XNA (2D) Tutorial

Pong

IAT410

Creating a new project

1. From the Start Menu, click **All Programs**, then the **Microsoft XNA Game Studio Express** folder, and finally **XNA Game Studio Express**.
2. When the Start Page appears, click the **File** menu, and then click **New Project**.
3. In the dialog box that appears, choose **Windows Game** and type a title for your project **MyPong** in the **Name** box. Type a path where you'd like to save your project in the **Location** box. Then click **OK**.

View the code

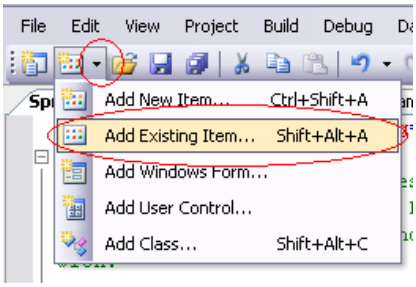
Some of the hard work has already been done for you. If you build and run your game now, the GraphicsDeviceManager will handle setting up your screen size and rendering a blank screen. Your game will run and update all by itself. It's up to you to insert your own code to make the game more interesting.

Much of the code to start and run your game has already been written for you, and all have places for you to insert your code.

- The **Initialize** method is where you can initialize any assets that do not require a GraphicsDevice to be initialized.
- The **LoadGraphicsContent** method is where you load any graphical assets such as models and textures.
- The **UnloadGraphicsContent** method is where any graphical assets can be released. Generally, no extra code is required here, as assets will be released automatically when they are no longer needed.
- The **Update** loop is the best place to update your game logic: move objects around, take player input, decide the outcome of collisions between objects, and so on.
- The **Draw** loop is the best place to render all of your objects and backgrounds on the screen.

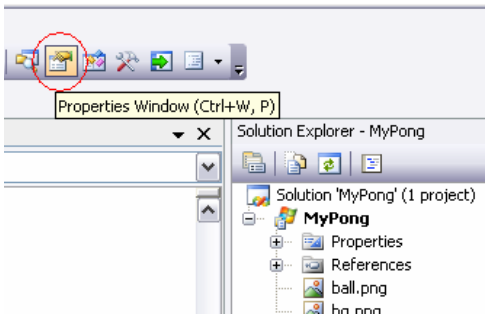
Adding sprites

1. Make sure you can see the Solution Explorer for your project on the right side of the window. If you cannot see it, click the **View** menu, and then click **Solution Explorer**. When it appears, you will see files associated with your project in a tree structure.
2. Click an arrow next to Add New Item icon, then click **Add Existing Item**.



Browse to your graphic (you need to import four graphics: ball, two paddles, and background. You can either create your own or reuse the ones you downloaded from the webCT last week). If you can't see any files, make sure you

change the **Files of type** selection box to read **Content Pipeline Files**. Click the graphic files, then click **Add**. Entries for the graphic files will appear in Solution Explorer.
When you add a graphic file, it is automatically added to the XNA Framework Content Pipeline, which will allow you to quickly and easily load the graphic in your game.
3. Open the Properties window by pressing F4, or by clicking **Properties Window** icon.



Click the entry for the graphic in the Solution Explorer. In the **Properties** window, look for the "Asset Name" property. Note the name; you'll use it in your code to load the graphic so it can be displayed in your game. We'll name the ball as **ball**, right paddle as **paddleR**, left paddle as **paddleL**, and background as **bg**.

Loading and drawing the sprites:

1. Now, you must write code that loads and displays the sprite on the screen. Back in the Code view of your game, find the **LoadGraphicsContent** method, and add the following lines in and above the method so it looks similar to this:

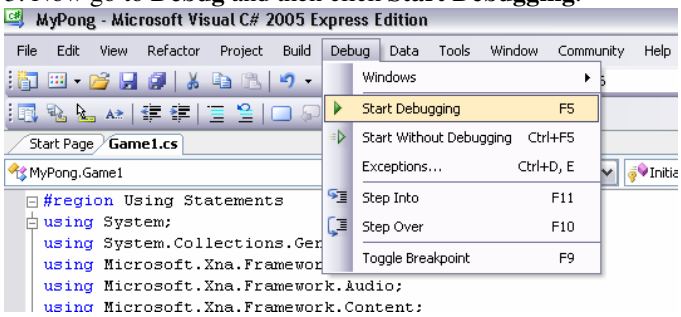
```
Texture2D myBall; //The 2D texture to be rendered
SpriteBatch spriteBatch; //The object to draw the sprites

protected override void LoadGraphicsContent(bool loadAllContent)
{
   if (loadAllContent)
   {
       spriteBatch = new SpriteBatch(this.graphics.GraphicsDevice);

       // Load the images into a texture object
       ContentManager contentLoader = new ContentManager(this.Services);
       myBall = contentLoader.Load<Texture2D>("ball");
    }
}
```

2. Now, add code to the Draw loop so it looks like this:

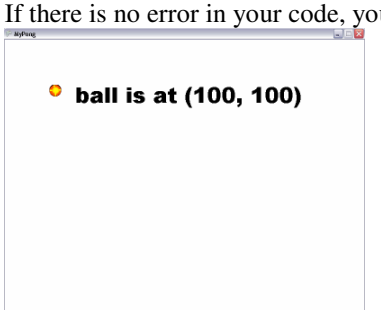
```
  protected override void Draw(GameTime gameTime)
  {
      graphics.GraphicsDevice.Clear(Color.White);
      spriteBatch.Begin(SpriteBlendMode.AlphaBlend);// transparency
      spriteBatch.Draw(myBall, new Rectangle(100, 100, myBall.Width, myBall.Height),
Color.White); //(Texture2D, destination rectangle (x, y, width, height), Color tint)

      spriteBatch.End();
      base.Draw(gameTime);
 }
```

This code draws the sprite on the screen each frame.

3. Now go to **Debug** and then click **Start Debugging**.



If there is no error in your code, you should be able to see the ball on the screen at position (100, 100):



4. Now you'll need to add two paddles and the background. You'll also need to position them at appropriate places. Your code should look similar to this:

```
Texture2D myBall, myPaddleL, myPaddleR, myBg;
SpriteBatch spriteBatch;
int paddleL_X, paddleR_X;
int paddleL_Y, paddleR_Y;
int ballX, ballY;
int viewportHeight, viewportWidth;

protected override void LoadGraphicsContent(bool loadAllContent)
{
   if (loadAllContent)
   {
       spriteBatch = new SpriteBatch(this.graphics.GraphicsDevice);
       ContentManager contentLoader = new ContentManager(this.Services);
       myBall = contentLoader.Load<Texture2D>("ball");
       myPaddleL = contentLoader.Load<Texture2D>("paddleL");
       myPaddleR = contentLoader.Load<Texture2D>("paddleR");
       myBg = contentLoader.Load<Texture2D>("bg");
    }
    // screen width and height
    viewportHeight = this.graphics.GraphicsDevice.Viewport.Height;
    viewportWidth = this.graphics.GraphicsDevice.Viewport.Width;

    // Center the paddles on the screen
    paddleL_Y = (viewportHeight / 2) - (myPaddleL.Height / 2);
    paddleR_Y = (viewportHeight / 2) - (myPaddleL.Height / 2);

    paddleL_X = 100;
    paddleR_X = viewportWidth - 100;

    // Center the ball on the screen
```

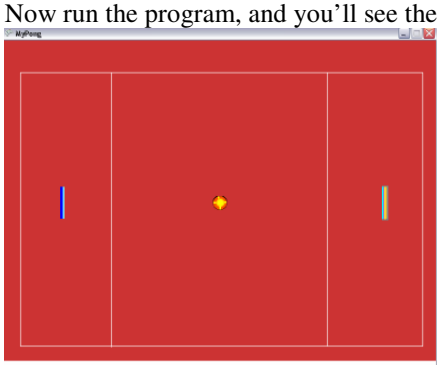
```
    ballX = (viewportWidth / 2) - (myBall.Width / 2);
    ballY = (viewportHeight / 2) - (myBall.Height / 2);
 }
```

Make sure the call to **ContentManager.Load** is using the "Asset Name" you saw in the Properties window in the previous step. This code will load and prepare your graphic to be drawn, and will reload your graphic if the graphics device is reset (such as in the case of the game window being resized).

6. Now, add following code to the Draw method so it looks similar to this:

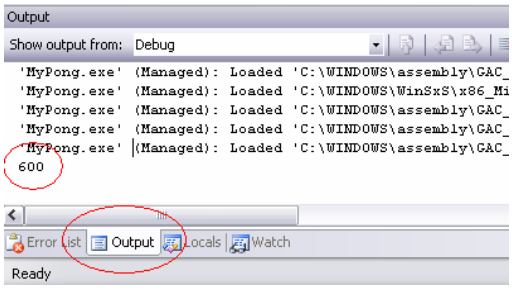
```
  protected override void Draw(GameTime gameTime)
  {
      graphics.GraphicsDevice.Clear(Color.White);
      spriteBatch.Begin(SpriteBlendMode.AlphaBlend);

      spriteBatch.Draw(myBg, new Rectangle(0, 0, viewportWidth, viewportHeight),
Color.White);
      spriteBatch.Draw(myBall, new Rectangle(ballX, ballY, myBall.Width,
myBall.Height), Color.White);

      spriteBatch.Draw(myPaddleL, new Rectangle(paddleL_X, paddleL_Y, myPaddleL.Width,
myPaddleL.Height), Color.White);
      spriteBatch.Draw(myPaddleR, new Rectangle(paddleR_X, paddleR_Y, myPaddleR.Width,
myPaddleR.Height), Color.White);

      spriteBatch.End();
      base.Draw(gameTime);
 }
```

Now run the program, and you'll see the following:



Note: **System.Console.WriteLine** function is useful for debugging. Let's output viewportHeight in the Output tub by running the program after inserting this code at the end of LoadGraphicsContent method:

```
System.Console.WriteLine(viewportHeight);
```



Moving the ball

1. First add new variables above the LoadGraphicsContent method:

```
int ballSpeedX = 3, ballSpeedY = 5;
```

2. Write a function called **moveBall** right below (but not inside) the **Update** loop:

```
protected void moveBall()
{
      ballX += ballSpeedX;
      ballY += ballSpeedY;
}
```

3. Now add a line of code to the **Update** loop so it looks similar to this:

```
protected override void Update(GameTime gameTime)
{
      if (GamePad.GetState(PlayerIndex.One).Buttons.Back == ButtonState.Pressed)
       this.Exit();


      moveBall();
      base.Update(gameTime);
}
```

4. Run the program to see if the ball actually moves.

Making the paddles move

1. Add new variables above **LoadGraphicsContent** method:

```
int paddleL_speedY = 10, paddleR_speedY = 10;
```

2. Write the following function **checkKeyboardInput** to move the right paddle.

```
protected void checkKeyboardInput()
{
       //Get the current state of the keyboard
       KeyboardState aKeyboard = Keyboard.GetState();

       //Get the current keys being pressed
       Keys[] aCurrentKeys = aKeyboard.GetPressedKeys();

       //Cycle through all of the keys being pressed and move the sprite accordingly
       for (int aCurrentKey = 0; aCurrentKey < aCurrentKeys.Length; aCurrentKey++)
       {
             // Change the direction of the paddleR and also prepare to move
             if (aCurrentKeys[aCurrentKey] == Keys.Up && paddleR_Y > 0)
             {
                   if (paddleR_speedY > 0)
                   {
                       paddleR_speedY = -paddleR_speedY;
                   }
                   paddleR_Y += paddleR_speedY;
             }
             if (aCurrentKeys[aCurrentKey] == Keys.Down && paddleR_Y < viewportHeight
- myPaddleR.Height)
             {
```

```
                    if (paddleR_speedY < 0)
                    {
                        paddleR_speedY = - paddleR_speedY;
                    }
                    paddleR_Y += paddleR_speedY;
                }
            }
}
```

3. Add a line of code to the Update loop (right after moveBall();):

```
checkKeyboardInput();
```

4. Follow similar steps to move the left paddle (you don't have to make a new method. You may just add lines of code inside the for loop).

Detecting Collisions

1. To make the ball bounce against the top and bottom boundaries, add the following function called **checkCollisions**.

```
protected void checkCollisions()
{
            if (ballY <= 0 || ballY >= viewportHeight - myBall.Height)
            {
                ballSpeedY = -ballSpeedY;
            }
}
```

2. Add a line of code to the **Update** loop (right after moveBall();):

```
checkCollisions();
```

3. Next, you need to check if the ball collided with the paddles. One approach is to use BoundingBoxes, which are used for collision detection in 3D. You can just zero out the Z value for this 2D tutorial. Add the following to checkCollisions function:

```
        //define the area by providing min(x, y, z) and max(x, y, z)
        BoundingBox bbball = new BoundingBox(new Vector3(ballX, ballY, 0), new
Vector3(ballX + myBall.Width, ballY + myBall.Height, 0));
        BoundingBox bbpaddleL = new BoundingBox(new Vector3(paddleL_X, paddleL_Y, 0),
new Vector3(paddleL_X + myPaddleL.Width, paddleL_Y + myPaddleL.Height, 0));
        BoundingBox bbpaddleR = new BoundingBox(new Vector3(paddleR_X, paddleR_Y, 0),
new Vector3(paddleR_X + myPaddleR.Width, paddleR_Y + myPaddleR.Height, 0));

        // ball collides with paddle
        if (bbball.Intersects(bbpaddleL) || bbball.Intersects(bbpaddleR))
        {
                    ballSpeedX = -ballSpeedX;
        }
```

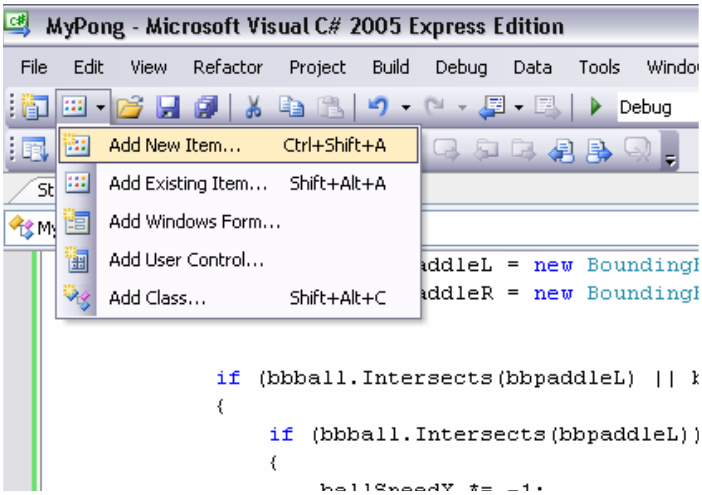
4. You also want to reset the ball's initial position when it collides with a boundary on the left or right part of the screen. Add the following checking to **checkCollisions** function:

```
            if (ballX <= 0 || ballX >= viewportWidth)
            {
                   ballX = viewportWidth / 2 + myBall.Width / 2;
```

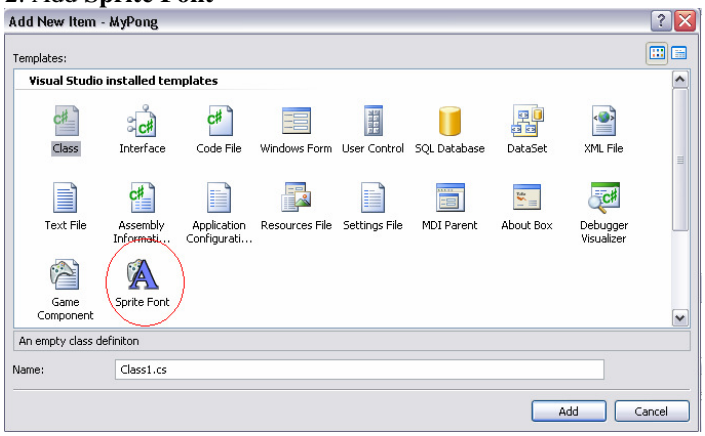
```
                ballY = viewportHeight / 2 - myBall.Height / 2;
        }
```

Counting Scores

1. Click **Add New Item** icon and select **Add New Item…**



2. Add **Sprite Font**



3. The file automatically opens up in a new tab. Change the **FontName** to **Arial**

```
<XnaContent xmlns:Graphics="Microsoft.Xna.Framework.Content.Pipeline.Graphics">
  <Asset Type="Graphics:FontDescription">

    <!--
    Modify this string to change the font that will be imported.
    -->
    <FontName>Arial</FontName>

    <!--
    Size is a float value, measured in points. Modify this value to change
    the size of the font.
    -->
    <Size>14</Size>
```

4. Now go back to your program **Game1.cs** and add new variables above the **LoadGraphicsContent** method:

```
SpriteFont myScoreFont;
int rightScore = 0, leftScore = 0;
```

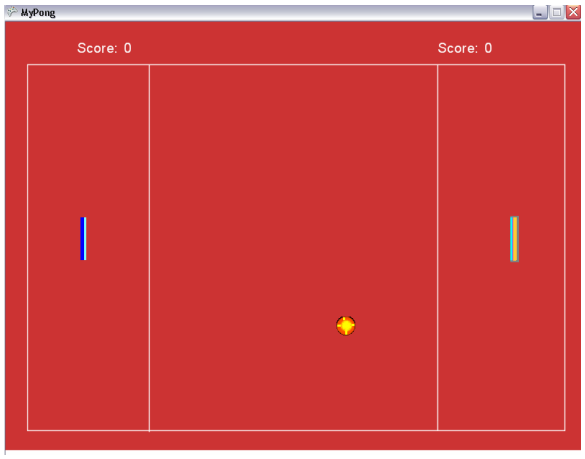
5. Add a line of code to the **LoadGraphicsContent** method (say, after myBg = contentLoader.Load<Texture2D>("bg");)

```
    myScoreFont = contentLoader.Load<SpriteFont>("spriteFont1");
```

6. Add the following code to the **Draw** loop to show the scores on the screen (before spriteBatch.End();):

```
      spriteBatch.DrawString(myScoreFont, "Score: " + rightScore, new
Vector2(paddleR_X-100, 25), Color.White); // (SpriteFont, string, (x,y), Color)

      spriteBatch.DrawString(myScoreFont, "Score: " + leftScore, new Vector2(paddleL_X,
25), Color.White);
```

7. Run your program to see if the scores (both 0) are displayed.



8. Add code to **checkCollisions** method so that the right player gets one point when the ball collides with a boundary on the left part of the screen and vice versa:

```
        if (ballX <= 0 || ballX >= viewportWidth)
        {
               if (ballX <= 0) // left boundary
               {
                   rightScore++;
               }
               else // right boundary
               {
                   leftScore++;
               }
               ballX = viewportWidth / 2 + myBall.Width / 2;
               ballY = viewportHeight / 2 - myBall.Height / 2;
         }
```

9. Now, you should be able to add more feartures on your own, such as determining a winner and displaying messages. It may be fun to write code to produce AI for the right player; the right paddle moves towards the ball by a restricted amount. To exit the game, use `this.Exit();`

Game1.cs Source Code
(Note: All variables are declared before the constructor)

```
#region Using Statements
using System;
using System.Collections.Generic;
using Microsoft.Xna.Framework;
using Microsoft.Xna.Framework.Audio;
using Microsoft.Xna.Framework.Content;
using Microsoft.Xna.Framework.Graphics;
using Microsoft.Xna.Framework.Input;
using Microsoft.Xna.Framework.Storage;
#endregion

namespace MyPong
{
    /// <summary>
    /// This is the main type for your game
    /// </summary>
    public class Game1 : Microsoft.Xna.Framework.Game
    {
        GraphicsDeviceManager graphics;
        ContentManager content;
        Texture2D myBall, myPaddleL, myPaddleR, myBg; //The 2D texture to be rendered
        SpriteBatch spriteBatch; //The object to draw the sprites
        int paddleL_X, paddleR_X;
        int paddleL_Y, paddleR_Y;
        int ballX, ballY;
        int viewportHeight, viewportWidth;
        int ballSpeedX = 3, ballSpeedY = 5;
        int paddleL_speedY = 10, paddleR_speedY = 10;
        SpriteFont myScoreFont;
        int rightScore = 0, leftScore = 0;

        public Game1()
        {
            graphics = new GraphicsDeviceManager(this);
            content = new ContentManager(Services);
        }

        protected override void Initialize()
        {
            // TODO: Add your initialization logic here

            base.Initialize();
        }

        protected override void LoadGraphicsContent(bool loadAllContent)
        {
            if (loadAllContent)
            {
                spriteBatch = new SpriteBatch(this.graphics.GraphicsDevice);
                // Load the images into a texture object
                ContentManager contentLoader = new ContentManager(this.Services);
                myBall = contentLoader.Load<Texture2D>("ball");
                myPaddleL = contentLoader.Load<Texture2D>("paddleL");
                myPaddleR = contentLoader.Load<Texture2D>("paddleR");
                myBg = contentLoader.Load<Texture2D>("bg");
                myScoreFont = contentLoader.Load<SpriteFont>("spriteFont1");
            }
            // screen width and height
            viewportHeight = this.graphics.GraphicsDevice.Viewport.Height;
            viewportWidth = this.graphics.GraphicsDevice.Viewport.Width;

            // Center the paddles on the screen
            paddleL_Y = (viewportHeight / 2) - (myPaddleL.Height / 2);
            paddleR_Y = (viewportHeight / 2) - (myPaddleL.Height / 2);
            paddleL_X = 100;
            paddleR_X = viewportWidth - 100;
```

```
        // Center the ball on the screen
        ballX = (viewportWidth / 2) - (myBall.Width / 2);
        ballY = (viewportHeight / 2) - (myBall.Height / 2);
    }

    protected override void UnloadGraphicsContent(bool unloadAllContent)
    {
        if (unloadAllContent)
        {
            // TODO: Unload any ResourceManagementMode.Automatic content
            content.Unload();
        }

        // TODO: Unload any ResourceManagementMode.Manual content
    }

    protected override void Update(GameTime gameTime)
    {
        // Allows the game to exit
        if (GamePad.GetState(PlayerIndex.One).Buttons.Back == ButtonState.Pressed)
            this.Exit();

        moveBall();
        checkKeyboardInput();
        checkCollisions();
        base.Update(gameTime);
    }

    protected void moveBall()
    {
        ballX += ballSpeedX;
        ballY += ballSpeedY;
    }

    protected void checkKeyboardInput()
    {
        //Get the current state of the keyboard
        KeyboardState aKeyboard = Keyboard.GetState();
        //Get the current keys being pressed
        Keys[] aCurrentKeys = aKeyboard.GetPressedKeys();
        //Cycle through all of the keys being pressed and move the sprite accordingly
        for (int aCurrentKey = 0; aCurrentKey < aCurrentKeys.Length; aCurrentKey++)
        {
            // Change the direction of the paddleR and also prepare to move
            if (aCurrentKeys[aCurrentKey] == Keys.Up && paddleR_Y > 0)
            {
                if (paddleR_speedY > 0)
                {
                    paddleR_speedY = -paddleR_speedY;
                }
                paddleR_Y += paddleR_speedY;
             }
            if (aCurrentKeys[aCurrentKey] == Keys.Down && paddleR_Y < viewportHeight - myPaddleR.Height)
            {
                if (paddleR_speedY < 0)
                {
                    paddleR_speedY = -paddleR_speedY;
                }
                paddleR_Y += paddleR_speedY;
            }
            if (aCurrentKeys[aCurrentKey] == Keys.LeftShift && paddleL_Y > 0)
            {
                if (paddleL_speedY > 0)
                {
                    paddleL_speedY = -paddleL_speedY;
                }
                paddleL_Y += paddleL_speedY;
            }
            if (aCurrentKeys[aCurrentKey] == Keys.LeftControl && paddleL_Y < viewportHeight - myPaddleL.Height)
```

```
            {
                if (paddleL_speedY < 0)
                {
                    paddleL_speedY = -paddleL_speedY;
                }
                paddleL_Y += paddleL_speedY;
            }
        }
    }
    protected void checkCollisions()
    {
        if (ballY <= 0 || ballY >= viewportHeight - myBall.Height)
        {
            ballSpeedY = -ballSpeedY;
        }
        //define the area by providing min(x, y, z) and max(x, y, z)
        BoundingBox bbball = new BoundingBox(new Vector3(ballX, ballY, 0), new
        Vector3(ballX + myBall.Width, ballY + myBall.Height, 0));
        BoundingBox bbpaddleL = new BoundingBox(new Vector3(paddleL_X, paddleL_Y, 0),
        new Vector3(paddleL_X + myPaddleL.Width, paddleL_Y + myPaddleL.Height, 0));
        BoundingBox bbpaddleR = new BoundingBox(new Vector3(paddleR_X, paddleR_Y, 0),
        new Vector3(paddleR_X + myPaddleR.Width, paddleR_Y + myPaddleR.Height, 0));
        // ball collides with paddle
        if (bbball.Intersects(bbpaddleL) || bbball.Intersects(bbpaddleR))
        {
            ballSpeedX = -ballSpeedX;
        }
        if (ballX <= 0 || ballX >= viewportWidth)
        {
            if (ballX <= 0) // left boundary
            {
                rightScore++;
            }
            else // right boundary
            {
                leftScore++;
            }
            ballX = viewportWidth / 2 + myBall.Width / 2;
            ballY = viewportHeight / 2 - myBall.Height / 2;
        }
    }

    protected override void Draw(GameTime gameTime)
    {
        graphics.GraphicsDevice.Clear(Color.White);
        spriteBatch.Begin(SpriteBlendMode.AlphaBlend);// transparency
        spriteBatch.Draw(myBg, new Rectangle(0, 0, viewportWidth, viewportHeight), Color.White);
        spriteBatch.Draw(myBall, new Rectangle(ballX, ballY, myBall.Width, myBall.Height),
        Color.White); //(Texture2D, destination rectangle (x, y, width, height), Color)
        spriteBatch.Draw(myPaddleL, new Rectangle(paddleL_X, paddleL_Y, myPaddleL.Width, myPaddleL.Height), Color.White);
        spriteBatch.Draw(myPaddleR, new Rectangle(paddleR_X, paddleR_Y, myPaddleR.Width, myPaddleR.Height), Color.White);
        spriteBatch.DrawString(myScoreFont, "Score: " + rightScore, new Vector2(paddleR_X - 100, 25), Color.White); // (SpriteFont, string,
(x,y), Color)
        spriteBatch.DrawString(myScoreFont, "Score: " + leftScore, new Vector2(paddleL_X, 25), Color.White);
        spriteBatch.End();
        base.Draw(gameTime);
    }
  }
}
```