Making Pong – Part 1

CS4S High School AppInventor Session

# Resources

The resources for this activity can be downloaded from the workshop website, from the “Pong Resources (zip)” link on the AppInventor session page.

Extract the zip folder (right click on the file, and select *Extract here*) to the *Downloads* folder or *My Documents* folder on the lab computer once it has finished downloading. Note that all of the work you do and files you upload today will be available on the AppInventor site, as your projects are stored “in the cloud”.

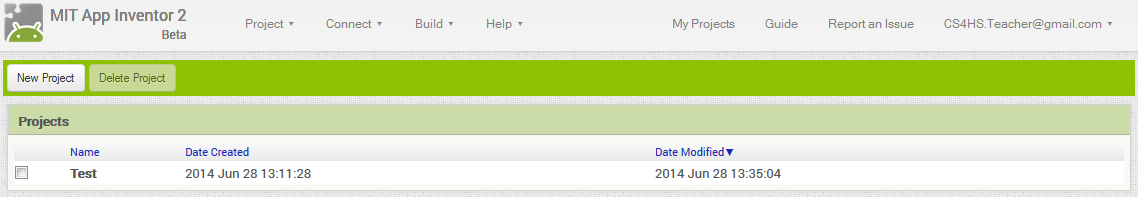
# Making Pong Tutorial - Part 1

Firstly, open up a web browser (today we’ll be using Mozilla Firefox).

Go to [http://ai2.appinventor.mit.edu](http://appinventor.mit.edu)

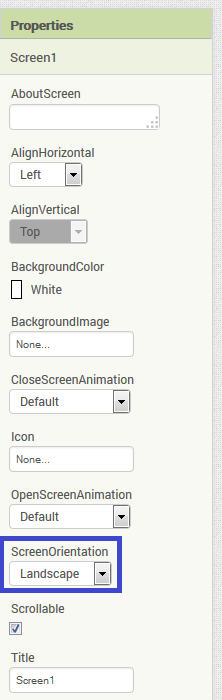
Log in using your Google account.

If you are not in the My Projects view (pictured below), then click Project > My Projects.



Next, click the New Project button and name the new project PongGame.

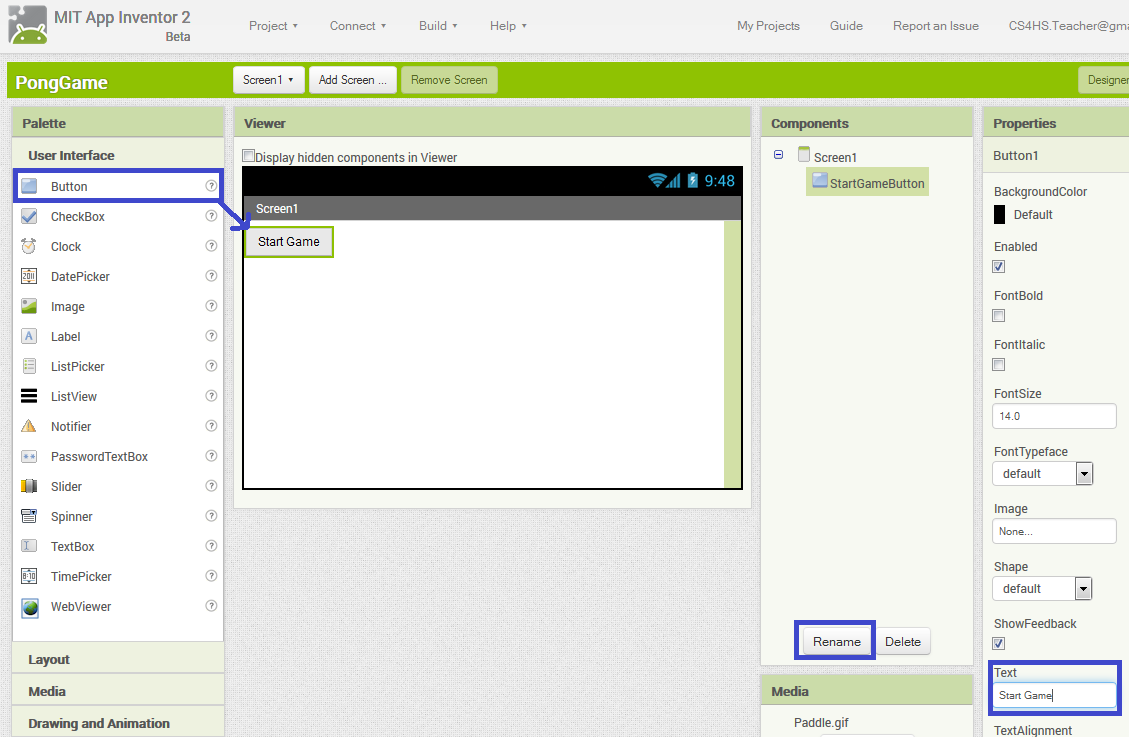
Once the new project has loaded, change the Screen Orientation to Landscape by clicking on the Screen1 component and selecting Landscape in the ScreenOrientation drop-down menu, this will allow us to play the game holding the tablet on its side.



Our first step is going to be add a “Start Game” button that will do the following when it is tapped:

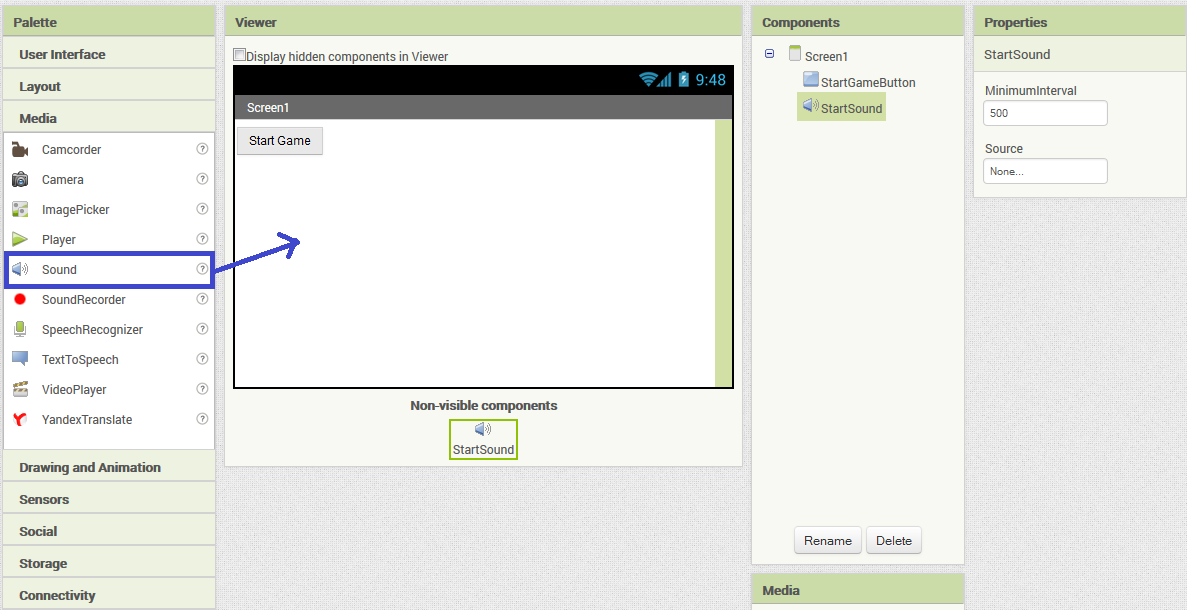
* Play a sound
* Make the tablet vibrate
* Make the Pong ball we add later start moving

Add a button to the layout in the Design view by dragging and dropping a Button from the User Interface menu in the Palette window across to the Screen as pictured below:



Rename the Button by using the Rename button highlighted above to something like “StartGameButton”. Change the text shown by the button to a message such as “Start Game” by using the Text input as highlighted in the image above.

Before being able to play a Sound, we’ll need to add a Sound to the layout in the Design view. To add a Sound drag and drop a Sound component from the Media sub-menu in the Palette window. Once added it will be displayed in the Non-visible components section of the Viewer window as shown below.

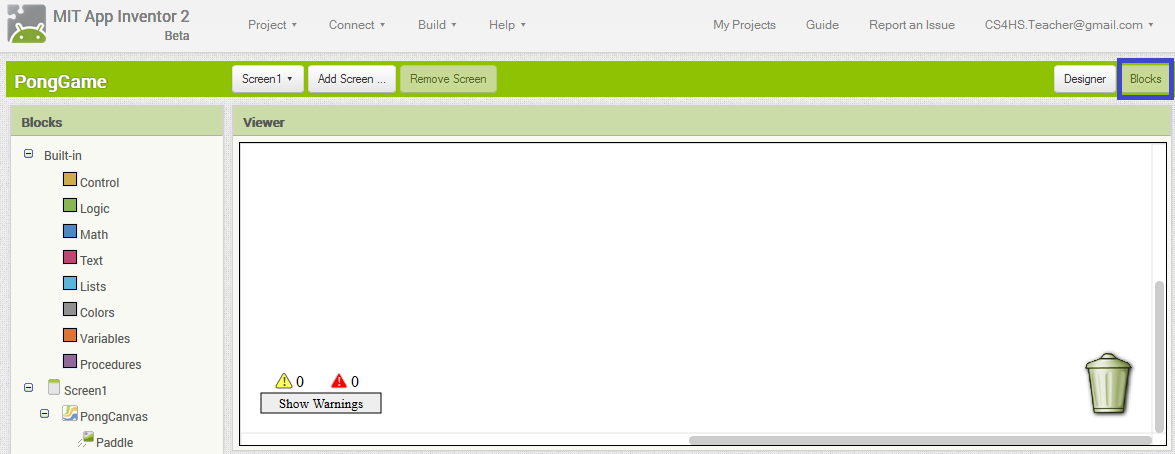


Rename the Sound1 component to something a bit more descriptive, such as “StartSound” by using the Rename button as you did for the Start Game Button.

Now we are going to add a media file for the sound to play. To do this, click on Source, Click Upload File… and navigate to the folder where you extracted the activity resources, and open the file named Start.mp3.

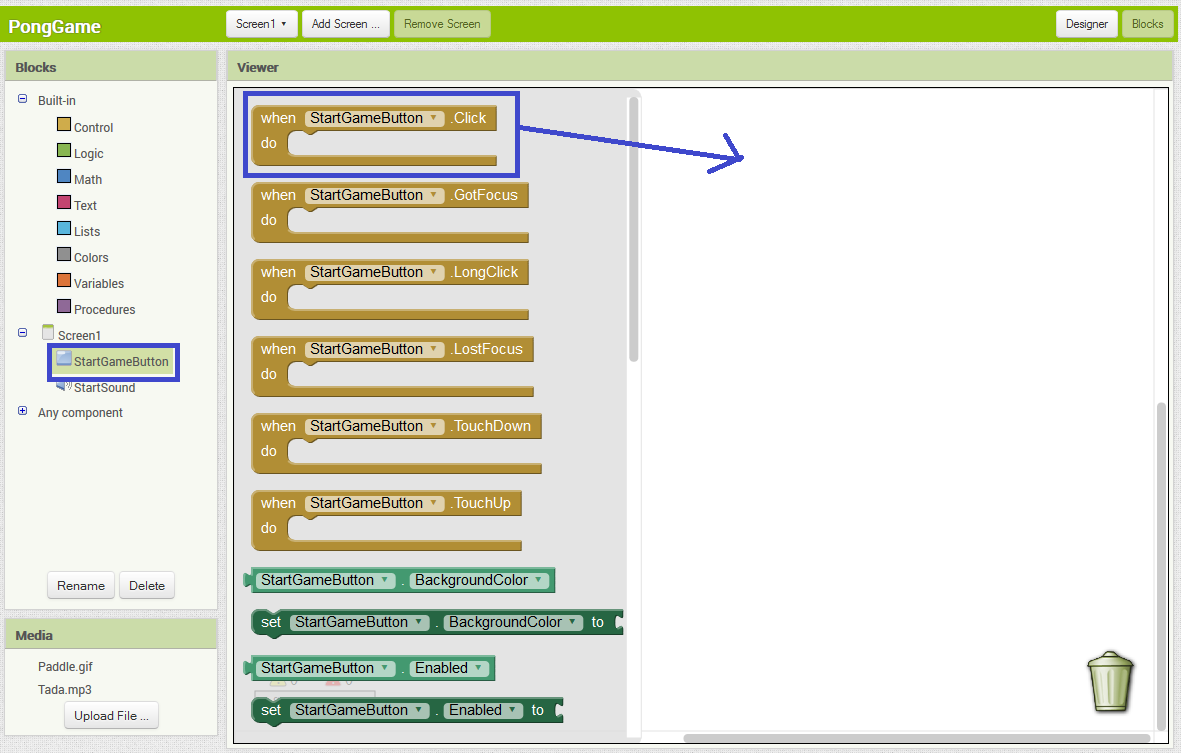
Now we are going to add some code (blocks) to play some sound and make the tablet vibrate.

Change to the Blocks Editor view by clicking “Blocks” as highlighted in blue below:



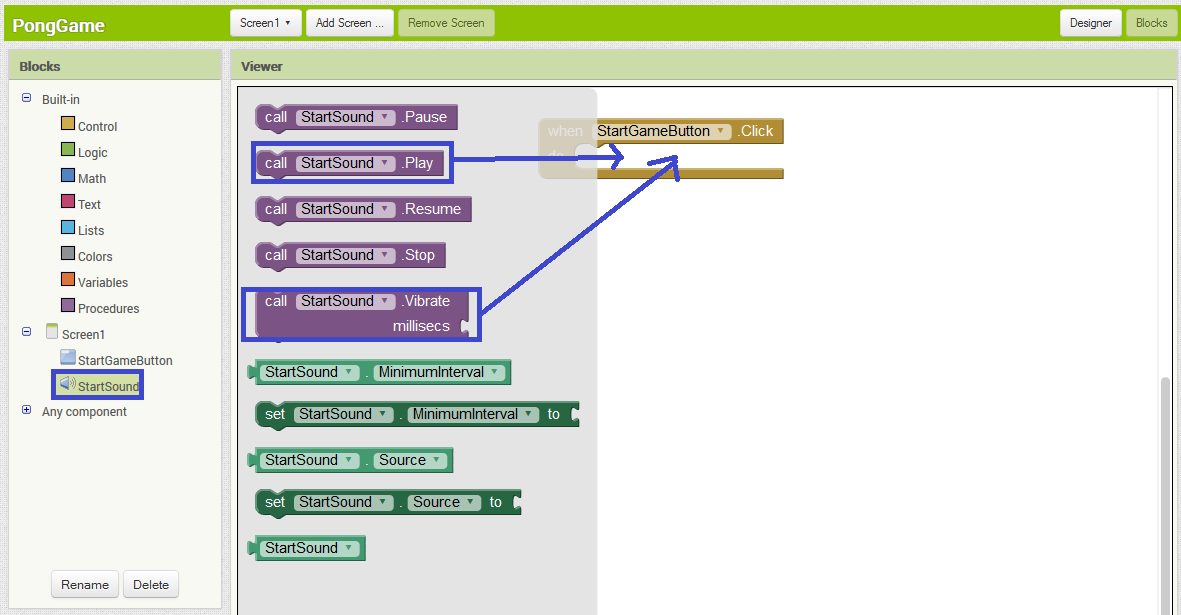
The Blocks view allows you to drag and drop blocks onto the viewer, these blocks offer events such as making code execute when a button is clicked, control statements such as while loops, and other common programming constructs like variables and procedures.

We want to put a block for when the Start Game Button is clicked, so click on the StartGameButton block in the Blocks window and drag the first block across to the right as pictured below:



Every block that is put inside this click block will run when the StartGameButton is ran. So we now need to put a block for vibrating the tablet, and a block for playing a sound inside of this click block.

To make the Start sound play on the click of the Start Game button, select Start Sound in the Blocks window and drag the *call StartSound.Play* block and the call *StartSound.Vibrate* block into the click block as pictured below:

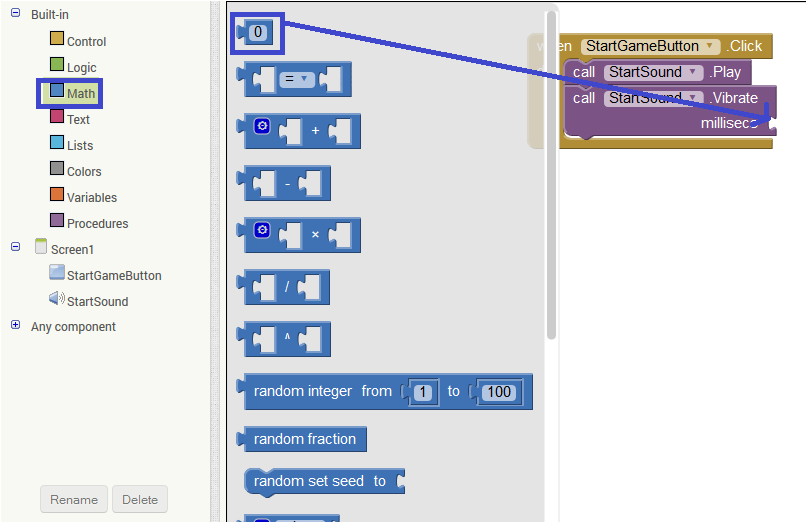


When you drag and drop the StartSound blocks into the Click block, they should click into place.

Tap the Start Game button and you should hear a “beep” noise. If it doesn’t make a noise when you tap it, make sure that the purple blocks are correctly in place in the Click block and that the tablet isn’t muted.

You will also notice that you get a message saying there is a “*Runtime Error: The Arguments are the wrong number of arguments for Vibrate”*. Have a think about what this could mean, and what is causing this error.

The error is caused because we haven’t specified the number of milliseconds the tablet should vibrate for. This is what the open part of the Vibrate block is for. We need to attach a block specifying the number of milliseconds it should vibrate. To do this, select Math in the Blocks window and drag the first block (the number block) into the millisecs slot of the Vibrate block as pictured below:

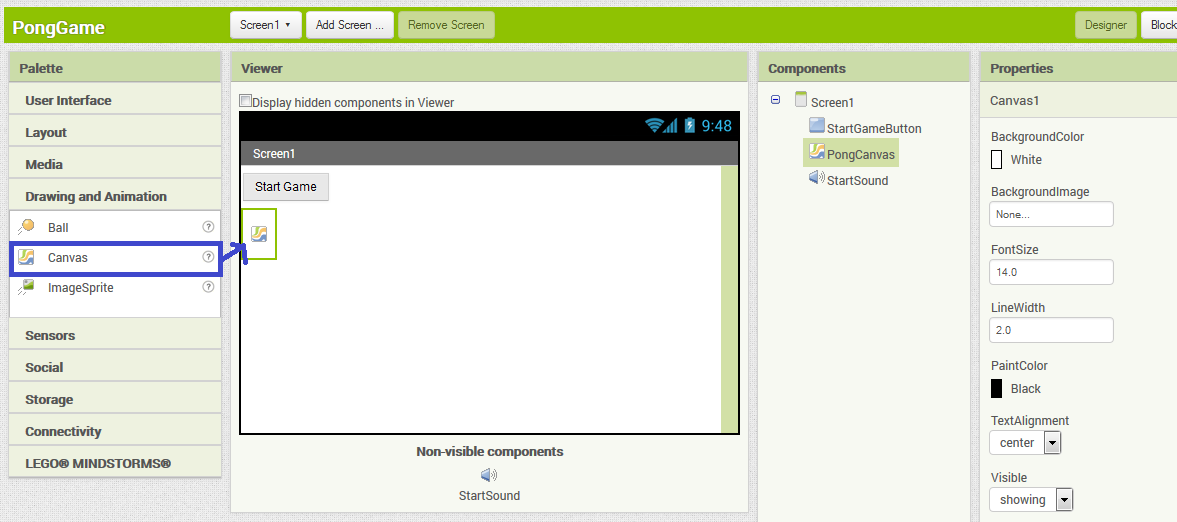


Once the number block has clicked into place, double click the 0 and change it to how long you think the tablet should vibrate for. Note that 1000 milliseconds = 1 second.

Tap the button now and the tablet should vibrate and play a sound.

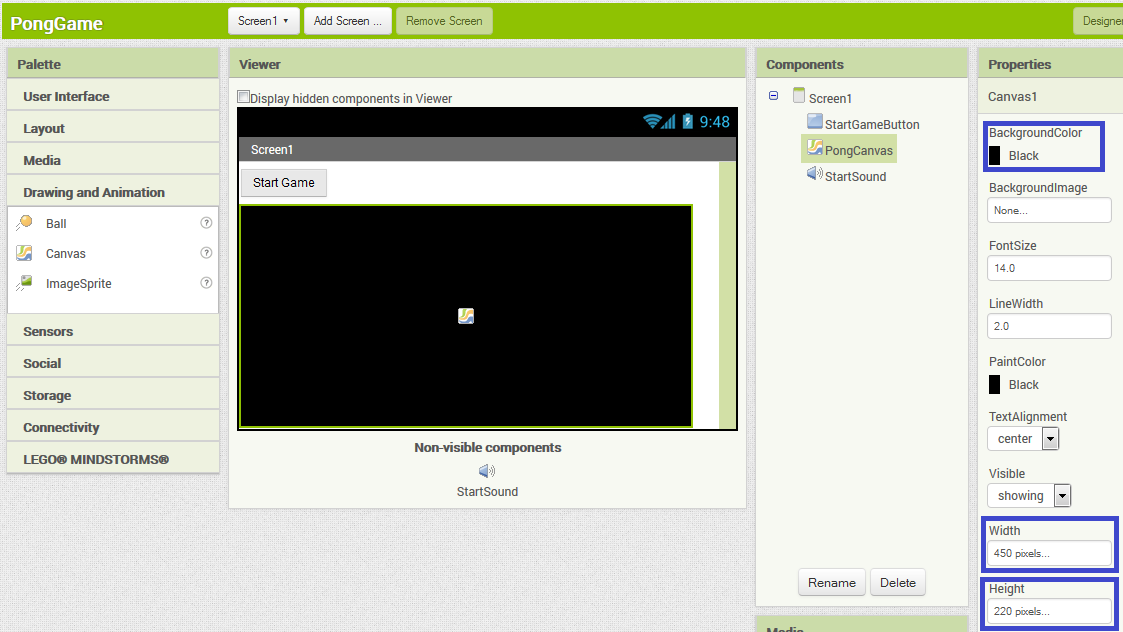
Now we are going to get started on making the main part of the app.

Switch back to the Designer view by clicking Designer in the top right corner. Select a Canvas component from the *Drawing and Animation* sub-menu in the Palette window and drag it onto the screen. The canvas component is a rectangular area which you can draw on, or put sprites (images) on and move them around. It is very useful for making animations and games.



Rename the canvas component by selecting it in the Components list, and clicking Rename. Rename it to “PongCanvas”.

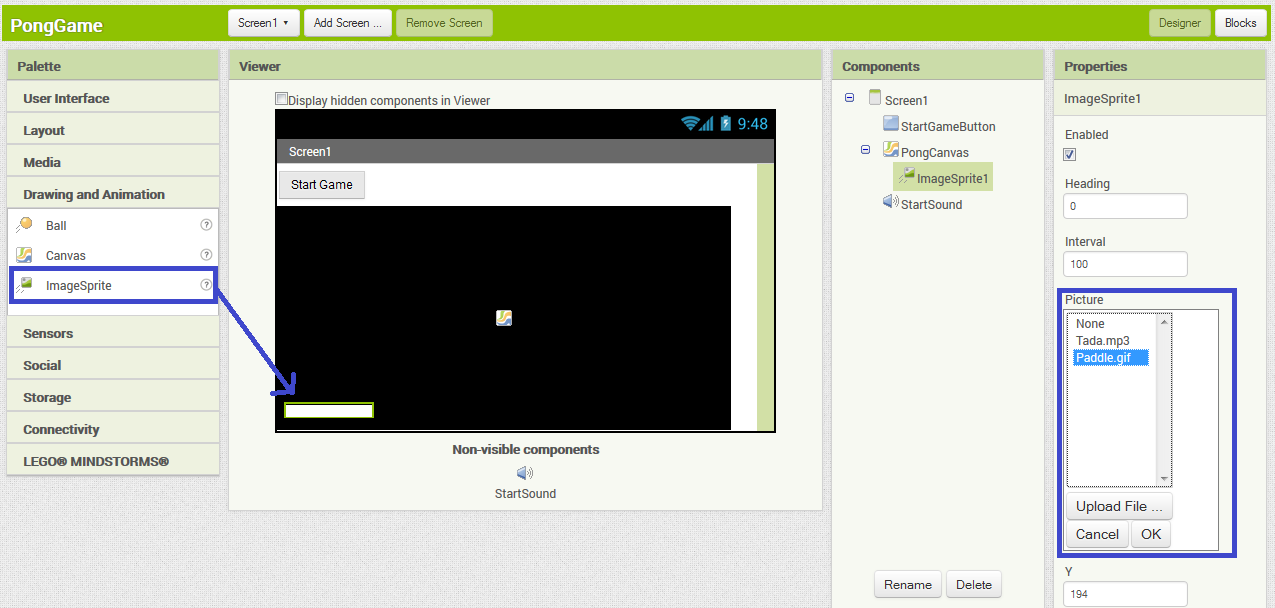
Change the size of the canvas by adjusting the Width and Height boxes in the Properties Window highlighted in blue below. To specify the exact size of pixels, click on the textboxes below Width and Height, choose the third option and fill in the textbox next to pixels. Use 450 for the Width and 220 for the Height as shown below.



Next, change the background colour of the canvas to Black (or your preferred colour) by clicking the text underneath the BackgroundColor in the Properties window, highlighted in blue above.

Now we’re going to get a paddle on the canvas, and get it moving.

To get a sprite on the canvas, click on the *Drawing and Animation* tab in the Palette window, select the ImageSprite component and drag it across to a position on the bottom of the canvas. The ImageSprite will not have an image of a paddle unless we give it one to use, so click in the box highlighted in blue below and click the Upload File button. Navigate to the directory where you extracted the activity resources and select the Paddle.gif image. Click OK and you should now have a canvas, which looks something like the one below.



Rename the ImageSprite to Paddle using the Rename button.

The next step is going to be to add a component to allow you to control the paddle’s movement by tilting the tablet. To do this, we are going to use the Accelerometer component, which gives us:

Acceleration in the X-Axis: This will be positive when the tablet is titled to the right, and negative when tilted to the left.

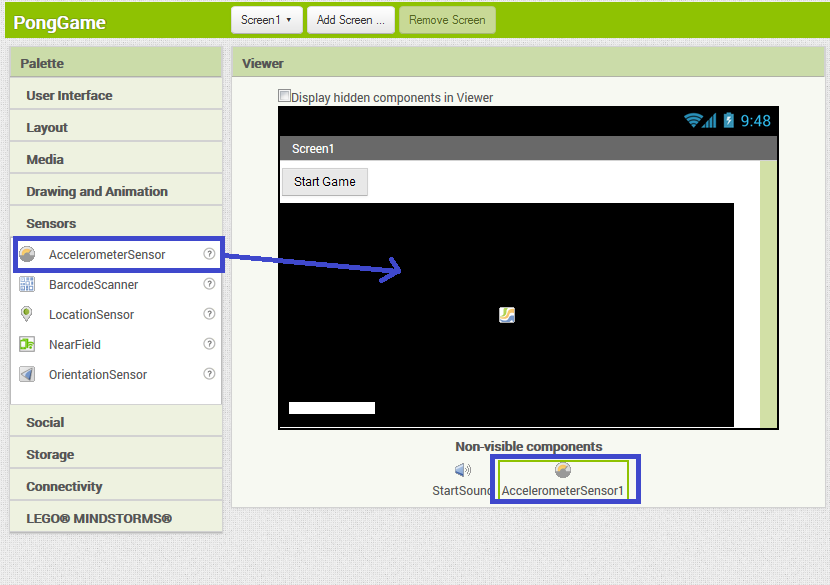
Acceleration in the Y-Axis: This will be positive when its bottom is raised, and negative when its top is raised.

Acceleration in the Z-Axis Positive when the display is facing up, and negative when the display is facing down.

In this example we are only concerned with tilting the tablet from side to side so we will only need to worry about the Acceleration in the X-Axis.

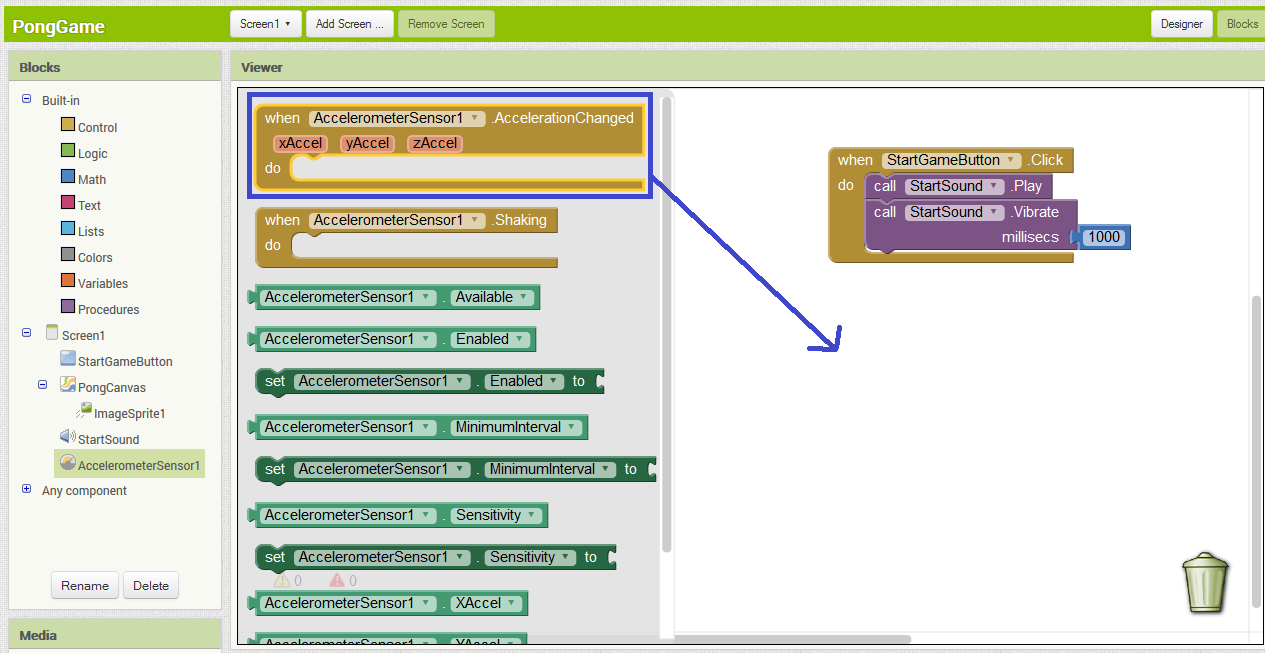
Alternatively, we could use the OrientationSensor, as it would give us the angle that the tablet is at, which we could use to move the paddle. However, using the AccelerometerSensor seems a bit more straightforward, and will allow us to move the paddle faster the more tilted it is.

To use the AccelerometerSensor, select the Sensors tab in the Palette window, select AccelerometerSensor and drag it to the canvas. If it added successfully, it will be listed in the Non-Visible components section as AccelerometerSensor1 as highlighted in blue below.



Now we are going to add code to the App to move the paddle using the AccelerometerSensor.

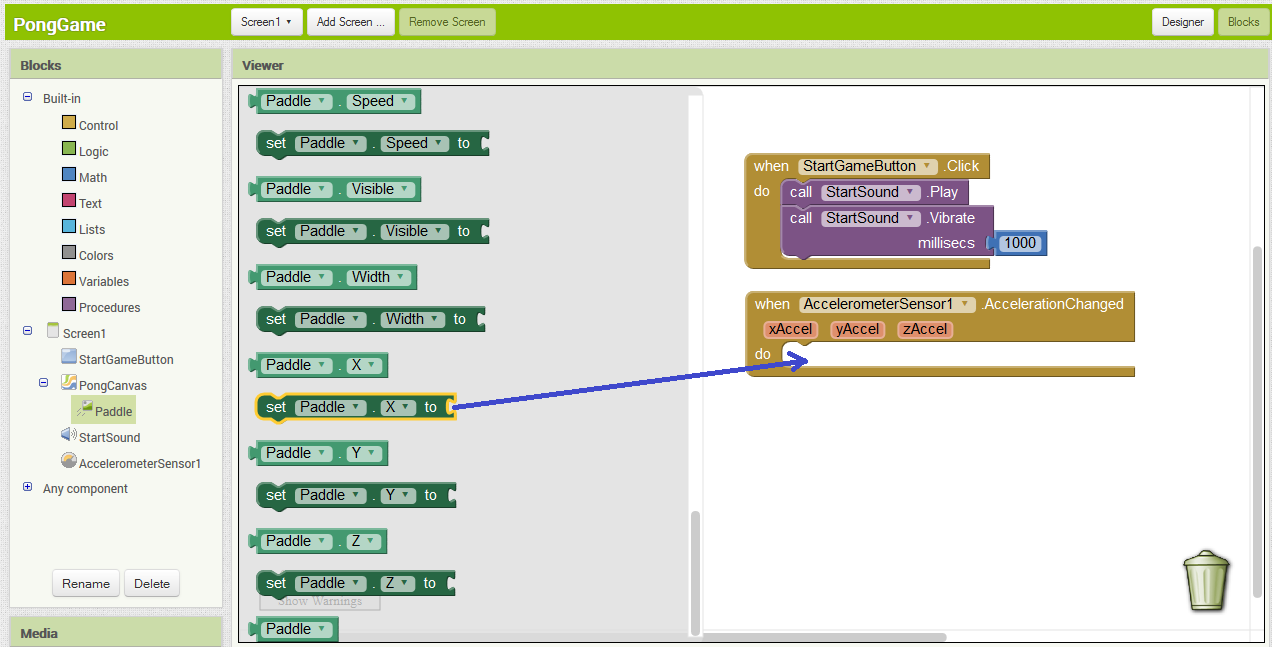
Now navigate to the Blocks view by selecting AccelerometerSensor1 in the Blocks window, and drag the AccelerometerSensor1.AccelerationChanged block across on to the code area as pictured below.



This event will automatically set the values of the variables XAccel, YAccel and ZAccel to the acceleration of the respective axes.

Now we are going to add code to make the paddle move according to how much the tablet has been tilted. First we are going to add a setter block, which we can use to change the x position of the paddle on the canvas.

To do this, select the Paddle in the Blocks window, and drag the block *set Paddle.x* blockinside the AccelerationChanged event handler block, and it will click into place.

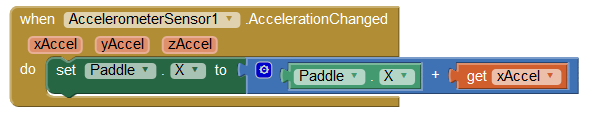


We want to change the x-coordinate of the paddle to a point on the canvas that is a few pixels away, moving the paddle across.

To do this, we will need to add the current position of the paddle with the acceleration speed, giving us the position the paddle needs to move to.

To do this sum, select Math and drag a + block and snap it into the Set *Paddle.X To* block. Now click Paddle, and drag the Paddle.X block into the first spot in the + block.

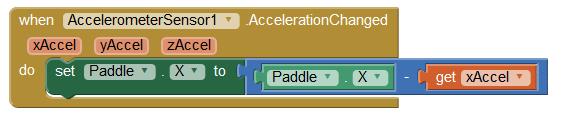
Now click on the xAccel button in the AccelerometerSensor1 AccelerationChanged block, select the get xAccel block and drag it into the second open spot on the x block.   
  
You should now have something like the following:



Now, tilt the tablet when the app has updated and see what it does. You will probably notice that it seems to move the opposite way that’d you’d expect.

The controls may seem unintuitive at this point (at least they did to me), because of the way the AccelerometerSensor gives you a positive value when tilting downwards to the right (ie when the tablets left side is raised).

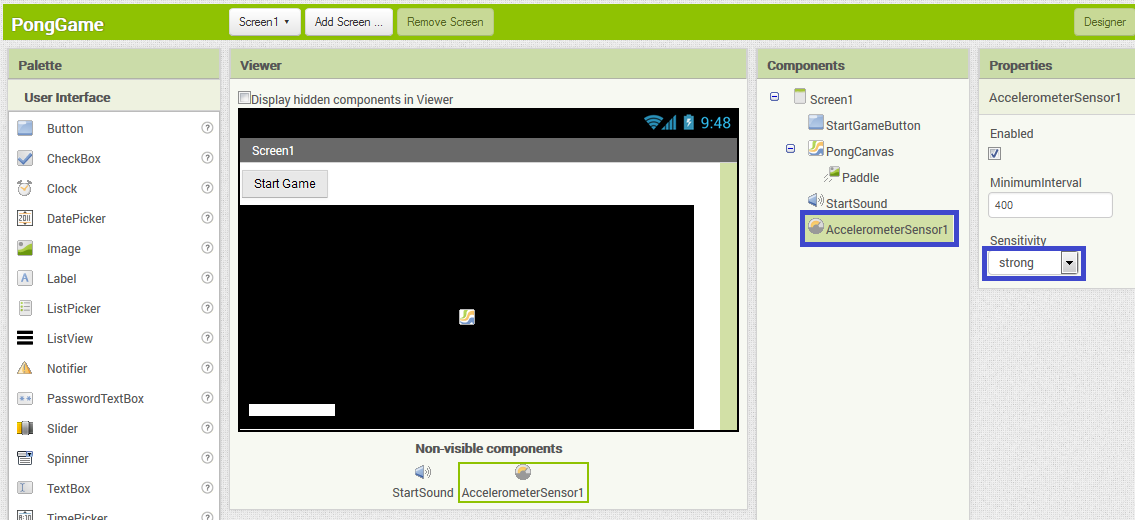
Instead of using a plus block, and summing the Paddle.X and xAccel we could use a minus block. Replacing the plus block should mean we end up with blocks like this:



Once you have made this change, tilt the tablet again and you will see that the paddle moves to the side of the tablet you tilt downwards.

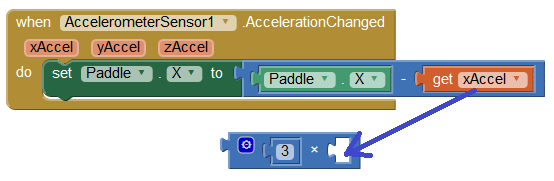
One thing that could be tweaked is that the paddle seems to move quite slowly, so we should probably adjust how quick it moves. A quick way of speeding it up is by adjusting the sensitivity of the Accelerometer, so that the acceleration in the x-axis is larger as you tilt the tablet.

To adjust the sensitivity move back to the Designer view by clicking Design in the top-right corner and select Accelerometer1 in the Components window. Select strong from the Sensitivity dropdown as pictured below:

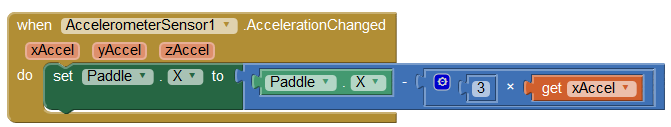


We can do this by making the value of the xAccel bigger by multiplying it when adding to the x-coordinate of the paddle.   
  
You may find that you would like the paddle to move quicker and to react more sensitively when you tilt the tablet. If so, go back to the Blocks view and click on select Math and drag an x (multiply) block onto the code area. Select a number block (also in the Math section) and drag it on the code area, double click the 0 in the number block and change it to any number you’d like. I suggest 3, but you can tweak this to whatever you like - the higher this number is, the more sensitive and quicker the paddle will move.

Put the number block into the first space and move the xAccel variable block into the second space of the multiply block, as pictured below:



Now place the multiply block into the second space of the minus block, as pictured below.



Run the App again and see if you are happy with the speed that the paddle moves. If it’s too slow you can increase the number above that is multiplied by the xAccel, or if it’s too fast you can decrease it.

Now we’ve got a paddle moving, let’s move on to Part 2, where we will get the ball moving.