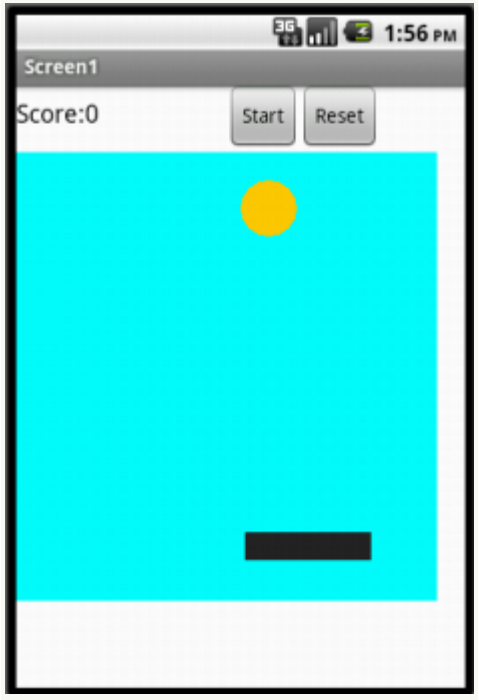
Pong



Pong is a simple game, which consists of a paddle (which is controlled by the user) and a ball. The ball bounces off the paddle and three walls. If the ball hits the paddle, the user gains points; if the ball hits behind the paddle, the game is over.

Learning Goals

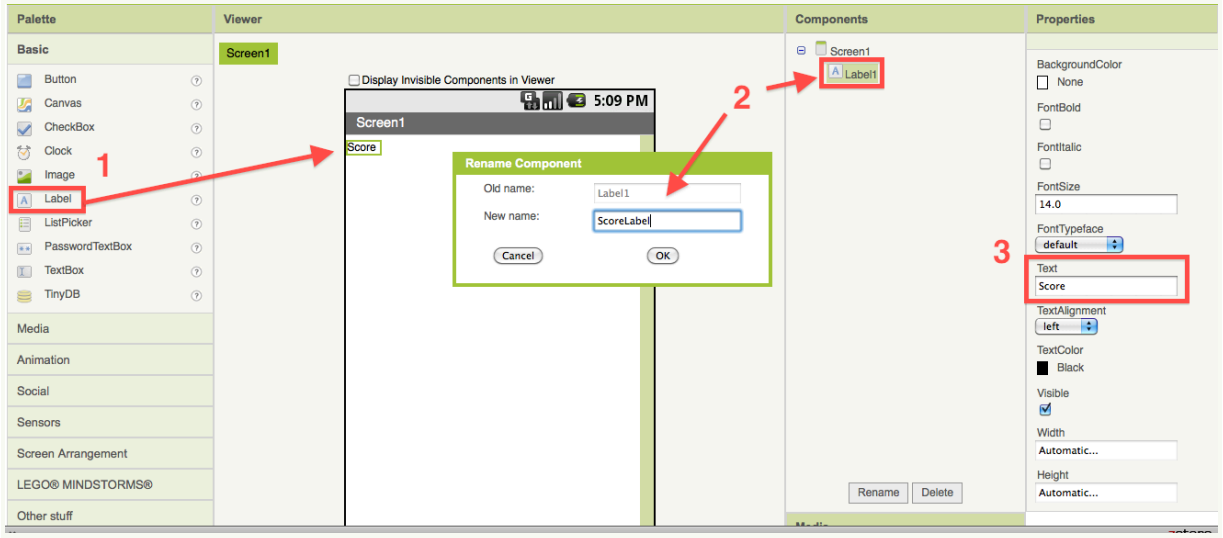
Completing this app will help you learn about:

- App Inventor environment: designer, blocks editor, emulator and/or physical phone
- App Inventor components: canvas, buttons, labels, animation sprites, procedures with no parameters, global variables, and conditionals

Lesson Details

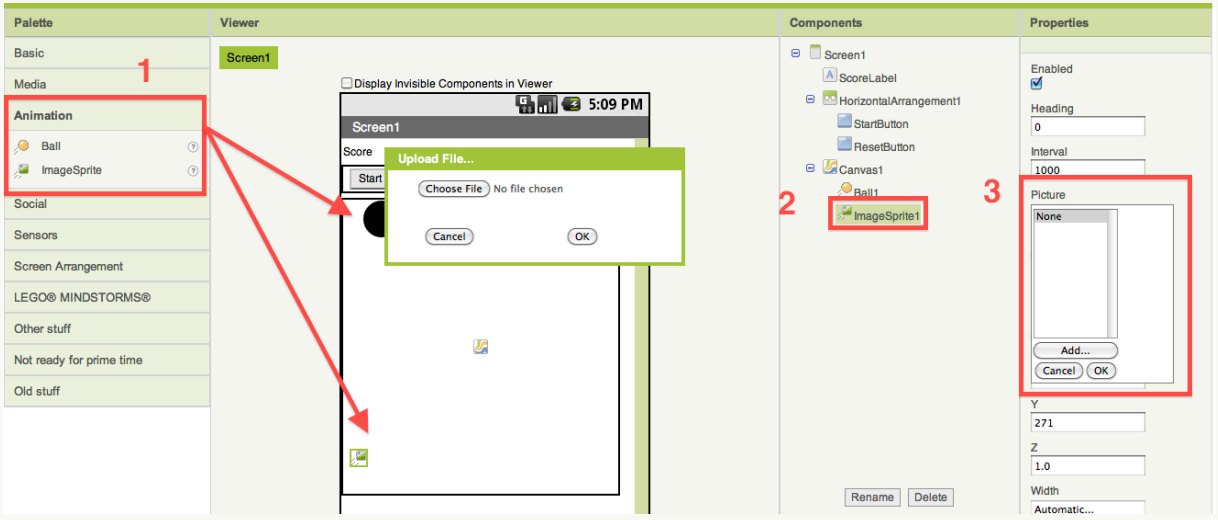
DESIGN: App Inventor Designer

1. To open the App Inventor Designer window, go to http://appinventor.mit.edu and sign in with a Google ID (Gmail account).
2. Download and save an image of a paddle here.
3. On the left column of the Designer, open the Basic palette, and first drag a Label component over to the Viewer (#1). Under the Components pane, highlight the Label1 component and change its name to "ScoreLabel" (#2). Under the Properties pane (right hand side), delete the display text of Label1 component to replace with “Score" (#3) and change its font size to 18, its width to 150 pixels, and height to be 30 pixels.



4. From the Basic palette, drag two Buttons to the Viewer under the labels. Rename the first button to "StartButton" and change its text field to "Start." Rename the second button to "ResetButton" and change its text field to "Reset."
5. Under the Screen Arrangement palette, drag a Horizontal Arrangement component to the viewer. Drag the ScoreLabel into the Horizontal Arrangement first, then drag the two buttons into the Horizontal Arrangement component next to the ScoreLabel component (you can see this under the Components pane too).

6. From the Basic palette, drag a Canvas component and set the width to 300 and the height to 390 pixels under the Properties pane. You can change the background color of the canvas to any color you like.
7. Under the the Drawing and Animation pane, drag a Ball and Image Sprite components onto the Canvas component in the Viewer (#1). Highlight the Image Sprite component (#2) and change its image to paddle.gif (#3). You can download the image here if you have not downloaded it yet.

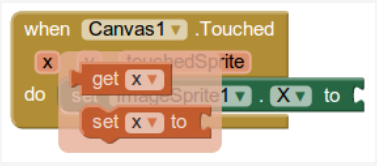


8. Highlight the Ball1 in the Components pane and go to the Properties pane to set the heading to 30, interval to 50, radius to 20, and speed to 5. You can change the color of the ball (PaintColor) to any color you like.

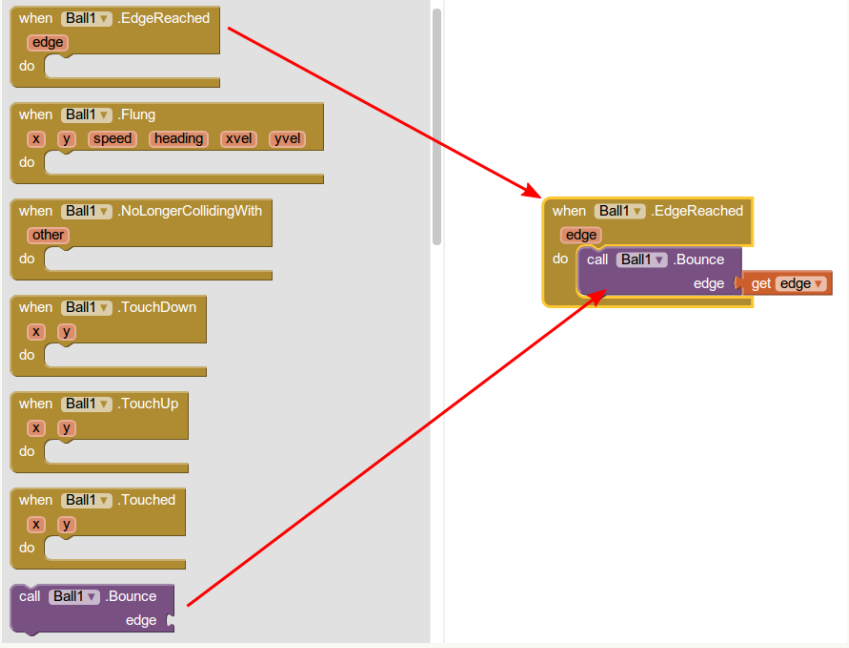
BUILD: Blocks Editor

In the upper right corner of the Designer, click on the Blocks button. Wait for a few moments while the blocks editor loads. Once the Blocks Editor is open, there are several options running along the left side of the screen.

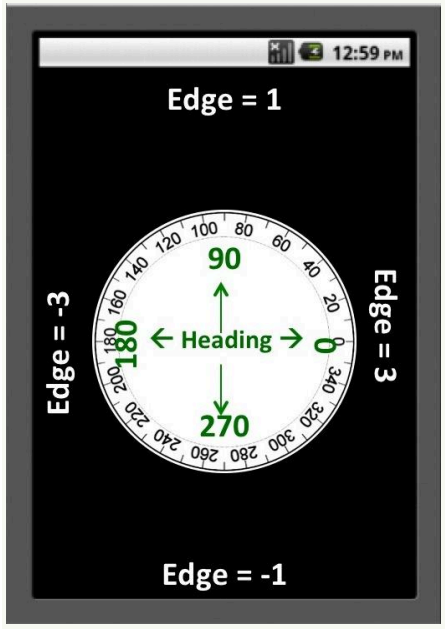
1. From the Screen1 palette, open the Canvas1 drawer and drag the when Canvas1.Touched block into the work area. This block automatically comes with three blocks: the name x, name y, and name touchedSprite blocks.
2. From the Screen1 palette, open the ImageSprite1 drawer and drag the set ImageSprite1.x to block into the "do" socket of the when Canvas1.Touched block.
3. From the when Canvas1.Touched block drag the get x block into the edge of the set ImageSprite1.x to block. They will click together like magnetic puzzle pieces.



4. From the Screen1 palette, open the Ball1 drawer and drag the when Ball1.EdgeReached block over the work area, as well as the call Ball1.Bounce edge block. After this is done, insert the get edge block from when Ball1.EdgeReached block into the socket of the call Ball1.Bounce edge block that is asking for an edge value. Bounce is a built-in function for sprites. It tells the sprite to bounce off the wall.



The *When Ball1.EdgeReached* block automatically generates the *edge* variable. When the ball reaches an edge it needs a way to report that information to the main program. It does this by storing a number in the *edge* variable. See the image below to learn about App Inventor canvas edge values.

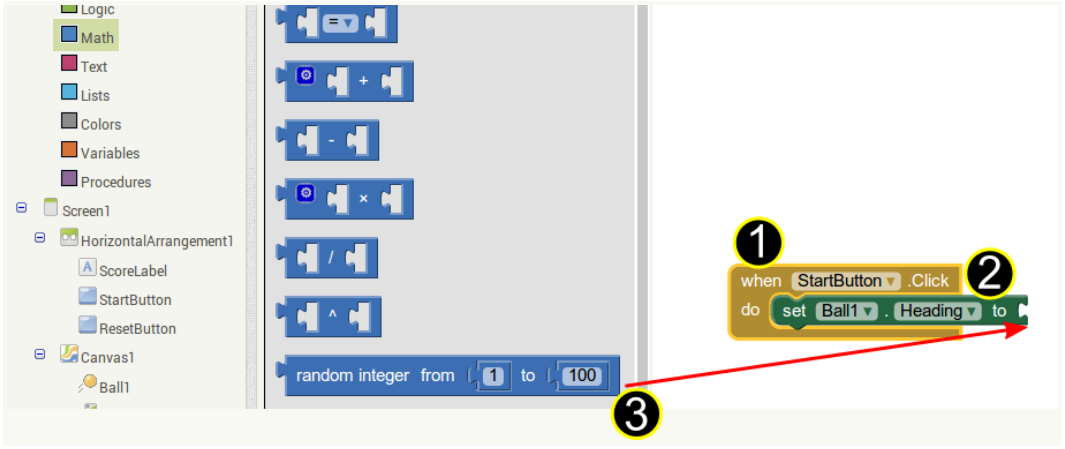


App Inventor assigns numeric values to the edges of a canvas as follows:
top = 1, right = 3, bottom = -1, left = -3

Heading values for animated objects go in a full circle like a compass, with values between 0 and 360 degrees.

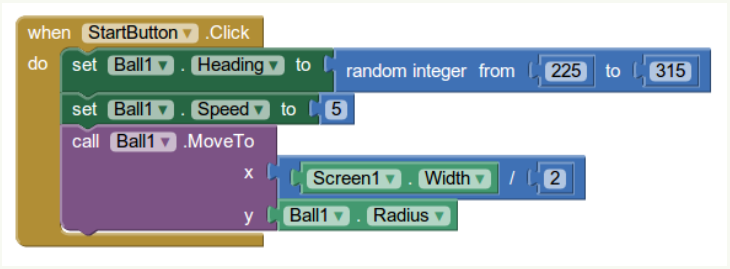
An object moving toward the top of the screen is said to have a heading of 90 degrees.

5. From the Screen1 palette, open the StartButton drawer and drag the when StartButton.Click block over to the Viewer (#1). Click on the Ball1 drawer and drag out a set Ball1.Heading to block and place it at the "do" section of the when StartButton.Click block (#2). From the Built-In palette's Math drawer, drag out the random integer block and add it to the end of the set Ball1.Heading to block (#3).

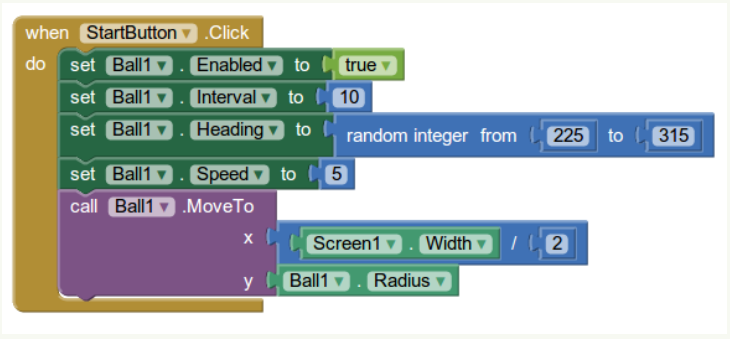


Type “225” into the number block for the “from” area and type "315" for the “to” area. This will make the ball take on a heading between 225 and 315 degrees when the start button is clicked (see heading diagram above).

6. From the Ball1 drawer, drag out a set Ball1.Speed to block and place it under the set Ball1.Heading to block. Pull out a number block from the Built-In palette's Math drawer, put it in the "to" socket and type in the value 5. This will move the ball 5 pixels in the direction of its heading each time its internal clock ticks. The "internal clock" of an object is called its *interval*. The Interval of a ball sprite is initially set at 1000 milliseconds (= 1 second) but you can change this in the properties of the sprite. Remember, at the beginning of this tutorial we changed the Interval property of Ball1 to 50ms.
7. From the Ball1 drawer, drag out a call Ball1.MoveTo block and put it under the set Ball1.Speed to block. From the Math drawer, drag out a division (/) block and put that in the "x" socket in the call Ball1.MoveTo block. Open the Screen1 drawer and drag out a Screen1.Width block and drop that in the first blank area in the division block. On the second blank area put a number block and type "2" (a number block appears with "2" in the text box). From the Ball1 drawer, drag out a Ball.Radius block and drop that into the "y" area of the call Ball1.MoveTo block. This will start the ball in the middle of the screen near the top each time you hit the start button. (Do you understand why? The "MoveTo" block tells Ball1 to move to the x coordinate that is half the width of the screen, and the y coordinate that is equal to the radius of the ball itself -- which is a low number so the ball will appear very close to the top of the screen).

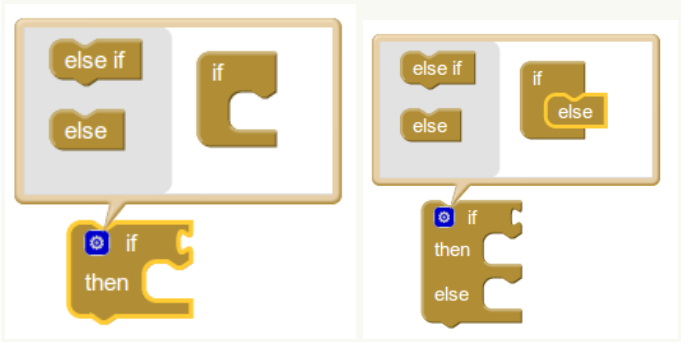


8. From the Ball1 drawer, drag out a set Ball1.Enabled to block. From the Logic drawer, drag out a “true” block and put it in the "to" socket. This will start the ball moving.
9. From the Ball1 drawer, drag out a set Ball1.Interval to block and set it to the number 10. This will cause the ball to move every 10 milliseconds (and it will move the number of pixels specified in the "speed" property). After the interval block is created, move the set Ball1.Enabled to and set Ball1.Interval to blocks onto the top of the set Ball1.Heading to block, so that they appear as the first blocks in the when StartButton.Click event block.



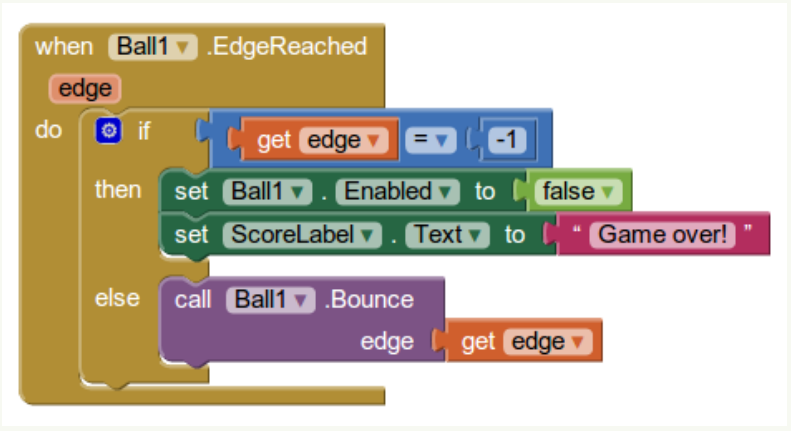
10. From the ImageSprite1 drawer, drag out a when ImageSprite1.Dragged block and drop it in the work area. From the same drawer, drag out a call ImageSprite1.MoveTo block and drop it in the when ImageSprite1.Dragged. Drag out the get CurrentX block from when ImageSprite1.Dragged and drop it at the "x" slot. From the ImageSprite1 drawer, drag out a ImageSprite1.Y block and drop it at the "y" slot. This will move the paddle in x (horizontally) when you drag it, but not move it in y (vertically).
11. From the Ball1 drawer, drag out a when Ball1.CollidedWith block and drop it in an open area. Drag out the set other to block from when Ball1.CollidedWith and drop it into the socket. From the ImageSprite1 drawer, drag out a ImageSprite1 block and drop it at the slot of the set other to block.
12. From the Ball1 drawer, drag out a set Ball1.Heading to block and drop it inside the when Ball1.CollidedWith block. Click on the Built-In palette and open the Math drawer, drag out a subtraction block and put it in the set Ball1.Heading to socket. On the first blank area put a number block and type "360". From the Ball1 drawer drag out a Ball1.Heading block and drop it in the second blank area. This will reverse the ball when it hits the paddle. (Do you understand why? Look at the heading diagram above to see what happens when you subtract 360 from any heading. Headings must be between 0 and 360, and a negative number is treated as if it is positive).
13. In your workspace, find the place where you put the when Ball1.EdgeReached block. We're going to add some new blocks to this event, so for now drag the call Ball1.Bounce block away from when Ball1.EdgeReached so that it becomes separated. Leave the call Ball1.Bounce block loose in the work area for now.

Under the Built-In palette, open the Control drawer and drag out an if then block. Convert it in a if then else block as shown below.



From the Math drawer, drag out an equal (=) block and drop it into the "test" socket. Drag out the get edge block from when Ball1.EdgeReached block and drop it in the first blank area. On the second blank area type "-1" (for the bottom or southern edge). This will test to see if the edge that was reached is the bottom edge of the canvas.

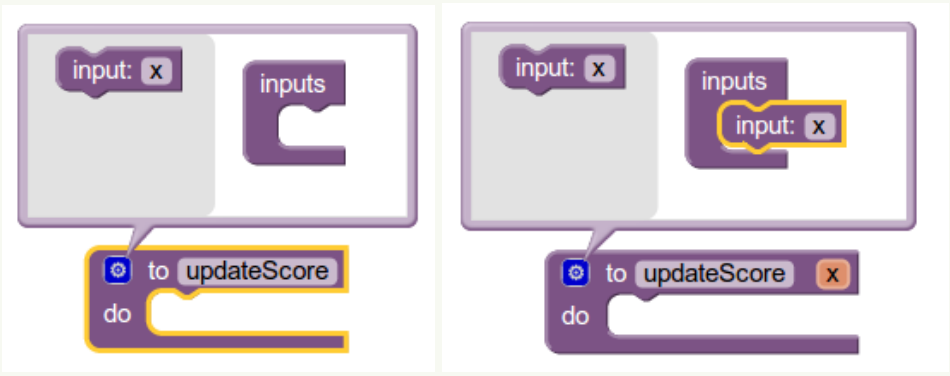
14. From the Ball1 drawer, drag out a set Ball1.Enabled to block and drop it in the "then" area of the if then else block. From the Logic drawer, drag out a “false” block and put it in the "to" socket. This will stop the ball from moving when it gets past the paddle. In addition, drag out a set ScoreLabel.Text to block from the ScoreLabel drawer, and attach it underneath the set Ball1.Enabled to block. From the Built-In palette, open the Text drawer, and drag out a text block and drop it after the "to". Click the text and change it to "Game Over!" This text will appear on the screen in the ScoreLabel when the ball gets past the paddle.
15. Now grab the orphaned Ball1.Bounce block that has been sitting alone in your workspace and drop it into the "else" socket of the if then else block. It should still have the edge block attached to its "edge" socket. This entire if then else block will cause the ball to bounce off of all edges except the bottom (southern) one. Steps 13 through 15 should make a set of blocks that looks like this:



16. From the Built-In palette, open the Variables drawer, drag out a initialize global to block and drop it in an open area. Click on "name" and type "score" to change the name of the variable to score. Put a number block with the value 0 in it into the "to" socket. This creates a variable named "score" and sets its

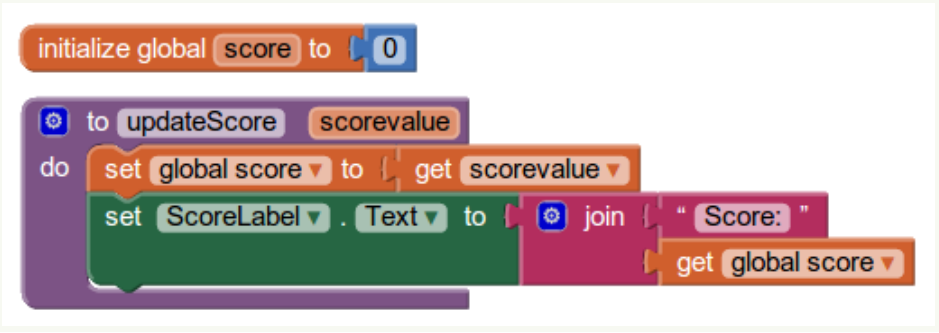
value to 0.

17. Under the Built-In palette, open the Procedures drawer, and drag out a to procedure do block and drop it in an open area. Click on "procedure" and rename it to "updateScore". Then create a parameter for the procedure as shown below:

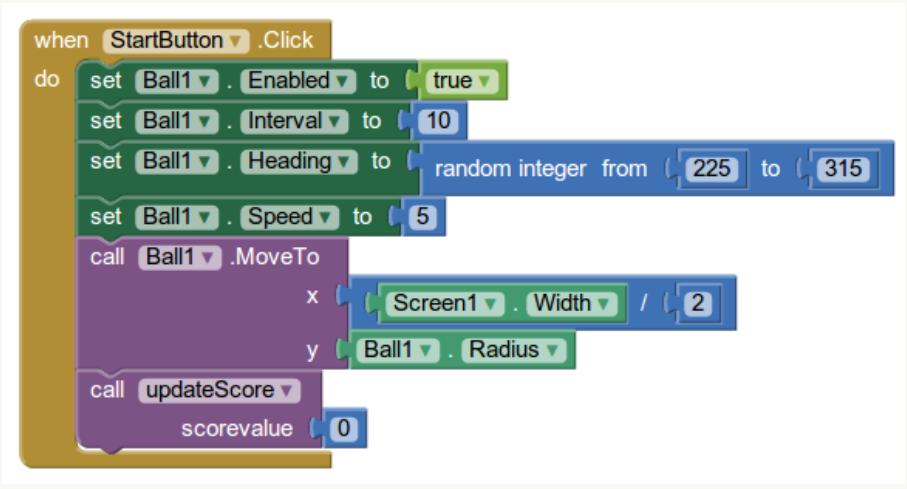


Change the “x” to “scorevalue”. This creates a parameter for the procedure that is named "scorevalue". A parameter is a temporary variable that holds a value for a procedure. The value is specified when the procedure is called.
From Variables drawer drag out a set to block, choose the “global score” variable and drop it in "do" area of the updateScore procedure. Drag out a get scorevalue block from the procedure and drop it in the "to" area. This sets the score variable to the passed value.

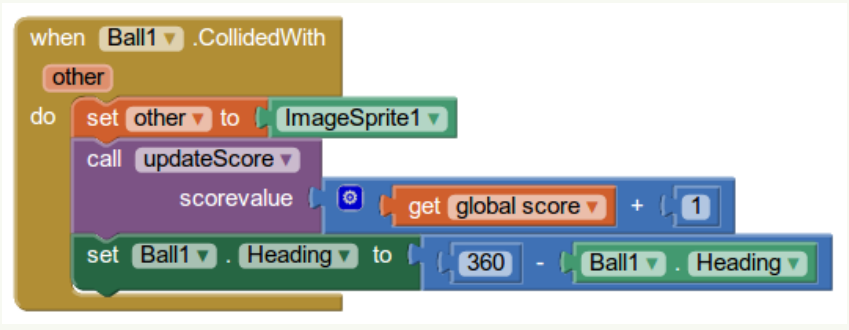
18. From the ScoreLabel drawer, drag out a set ScoreLabel.Text to block and drop it after the previous block. From the Built-In palette, open the Text drawer, and drag out a join block and drop it after the "to". Set the first blank area to the text "Score:" and set the second blank area to get global score from the Variables drawer. This will set the text of the score label to a string that joins together "Score:" and the actual value of the score variable.



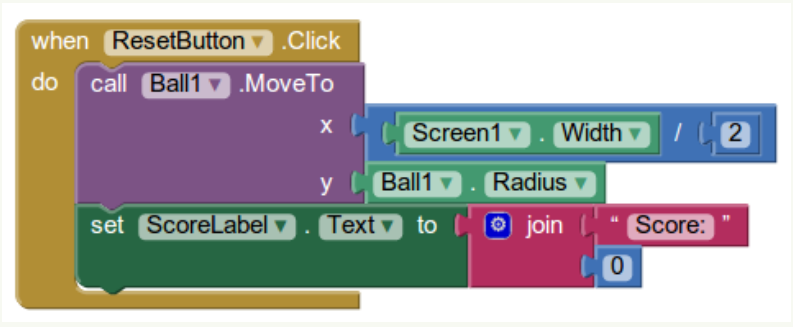
19. Open the Procedures drawer, and drag out a call updateScore block and drop it at the end of the when StartButton.Click next to the block and type 0 to update the score to 0.



20. Open the Procedures drawer, drag out another call updateScore and drop it in the when Ball1.CollidedWith block. Under the Built-In palette, open the Math drawer drag out an addition (+) block and drop it at the end of the call updateScore block. From the Variables drawer, drag out a get global score block and drop it in the first blank area. Click in the second blank area and type 1. This will increment the score by 1.



21. Open the ResetButton drawer and drag out the when ResetButton.Click to the open area. Go back to the set of blocks surrounded by the when StartButton.Click block that you created before, highlight the block call Ball1.Move to and hit control (ctrl) key and c key together to copy the block. Then click the open area (anywhere) and hit control (ctrl) and v key together to paste the block you copied. Drag the whole copied block in the when ResetButton.Click.
From the ScoreLabel drawer, drag the set ScoreLabel.Text to block and place it under the previous block. Open the Text drawer and drag out a join block and drop it after the set ScoreLabel.Text to. Set the first blank area to the text "Score:" and set the second blank area to the number zero. (Note: Would it also work to call the procedure UpdateScore with 0 as the scorevalue parameter? Why or why not?).



Done! For a little fun, try changing the color or size of the ball or paddle to personalize your app.

Challenge 1: Changing the speed and the size of the ball!

Can you increase the speed of the ball and decreases the size of the ball when the score increases an increment of 10 ? (Hint: Under the Math drawer, there is the modulo block. The remainder option returns the result of dividing the two numbers and taking the remainder. For example, remainder(11,5) = 1, remainder(-11, 5) = -1, remainder(11, -5) = 1, and remainder(-11, -5) = -1).

Challenge 2: Adding sounds!

Download the audio files from the App Inventor Media Library page, and upload them to the Designer. Make the "noink" sound play when the ball hits the edge of the wall, the "ta-da" sound when the speed of the ball increases, and the buzzer sound when the ball hits behind the paddle.

More Challenges:

- Give the player three lives so that they get three tries before "Game Over".
- If you program multiple lives, decrease the score by 1 each time the player loses a life.
- Investigate what happens if you change the range of random numbers for the start heading when the start button is clicked.
- Try making the app respond to titling of the phone instead of dragging the paddle

Improve this game - fix the "bug" that happens if you click the start button after a game is over:

Notice that when the game is over, if the player clicks "start" instead of "refresh", the message "Game Over" remains on the screen until the player scores the first point. How can this be fixed? Is it really necessary to have a start button and reset button?