MonoGame Blocks Tutorial Part 8

Contents

Part 8 – The Final Frontier: Drawing Text

Time Required: 30 minutes

Our game is somewhat playable now, but it doesn't show the Score, or the number of Balls remaining, and we don't haven't Start or Game Over messages displayed on the screen. In this final part, we'll add these finishing touches. As we mentioned earlier, DirectX (which MonoGame uses) doesn't directly have a mechanism for drawing text. Text is actually drawn on the screen like any other image. Bitmap fonts have an image for each character in the character set, and whenever we tell MonoGame to write text to the screen, it is actually drawing those images for us. That's why we had to add the `spriteFont` in Part 3. That caused the pipeline tool to generate the necessary images. The `SpriteBatch` class has a "DrawString" method that we'll use to draw text on the screen.

One piece of information that we need to display, is how many balls the user still has available to play. Games like this usually show an icon for the ball along with a number, so that's what we'll do here. In our "Game1.cs" file add the following line to add another ball object to our game:

```
// Create reference variables
private GraphicsDeviceManager _graphics;
private SpriteBatch _spriteBatch;
GameContent gameContent;

private Paddle paddle;
private Wall wall;
private GameBorder gameBorder;
private Ball ball;
// Used to draw image next to remaining ball count at top of screen
private Ball staticBall;
```

In the same file, in the `LoadContent` method, add the lines to the end of the method to create the instance of the ball we just added:

```
// Create static ball next to score
staticBall = new Ball(screenWidth, screenHeight, _spriteBatch, gameContent);
staticBall.BallX = 25;
staticBall.BallY = 25;
staticBall.IsBallVisible = true;
staticBall.UseRotation = false;
```

We set the `UseRotation` property to false because we don't want this ball spinning. Now let's add our text messages to the game. We'll need the following:

- Number of balls remaining
- Score
- Start Game Message
- Game Over Message

These will all be added to the `Draw` method in "Game1.cs". Add the indicated lines to the `Draw` method. We'll discuss what they do in a bit:

```
161         protected override void Draw(GameTime gameTime)
162         {
163             GraphicsDevice.Clear(Color.Black);
164 
165             // Begin drawing to buffer
166             _spriteBatch.Begin();
167 
168             // Call the game objects Draw methods
169             paddle.Draw();
170             wall.Draw();
171             gameBorder.Draw();
172 
173             if (ball.IsBallVisible)
174             {
175                 bool inPlay = ball.Move(wall, paddle);
176                 if (inPlay)
177                 {
178                     ball.Draw();
179                 }
180                 else
181                 {
182                     ballsRemaining--;
183                     readyToServeBall = true;
184                 }
185             }
186 
187             staticBall.Draw();
188 
189             string scoreMsg = "Score: " + ball.Score.ToString("00000");
190             Vector2 space = gameContent.labelFont.MeasureString(scoreMsg);
191             _spriteBatch.DrawString(gameContent.labelFont, scoreMsg,
192                                     new Vector2((screenWidth - space.X) / 2,
193                                     screenHeight - 40), Color.White);
194             if (ball.BlocksCleared >= 70)
195             {
196                 ball.IsBallVisible = false;
197                 ball.BlocksCleared = 0;
198                 wall = new Wall(1, 50, _spriteBatch, gameContent);
199                 readyToServeBall = true;
200             }
201             if (readyToServeBall)
202             {
203                 if (ballsRemaining > 0)
204                 {
205                     string startMsg = "Press <Space> or Click Mouse to Start";
206                     Vector2 startSpace = gameContent.labelFont.MeasureString(startMsg);
207                     _spriteBatch.DrawString(gameContent.labelFont, startMsg,
208                         new Vector2((screenWidth - startSpace.X) / 2,
209                         screenHeight / 2), Color.White);
210                 }
211                 else
212                 {
213                     string endMsg = "Game Over";
214                     Vector2 endSpace = gameContent.labelFont.MeasureString(endMsg);
215                     _spriteBatch.DrawString(gameContent.labelFont, endMsg,
216                         new Vector2((screenWidth - endSpace.X) / 2,
217                         screenHeight / 2), Color.White);
```

```
218                 }
219             }
220             _spriteBatch.DrawString(gameContent.labelFont, ballsRemaining.ToString(),
221                 new Vector2(40, 10), Color.White);
222
223             // Write buffer to screen
224             _spriteBatch.End();
225
226             base.Draw(gameTime);
227         }
228     }
229 }
```

The first line you added tells the new "Balls Remaining" `staticBall` to draw itself: `staticBall.Draw`. Next, we create the score message by taking a string literal "Score: ", and concatenating a leading-zero string with the game score from the `Ball` class. We want to center this on the screen, so we use the `SpriteFont MeasureString` method to determine how much space this string will take on the screen. We then call the `SpriteBatch DrawString` method. The first argument we pass is the font we want to use: `labelFont`. Next, we pass the string to be displayed, our score.

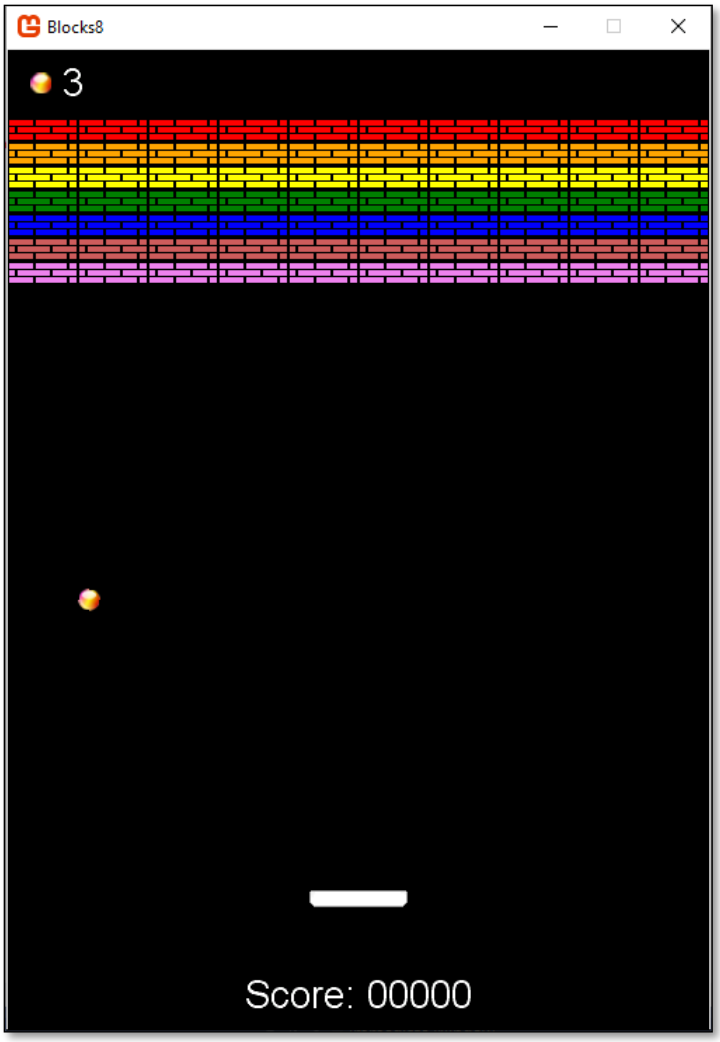
The next parameter is a vector containing the `X` and `Y` coordinates where the text is to be drawn. We compute the `X` coordinate using the width of the screen minus the length of the string displayed and dividing by two, to center it on the screen, and position it 40 pixels from the bottom of the screen. The final parameter is the color that should be used to draw text, in this case white.

We've also added some game logic to check if all 70 Blocks have been cleared. If so, we'll hide the ball, and set up for a new game level by creating a new wall, and setting a flag indicating that we are ready to serve a ball.

Next, if we are ready to serve a ball, and there is at least one game ball left to play, we display the game start message. If there are no balls left, we display the game over message.

Finally, we display the balls remaining count next to the ball icon at the top of the screen.

Phew, we're done! Press **F5** and you should see the messages and be able to play the game in all of its glory!



We could add more polish. We could display the game level, speed up the ball, or shrink the size of the paddle at higher levels, like the old arcade game does. That will be left to you.

We have covered most of the basics you will need to create your own 2D game. We learned how to add and build assets so that MonoGame can consume them. We learned how to load assets using the content manager. We learned how to draw images and play sounds. We learned how to draw lines and rectangles on the screen. We learned how to draw text on the screen and get mouse and keyboard inputs. We learned how to move and rotate images.

Good luck with your future gaming projects!

Assignment Submission

Zip up the project folder. Submit in Blackboard.