MonoGame Blocks Tutorial Part 7

Contents

Part 7 – Playing Sound Effects

Time required: 30 minutes

In the previous episode, we added the ball to the game, and made the game mostly functional. But, we are still missing sound effects.

Now we're ready to add sound effects to our game. We've already done the hard part. The sounds are already loaded in our content manager. We just need to insert code to play them when the appropriate events occur. We'll add a sound effect for each of these game events:

- User Launches a ball
- Ball hits a wall
- Ball hits a block
- Ball hits the paddle
- Ball falls out of play

You'll notice that all of these occur based on some ball event, so the logical place to add our sounds will be in the *Ball* class. First, let's add a method which will make the actual call for us to MonoGame to play the sound. In the "Ball.cs" file add the following `PlaySound` method:

```
public static void PlaySound(SoundEffect sound)
{
    float volume = 1;
    float pitch = 0.0f;
    float pan = 0.0f;
    sound.Play(volume, pitch, pan);
}
```

There's not much to it. The `volume` argument determines the volume to play the sound effect from 0 (silent) to 1 (full volume). The `pitch` argument allows you to change the pitch of the sound effect (you can raise or lower the octave by adding or subtracting -1). You can also provide fractional values for more refined adjustments. The `pan` argument is like the balance control on your stereo. It controls how much sound comes out of the left and right speaker. A value of 0 means balanced between left and right speakers. A value of "-1" means left speaker only. A value of "1" means right speaker only. Any other values between these can provide a mix between the two speakers. The `sound.Play` method call tells MonoGame to play the sound. Now we add a call to this method whenever a sound-worthy event happens.

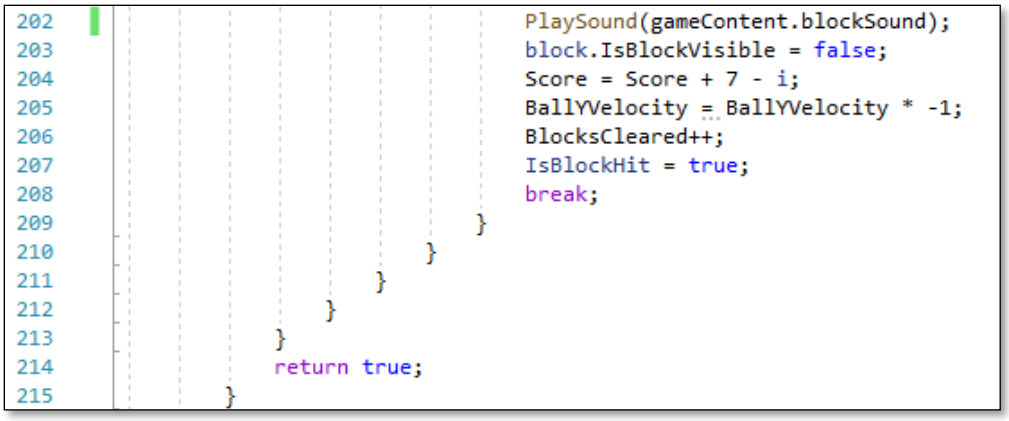
All of our sounds are caused by the ball. We'll add all of our sound effects to the "Ball.cs" file. In the "Launch" method, add the indicated line:

```
public void Launch(float x, float y, float xVelocity, float yVelocity)
{
    if (IsBallVisible == true)
    {
        return;  // Ball already exists, ignore
    }
    PlaySound(gameContent.startSound);
    IsBallVisible = true;
    BallX = x;
    BallY = y;
    BallXVelocity = xVelocity;
    BallYVelocity = yVelocity;
}
```

That will play a sound effect when we launch a new ball. The remaining sounds will occur when the ball interacts with the play field. Add the indicated lines to the "Ball.cs" file in the `Move` method as indicated:

```
92          public bool Move(Wall wall, Paddle paddle)
93          {
94              if (IsBallVisible == false)
95              {
96                  return false;
97              }
98              BallX = BallX + BallXVelocity;
99              BallY = BallY + BallYVelocity;
100
101             // Check for wall hits
102             if (BallX < 1)
103             {
104                 BallX = 1;
105                 BallXVelocity = BallXVelocity * -1;
106                 PlaySound(gameContent.wallBounceSound);
107             }
108             if (BallX > ScreenWidth - BallWidth + 5)
109             {
110                 BallX = ScreenWidth - BallWidth + 5;
111                 BallXVelocity = BallXVelocity * -1;
112                 PlaySound(gameContent.wallBounceSound);
113             }
114             if (BallY < 1)
115             {
116                 BallY = 1;
117                 BallYVelocity = BallYVelocity * -1;
118                 PlaySound(gameContent.wallBounceSound);
119             }
120             if (BallY > ScreenHeight)
121             {
122                 IsBallVisible = false;
123                 BallY = 0;
124                 PlaySound(gameContent.missSound);
125                 return false;
126             }
127
128             // Check for paddle hit
129             // Paddle is 70 pixels.
130             // Logically divide it into segments that will determine the angle of the bounce
131             Rectangle paddleRect = new Rectangle((int)paddle.PaddleX, (int)paddle.PaddleY,
132                                                  (int)paddle.PaddleWidth, (int)paddle.PaddleHeight);
133             Rectangle ballRect = new Rectangle((int)BallX, (int)BallY,
134                                                (int)BallWidth, (int)BallHeight);
135             if (HitTest(paddleRect, ballRect))
136             {
137                 PlaySound(gameContent.paddleBounceSound);
138                 int offset = Convert.ToInt32((paddle.PaddleWidth - (paddle.PaddleX +
139                                               paddle.PaddleWidth - BallX + BallWidth / 2)));
140                 offset = offset / 5;
141                 if (offset < 0)
142                 {
143                     offset = 0;
144                 }
145                 switch (offset)
```

```
146                 {
147                     case 0:
148                         BallXVelocity = -6;
149                         break;
150                     case 1:
151                         BallXVelocity = -5;
152                         break;
153                     case 2:
154                         BallXVelocity = -4;
155                         break;
156                     case 3:
157                         BallXVelocity = -3;
158                         break;
159                     case 4:
160                         BallXVelocity = -2;
161                         break;
162                     case 5:
163                         BallXVelocity = -1;
164                         break;
165                     case 6:
166                         BallXVelocity = 1;
167                         break;
168                     case 7:
169                         BallXVelocity = 2;
170                         break;
171                     case 8:
172                         BallXVelocity = 3;
173                         break;
174                     case 9:
175                         BallXVelocity = 4;
176                         break;
177                     case 10:
178                         BallXVelocity = 5;
179                         break;
180                     default:
181                         BallXVelocity = 6;
182                         break;
183                 }
184                 BallYVelocity = BallYVelocity * -1;
185                 BallY = paddle.PaddleY - BallHeight + 1;
186                 return true;
187             }
188             bool IsBlockHit = false;
189             for (int i = 0; i < 7; i++)
190             {
191                 if (IsBlockHit == false)
192                 {
193                     for (int j = 0; j < 10; j++)
194                     {
195                         Block block = wall.BlockWall[i, j];
196                         if (block.IsBlockVisible)
197                         {
198                             Rectangle BlockRect = new Rectangle((int)block.BlockX, (int)block.BlockY,
199                                                                 (int)block.BlockWidth, (int)block.BlockHeight);
200                             if (HitTest(ballRect, BlockRect))
201                             {
```



```
202                                 PlaySound(gameContent.blockSound);
203                                 block.IsBlockVisible = false;
204                                 Score = Score + 7 - i;
205                                 BallYVelocity = BallYVelocity * -1;
206                                 BlocksCleared++;
207                                 IsBlockHit = true;
208                                 break;
209                             }
210                         }
211                     }
212                 }
213             }
214             return true;
215         }
```

It's time to run the game again to hear your new sounds effects in full glory! Press **F5** to run the game and confirm that you hear a sound whenever the ball interacts with something.

We're almost done! But as in life, a game must keep score. To do that, we'll need to tackle the final task in our tutorial—drawing text on the screen. We'll do that in the final part of this tutorial.

Assignment Submission

Zip up the pseudocode and the project folder. Submit in Blackboard.