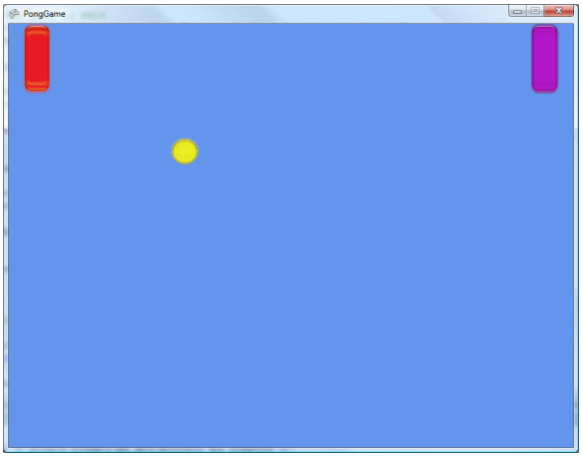
Department of Computer Science

08120 Programming 2
Week 32 Practical 2013/2014
Making Pong

You have been employed by an XNA games company to finish the development of a Pong game. The original programmer has left the company and is now working on the game of the film of the game "Need for Speed: Most Lucrative ".

Pong Game



The game is not yet playable. The ball bounces around the screen but the game does not detect collisions between the bats and the ball, and the player cannot control the bat movement.

You must add the extra code to make a playable version of the game for two players who can use the keyboard to control the game.

You can download a Zip archive containing a version of this game from the Sharepoint site

Now you can create the program itself:

1. Log in and copy the starter game project from the Sharepoint site.
2. Unzip the game files and open up the game project using Visual Studio 2012.
3. Run the game and note that the ball bounces around the screen. However, it will not interact with the paddles and the player is not able to move the paddles up and down. We are going to add all this to the game.

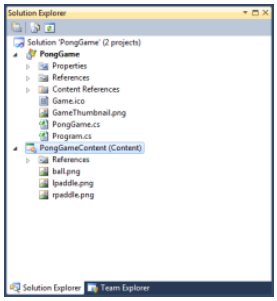
Adding a Background

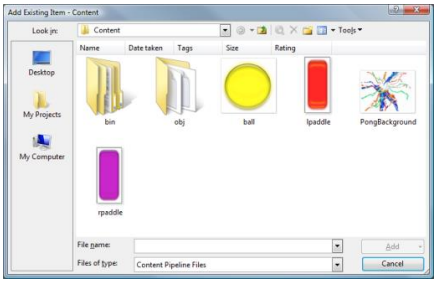
The first thing you need to do is add a background image to replace the blue screen.

Finding an Image

The image will be loaded into a texture and used to draw the backdrop. The first thing you must do is add the file to the content for the game.

1. Click on the Content item in the Solution Explorer to select it as shown on the right. Hold down CTRL+SHIFT and press A to open up the `"Add Existing Item – Content"` menu





2. Browse to your content item and select it, or select the supplied `PongBackground` image

3. Once you have selected the item, click `Add` to add it.

4. You can also add assets by simply dragging them and dropping them onto the content folder, but you need to make sure that you use the correct name when you load the asset in the program.

Creating Game Data for the Background

Now you need to create a texture variable and load the background into it.

5. Use the Solution Explorer to open the file `PongGame.cs` and find the section of the program source which contains the game data. Add a statement that declares a new `Texture2D` variable called `backgroundTexture`.

```
Texture2D backgroundTexture;
```

The texture will be drawn in a rectangle which fills the entire screen. You need to declare this variable as another item of game data.

6. Add a statement that declares a new `Rectangle` variable in the appropriate area:

```
Rectangle backgroundRectangle;
```

Setting Up the background in LoadContent

The game will set up the texture and the rectangle in the `LoadContent` method. The rectangle must be loaded from the background texture and the rectangle must be set up to be the size of the entire screen.

7. Find the `LoadContent` method and add the following statements into it:

```
backgroundTexture = Content.Load<Texture2D>("PongBackground");
backgroundRectangle = new Rectangle(
    0, 0,    // top left hand corner
    Window.ClientBounds.Width,
    Window.ClientBounds.Height); // size of screen display
```

Drawing the Background in the Draw method

The game must draw the background each time the screen is updated. The background must be the first thing that is drawn, so that all the other game items are drawn on top of it. The drawing is performed by the `Draw` method.

8. Find the `Draw` method and add the following statement into it:

```
spriteBatch.Draw(backgroundTexture, backgroundRectangle,
                                    Color.White);
```

If you now run the game you should find that the paddle and ball are drawn on the background. Because these items are slightly transparent you should be able to see the background through them.

Controlling the Bats with the Keyboard

The game now looks a little better, but you need to add code to control the bat movement from the player. The program can 'move' the bat by changing the X or Y position of the rectangle where the bat is drawn. The game is going to use the keyboard to control this. You are going to add code to the `Update` method to check the keyboard and change the bat position if particular keys are pressed.

```
KeyboardState keystate = Keyboard.GetState();

if (keystate.IsKeyDown(Keys.A))
{
    lPaddleRectangle.Y -= lPaddleSpeed;
}
```

The code above will cause the left hand paddle to move up when the A key is pressed on the keyboard. It creates a variable that holds the present state of the keyboard and then calls the `IsKeyDown` method on that to see if a particular key is pressed. Because the `Update` method is called 60 times a second the position of the paddle will be repeatedly updated if the key is held down. The speed of the paddle is controlled by the value in the variable `lPaddleSpeed`.

1. Create a member variable called `lPaddleSpeed` and set it to a sensible value.
2. Find the `Update` method and add the above statements into it.
3. Run the game and note that you can make the left paddle move up by holding the A key down. (Remember that Y is zero at the **top** of the screen).
4. Now add code that tests for `Keys.Z` to move the left hand paddle down and `Keys.K` and `Keys.M` for up and down for the right hand paddle. Note that you don't need to create another `keystate` value, you can use the one that was set in the first line. You will however need an `rPaddleSpeed` variable.
5. You also need a way for the player to stop the game by pressing a key. Add the following test to the `keystate` variable. This will exit the game if the `Esc` key is pressed on the keyboard:

```
if (keystate.IsKeyDown(Keys.Escape))
{
    this.Exit();
}
```

Bat and Ball Collisions

Next you want to detect when the bat and the ball collide, so that the ball can be made to "bounce" off the bat. The `Rectangle` type provides a method called `Intersects` which returns true if two rectangles intersect, i.e. one collides with the other. The game can use this to make the ball "bounce" off the paddle:

```
if (lPaddleRectangle.Intersects(ballRectangle))
{
    ballXSpeed = -ballXSpeed;
}
```

This code will cause the ball to reverse direction if it "hits" the bat.

1. Add the above statements into the `Update` method and run the program. Note that the game now bounces the ball off the left hand paddle.
2. Add the code to make the ball bounce off the Right hand paddle.

Detecting Scores

In the Pong game the player must stop the ball from hitting the back wall. If this happens the other player scores a point. At the moment the game does not detect this, although the statements to test for collisions between the ball and the back wall are present in the program.

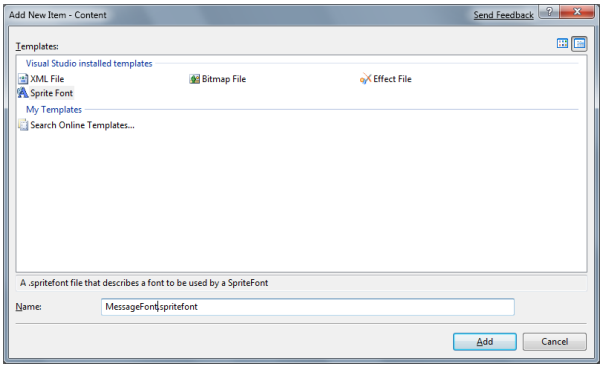
The game must maintain two variables, `leftScore` and `rightScore`. When either value reaches 5 the game is over, and the player with the score of 5 has won.

1. Add integer two variables to the game to keep track of the score.
2. Change the `Update` method so that the appropriate scores are updated when the ball hits the back wall.
3. Add another test so that when a score reaches 5 the game automatically ends.

Displaying Scores

At the moment the players can't see the scores. The game should display the score values on the screen. To do this you will need to create a `SpriteFont` from a font resource and use this to draw the score values on the screen. Fonts are added in just the same way as other resources, but rather than use an existing font you will have to create a new one.

1. Right click on the Content item to add a new item as you did in step 4, but add a `New Item` instead of an existing one. This will open up the "`Add New Item – Content`" dialog.



2. Select `SpriteFont` from the available templates and give the new item the name `MessageFont`, as shown above. This will add a new font to the Content for the game and open the font description automatically.
3. You will need to change the name of the font in use, and the size of the font so that it is displayed correctly. In the `MessageFont.SpriteFont` file make the following changes to the `FontName` and `Size` elements as below:

```
<FontName>Arial</FontName>

<!--
Size is a float value, measured in points. Modify this value to change
the size of the font.
-->
<Size>30</Size>
```

4. Next you have to create a variable in the Game Data to hold the font. The variable should be declared in the Game Data part of the `PongGame` class.

```
// Message Font
SpriteFont messageFont;
```

5. As with other resources, the font must be loaded by the Content Manager when the game starts. Add the following line to the `LoadContent` method in the `PongGame` class:

```
messageFont = Content.Load<SpriteFont>("MessageFont");
```

6. Finally you need to use the font to draw the text of the score. You do this in the `Draw` method, after the background has been drawn:

```
spriteBatch.DrawString(
    messageFont,            // font to use
    "Hello World",          // string to display
    new Vector2(100, 20),   // position of the text
    Color.Black);           // Colour of the text
```

7. If you run the game you will see that the text is displayed on the screen.
8. You can now add score display to your program.

Finishing the Pong Game

You can now create a playable Pong game with the following properties:

- The game is started by pressing the space bar. When the game starts the scores are set to 0 and the bats are placed in the centre of the screen You can check for this using the following test:

```
if (keystate.IsKeyDown(Keys.Space))
```

- The score is displayed on the top of the screen.

- When one of the players reaches a score of 5 the game is over and an appropriate message is displayed on the top of the screen. Once a score of 5 has been reached the paddles will no longer respond to the keyboard (in a real game the program would be waiting for the player to put some more money in the machine). The only way to start a new game is to press the Space bar again.

Once you have your advanced game working you should show it to a demonstrator.

Rob Miles

March 2014