Making Pong – Part 2

Building Mobile Apps

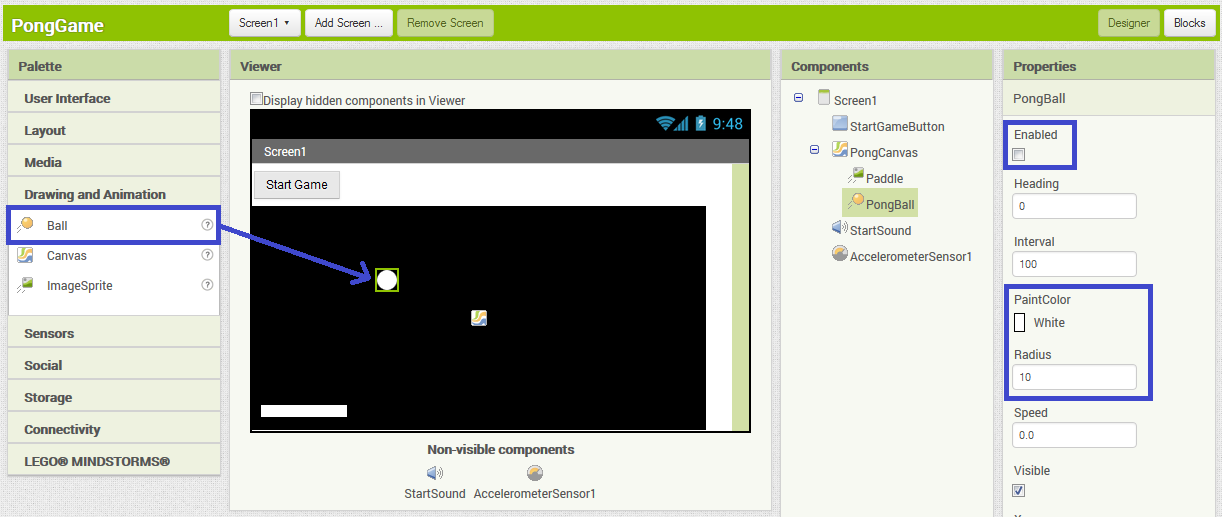
# Making Pong Tutorial - Part 2

Now you’ve got a paddle moving, it’s time to get a ball on the canvas and get it moving.

In the Designer view, in the Animation section of the Palette Window select the Ball component and drag it onto the canvas. Rename the Ball to PongBall, instead of Ball1 by using the Rename button.

The ball is black and has a radius of 5 by default, you can use the PaintColor selector and Radius textbox highlighted in blue below to adjust them to the colour and size you want. In this example, I have used a white ball with a radius of 10 to make it stand out a bit more, and as a tribute to Pong’s simple colour scheme.

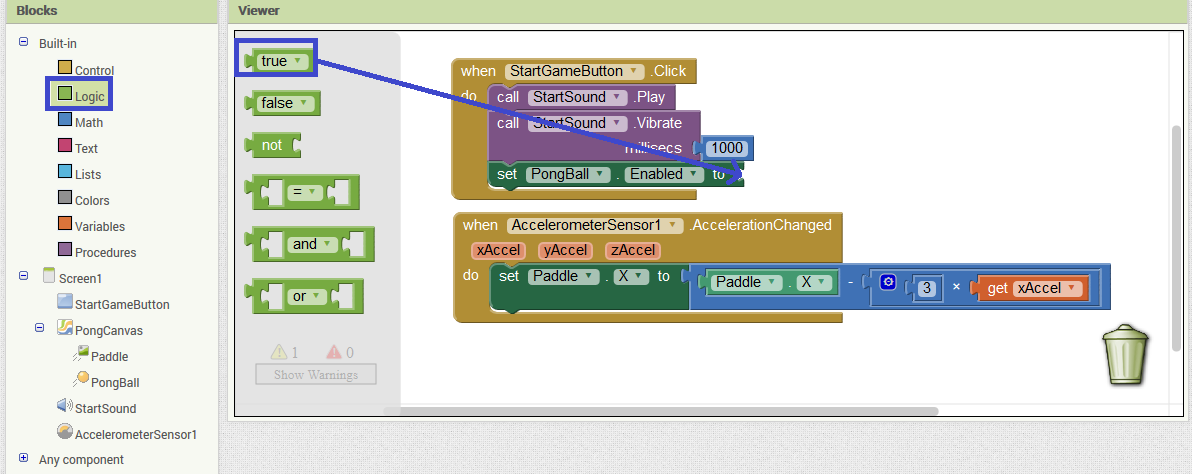
Un-tick the “Enabled” checkbox in the Ball’s Properties window as shown below.



We have all the pieces on the User Interface we need, now let’s code it in the Blocks view.

We are going to use the Start Game button to make a new game start every time it’s tapped. To get the ball moving, we have to enable it, when the Start Game button is tapped. To do this, click the PongBall window, and drag the set Enabled To block into the when StartGameButton Click block. It should click into place.

Next we need to select Logic in the Blocks window and drag the true block into the set Enabled To block as pictured below:

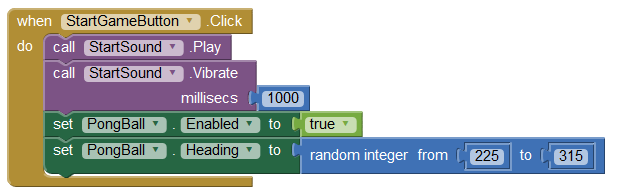


To get the ball moving we have to give it a direction to move in, a speed to move at and a time interval for its position to update.

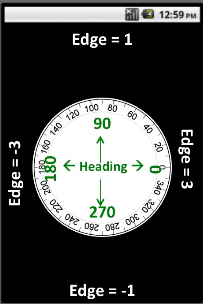
To start the game, we will make the ball move diagonally down to the right of the screen. To do this we select PongBall in the Blocks window and drag the set Ball.Heading to block onto the code area.

We don’t want the ball to always head in the same direction to start with, so next select Math in the Blocks window and drag the random integer from 1 to 100 block and click it into the set Ball.Heading to block. Change the first number from 1 to 225, and the second number to 315.

Once you have completed steps you should have something like the following:



What this means is that when the Start Game Button is clicked, the ball will be enabled and start moving towards the bottom-right. We have said that the Pong Ball will start heading to a random integer between 225 and 315. This is because App-Inventor uses the following way of determining where the ball should head:

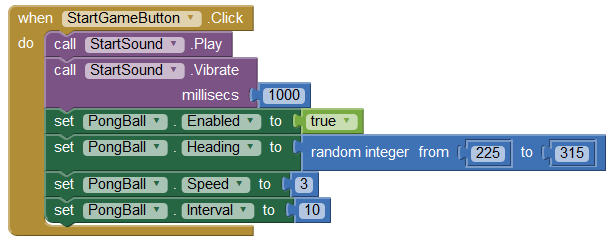


As you can see between 225 and 315, is heading vertically downwards.   
  
You may also notice the Edge values around the side. This means that when the ball collides with an edge, if the value of edge is 3 it has collided with the right side of the canvas, when it is -1 it has collided with the bottom of the canvas and so on. This will come in handy later, when we add the blocks for handling when the ball gets past the paddle and hits the bottom of the screen.

Now we will set the speed and the interval of the ball to control how fast the ball moves. To do this, select PongBall in the Blocks window and drag the set Ball.Speed to block across to the code area. Next create a number by selecting Math and dragging a number across to the code area. Change the 0 in the number block to any number, for this example I suggest 3, and snap it into the set Ball.Speed to block.

Now we will change the Ball’s interval. The interval of a sprite is how often (in milliseconds) that the position of the sprite updates on the canvas. To set the interval, select PongBall and drag the set Ball.Interval to block across to the code area. Now drag another number block onto the code area using the same method as for setting the Ball speed. Set this number to any number you like, although for this example I would suggest 10.

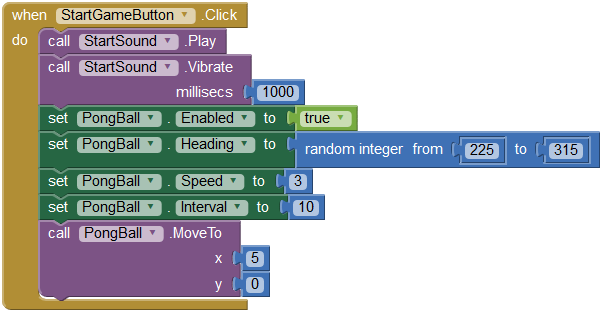
Once you have completed this step, you should have a block layout like this:



Now run the application, and tap your Start Game button. The ball will start moving towards the bottom of the screen. You will probably notice once the ball reaches the bottom of the canvas it rolls along the bottom edge of the canvas.

Next, we will make the ball sprite move to the top of the screen when the Start Game Button is touched, so that we can restart the game after the ball has stopped.

To do this, select PongBall in the Blocks window, and drag the Ball.MoveTo block across to the code area. Now drag 2 number blocks across to the code area, by selecting Math in the Blocks window. Change 1 of the number blocks to 5, and one to 0. Drag the 5 block into the X slot of the Ball.MoveTo block, and drag the 0 block into the Y slot of the Ball.MoveTo block. You should now have something like this:



Give the app a run, and the ball should move to the top left whenever the Start Game Button is tapped.

Our next step is to add blocks to handle the collision of the ball with the different sides of the canvas, and when the ball collides with the paddle.

We want the ball only to stop when it reaches the bottom edge, for all the other edges of the canvas we want it to bounce off. To do this, we can use the PongBall.EdgeReached event.

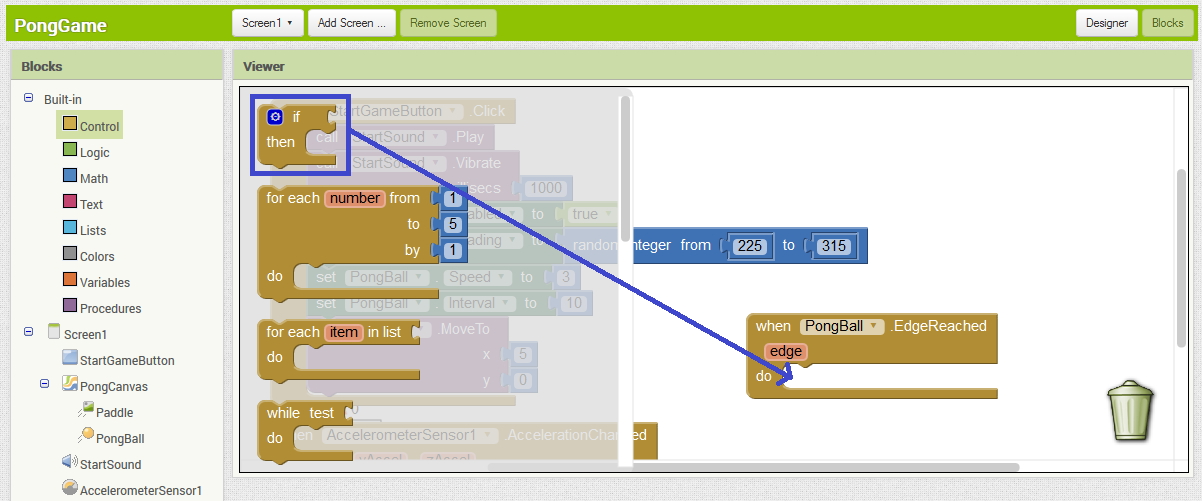
Select PongBall in the Blocks Window and drag across the PongBall.EdgeReached event across to the code area. Next, we are going to add some logic to this event to determine which edge the ball has collided with.

In AppInventor, in the EdgeReached event the edge variable is set to a number depending on the edge that the sprite has collided with, as shown in the diagram with edge numbers above.

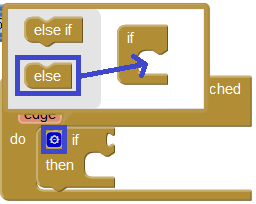
So, we need to check when the edge is set to -1, and stop the ball (by disabling it).

To do this, we are going to use an if-else block and check if the edge is -1.

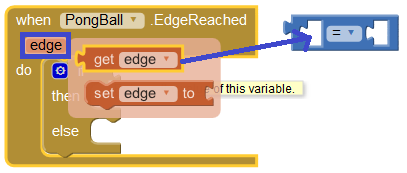
To add an if-then-else block, select the select Control in the Blocks window and drag and drop the if-else block into the Ball.EdgeReached block.



We are going to add an else piece to the if then block. This can be done by clicking the blue icon and dragging the else block into the if block as shown below:

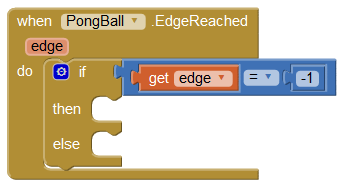


Next, we are going to add a test condition to the if-else block that checks whether the edge is -1. To do this, click on the Math section, and drag a = block onto the code area. Now select the orange edge, select the get edge block and drag it into the first spot of the = block.

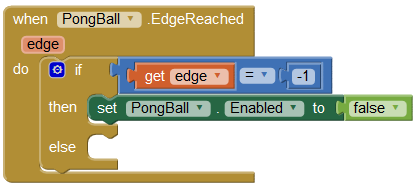


Next, go to Built-In, Math and drag a number block into the second space, changing the number to -1. Now drag the = block into the test space of the if-then-else block.

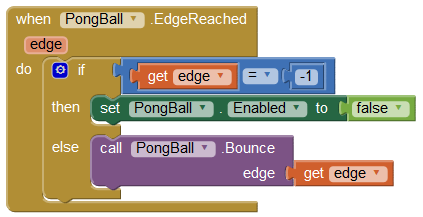
You should now have something like this:



Next, we need to add a statement to disable the ball (so that it stops moving) once the ball has reached the bottom edge. To do this, select PongBall in the Blocks window and drag the set Enabled to block over to the then space. Now click select Logic and drag the false block into the space in the set Enabled to block. You should now have something like this:

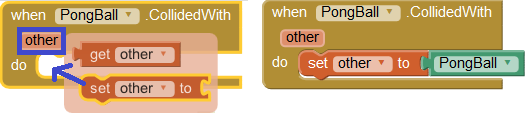


Now we need to make the ball bounce off the other walls, to do this select PongBall in the Blocks window and drag the Ball.Bounce block into the else block. Next click on the orange edge button and drag the get edge block into the edge spot in the Ball.Bounce block.  
  
You should now have something like this:

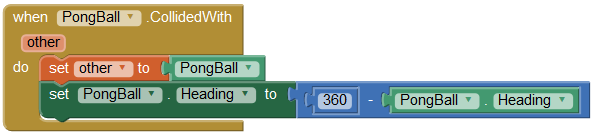


Tap Start Game a few times and notice that if the ball collides with the left side of the canvas, it bounces off. Now that the ball is bouncing off the edges, it’s time to make the ball bounce off the paddle.

To do this, select PongBall and drag the Ball.CollidedWith block onto the code area. Click the orange other button and drag the set other to block into the CollidedWith block. Select PongBall and drag and drop the last block into the set other to block as shown below.



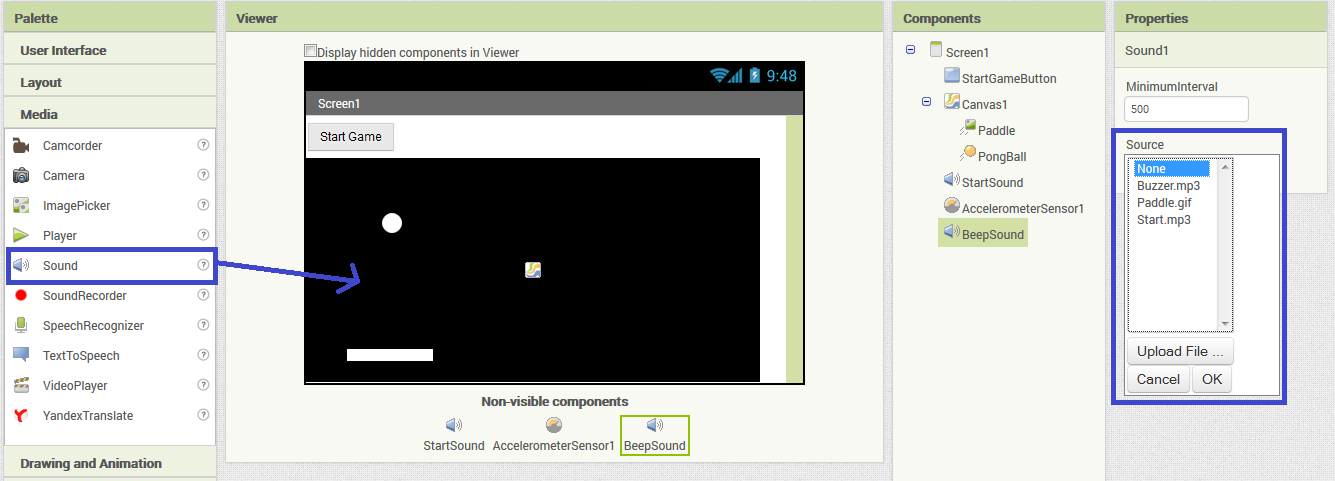
Select PongBall and drag the set Ball.Heading to block into the Ball.Collided block, and then select Math and drag a - (minus) block into the set Ball.Heading to block. In the first spot of the minus block, put the number 360, in the second, drag the Ball.Heading block into it. You should now have something that looks like the following:



Now run the app, and you should see ball bounce around the screen and bounce off the paddle.

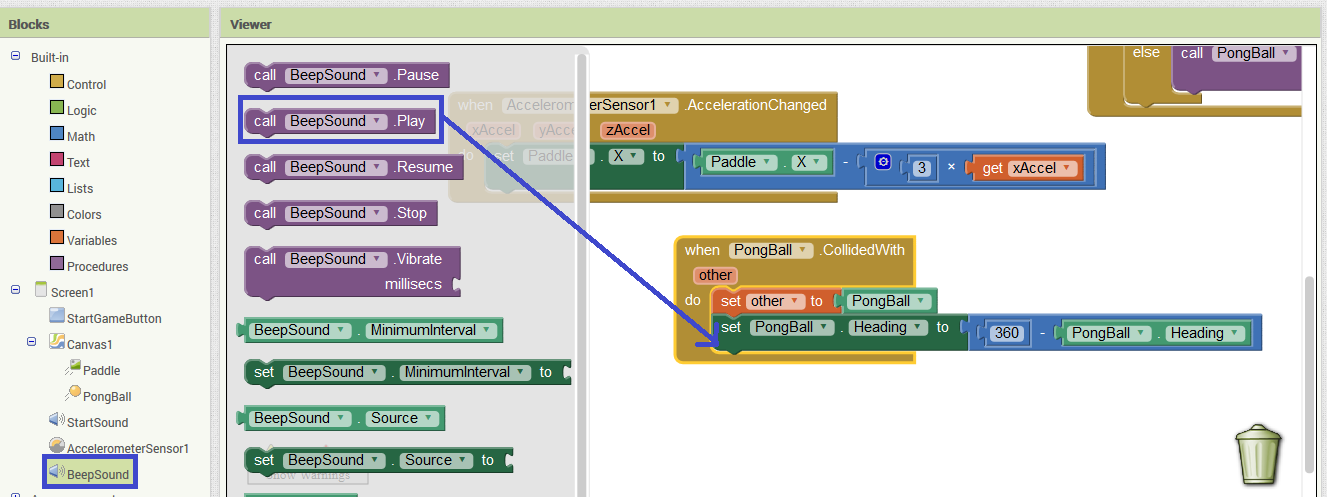
This app wouldn’t be a proper Pong game without a “beep” noise every time you hit the ball with the paddle. To add this Sound, go back to the Designer view by clicking Designer in the top-right corner.

Add a Sound by dragging and dropping a Sound component onto the canvas from the Media Components section, as pictured below:



Rename the Sound to something like “BeepSound”. Click the Source input in the Properties window for the Sound as shown above. Next, browse to the folder where you extracted the activity resources and open the file named “Beep.mp3”.  
  
Now we need to set up the block to play this beep sound whenever the paddle hits the ball. Go back to the Blocks view by clicking Blocks in the top-right corner.

Now, select BeepSound in the Blocks window and drag the BeepSound.Play block into the When PongBall.CollidedWith block as shown below:



Start a new game and you should now hear a beep every time the ball hits the paddle.

Congratulations! You now have a working Pong game!

# Some Extra Tweaks

If you finish early you might want to try and implement some more features. Some examples are:

* Adding a Score to keep track of how many times you’ve hit the wall
* Adding more sound effects, for example a buzzer noise when you lose a game
* Making it that the ball speeds up every time it hits the paddle to make it more difficult

If you need any help or extra ideas, please let me know.