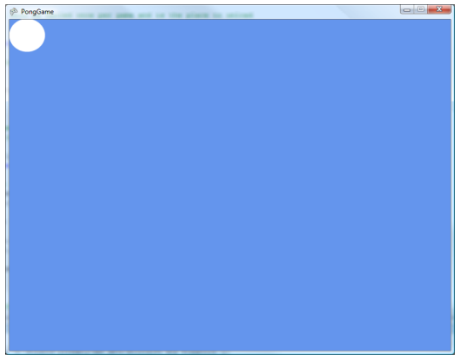
```
new Rectangle(
      0, 0,
      ballTexture.Width, ballTexture.Height),
      Color.White);
```

- `Rectangle` is a `struct` type which contains a position and a size
- The code above creates a rectangle positioned at `0,0` (top left hand corner) the same size as `ballTexture`
- We could move our ball by changing the content of the rectangle

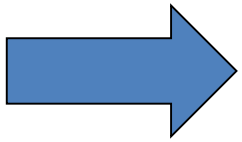
Rectangle Fun

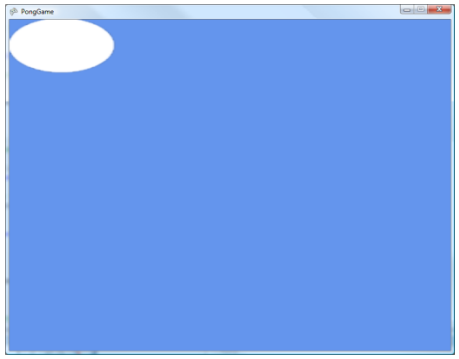
```
new Rectangle(
  0, 0,
  ballTexture.Width,
  ballTexture.Height)
```



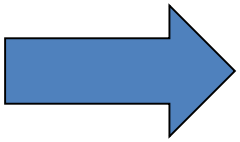


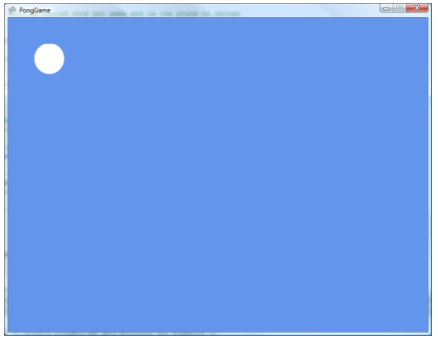
```
new Rectangle(
  0, 0,        // position
  200,100)     // size
```





```
new Rectangle(
  50, 50,       // position
  60, 60)       // size
```





Positioning the Ball using a Rectangle

```
Texture2D ballTexture;
Rectangle ballRectangle;
```

- If we want to move the ball around the screen we must add a variable to keep track of the ball position
- This is another member of the game class
- We declare it in the class so that all the methods in the game can use it
- We will initialise it in the `LoadContent` method

Screen Size and Scaling

- We need to make our game images fit the size of the screen
- We can find out the size of the screen

```
Window.ClientBounds.Width
Window.ClientBounds.Height
```

- The game has a `Window` property we use to get this
- We can use this information to scale the sprites to fit our screen

Creating the ballRectangle variable

```
ballRectangle = new Rectangle(
    0, 0,
    Window.ClientBounds.Width / 30,
    Window.ClientBounds.Width / 30);
```

- If we use this rectangle to draw our ball texture it will be drawn as a square which is a thirtieth of the width of the screen
- We can then set the position parts of the rectangle to move the ball around

Drawing with the ballRectangle variable

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

    spriteBatch.Begin();
    spriteBatch.Draw(ballTexture, ballRectangle, Color.White);
    spriteBatch.End();

    base.Draw(gameTime);
}
```

- This version of `Draw` uses the `ballRectangle` variable to position the ball on the screen

Moving the ball around the screen

- At the moment the ball is drawn in the same position each time
- To move the ball around we need to make this position change
- We need to give the game an update behaviour
- We must add code to the `Update` method in the game
- The `Update` method is where we manage the state of the "game world"

The Update Method

```
protected override void Update(GameTime gameTime)
{
    // TODO: Add your update logic here

    base.Update(gameTime);
}
```

- The `Update` method is automatically called 60 times a second when a game is running
  - 30 times a second for a Windows Phone game
- It is in charge of managing the "game world" of a game
  - In a pong game this means updating the bat and the ball positions and checking for collisions

Simple Update Method

```
protected override void Update(GameTime gameTime)
{
    ballRectangle.X++;
    ballRectangle.Y++;

    base.Update(gameTime);
}
```

- Each time the game updates the X and Y position of the ball is increased
- This will cause it to move across and down the screen
- Note that I call the base method to allow my parent object to update too

GameTime

- At the moment the `Update` method is called sixty time a second or once every 16.66 milliseconds
- We can also let the update "free run", in which case we need to know the time since the last call so we can move objects the right distance
- This is what the `GameTime` parameter is for, it gives the time at which the method was called

Summary

- XNA is a Framework of methods that are used to write games
- You create the games using Visual Studio and XNA 4.0 Game Studio
- Games have initialise, update and draw behaviours
- Game objects are held as texture objects and their position and dimensions as rectangle structures