Assignment 2

Due: Sunday, July 5th, 2020 by 11:59 PM CDT

- Submit **one PDE file** for **the highest programming question completed.**
- Submit **one PDF file** with your answers to the written questions.
- Assignments must follow the programming standards document published on the course website on UM Learn.
- After the due date and time assignments may be submitted but will lose 2% of marks per hour late or portion thereof. The date and time of the last file submitted controls the mark for the entire assignment.
- You may submit a question multiple times, but only the most recent version will be marked.
- These assignments are your chance to learn the material for the exams. *Code your assignments independently*. We use software to compare all submitted assignments to each other, and pursue academic dishonestly vigorously.

Part 1: Programming

Hand in one **pde file** for the highest question completed. Name your program using your last name, assignment number, and the highest question number that you completed/attempted, for example: **LastnameFirstnameA2Q3.pde**. Submit this file in the Assignment 2 submission folder on UM Learn.

The marker will run your programs, and may change the canvas size and the value of the constants. It should still work if any constants are changed in a reasonable way.

This assignment includes material from **Units 1-11**, including floating point variables, Boolean variables, trigonometry, and if-statements.

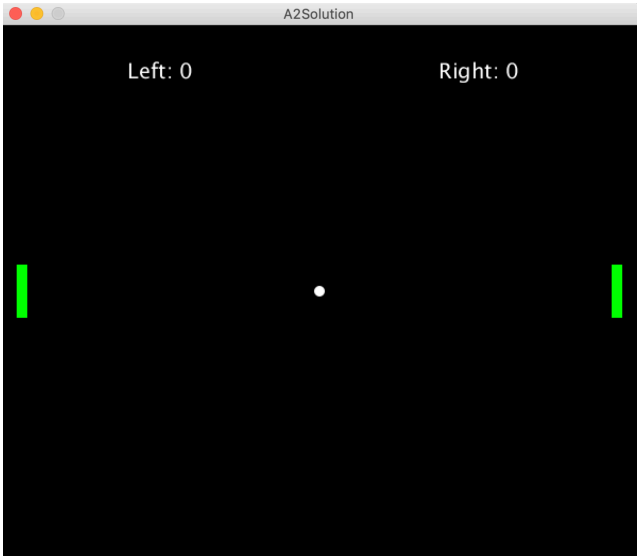
Background: Pong

Pong was one of the first arcade video games (https://en.wikipedia.org/wiki/Pong) and simulates a game of table tennis. We're going to create a similar game, in which two players control paddles that move vertically on opposite sides of the screen. A ball is hit back and forth, bouncing off the paddles. If the ball gets past an opponent's paddle, that player receives a point. The game ends when one player reaches 11 points.

Each question in this assignment builds on the previous question. Make sure each question is fully working before moving on to the next.

You must use functions. Each function should have a single purpose. Your draw block should be very simple, with calls to your other functions that do the work for you.

Question 1: Game Setup [2 marks]



Set up an Active Processing program that will draw the initial configuration of the game.

Write a **`drawGame()`** function that will draw the canvas, similar to the one shown at right.

You are responsible for drawing the ball and paddles. A function to draw the score is provided below.

Create globals to store:

- the position of each paddle
- the size of the paddles
- the size of the ball, and
- the score for each player called **`leftScore`** and **`rightScore`**.

(Which globals should be variable and which should be constant? Should they all be floats, or should some be integers?)
Choose your own colour scheme and a reasonable paddle size. Place the paddles a reasonable distance from the edge of the canvas. The ball will begin in the centre of the canvas.

Copy and paste the following **`drawScore()`** function into your program. Call it from your **`drawGame()`** function. We will learn about text in Unit 8.

```
void drawScore(){
    textSize(20);
    String toPrint = "Left: " + leftScore;
    text(toPrint, width/4-textWidth(toPrint)/2, 50);
    toPrint = "Right: " + rightScore;
    text(toPrint, width*3/4-textWidth(toPrint)/2, 50);
}
```

In order for the **`drawScore()`** function to work, you must keep track of the score in global variables named **`leftScore`** and **`rightScore`**.

Question 2: Moving the Paddles [5 marks]

The **left paddle** will be controlled by the keyboard. When the W key is pressed, the paddle will move up. When the S key is pressed, the paddle will move down. The paddles should remain on the canvas at all times.

To move the left paddle:

- We already know that we can check if a key is pressed by looking at the value of the built-in **`keyPressed`** boolean variable.

- If a key is pressed, we can then look at which key was pressed.
- Processing provides a built-in variable called **`key`** that will tell you which key was pressed. You can check its value like this:

```
if (key == 'a') {
   // ... do something
}
```

- Note that the letter of the key you're looking for is lowercase in single quotation marks.
- Make sure to check if a key is pressed before looking at the **`key`** variable.
- When the correct key is pressed, make the left paddle move. Choose how fast the paddles should move (i.e. how many pixels per frame). I used 3 pixels per frame.

The **right paddle** will also be controlled by the keyboard. When the I key is pressed, the paddle will move up. When the K key is pressed, the paddle will move down.

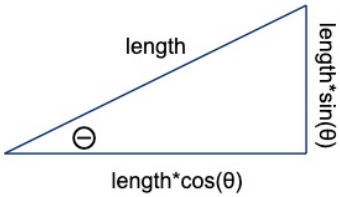
To move the right paddle use the same steps as above for the left paddle. Both paddles should move at the same speed.

Caution: Do not use the **`keyPressed()`** function to move the paddles. This function will only run once each time you press a key, moving your paddle only one step at a time and making the game frustratingly slow. You want the ability to hold a key and have your paddle move for as long as the key is held. Be sure to test the **`keyPressed`** variable in a function that is run in every frame.

Question 3: Add the Ball [4 marks]

Make the ball move, and bounce off of the paddles and the top and bottom of the canvas.

You have already set the initial position for the ball as the centre of the canvas. Set an initial direction for the ball, making sure that the ball has enough movement in the x direction to make the game interesting. You don't want the ball bouncing up and down many, many times as it crosses the screen. There are many ways to accomplish this, and you are encouraged to experiment. One possible method to set the direction of the ball is:



- Generate a random direction (**Θ**) between **`0`** and **`TWO_PI`**
- Convert this direction to x and y speeds using trigonometry. **HINT**: Think of the total ball speed as the hypotenuse in a right angle triangle.
- Check to make sure that the direction (**Θ**) is such that the y component is not too large and the x component is not too small, because this would make a boring game with the ball bouncing up and down and not really going left and right. If the motion is too vertical, set the ball direction to a more interesting angle. Choose your own limits.

Test that at this point your program has the ball moving in a random straight line away from the centre of the canvas. Then...

Make the ball bounce off the top and bottom of the canvas:

- If the ball hits the top or bottom walls, multiply the y component of the movement by -1 to simulate a bounce.

Make the ball bounce off the paddles. Checking if the ball hits a paddle actually can get quite complicated. We will do a simplification:

- Check if the centre of the ball hits the paddle. [**Hint**: This is identical to checking if the mouse hits a button drawn on the canvas.]
- If the centre of the ball hits a paddle, multiply the x component of the movement by -1 to simulate a bounce (for now... you will change this below).

Test your program thoroughly at this point!

Question 4: Make it Interesting! [3 marks]

When the ball hits a paddle, rather than simply bouncing with the same angle and speed, add some variety. Make the speed and direction of the ball change when the ball bounces off the paddles.

When the ball hits a paddle, choose a **new direction**:

- Choose a new random direction, and recalculate the x and y components of the speed.
- Use the same limits on the direction that you used when first setting the ball in motion, but also make sure that the new direction is appropriate for the paddle the ball hit (i.e. if the ball hit the left paddle, the ball must end up moving right).

When the ball hits a paddle, choose a **new speed** according to where it hit the paddle:

- Define two new global constants, **`MIN_SPEED`** and **`MAX_SPEED`**, at the top of your program.
- The ball speed should be set to **`MIN_SPEED`** when the ball hits the centre of the paddle, to **`MAX_SPEED`** when the ball hits the edge of a paddle, and it should change smoothly in between. [**Hint**: Calculate the distance from the centre of the paddle to the centre of the ball, and scale that distance into the speed.]

Question 5: Keep Score [3 marks]

If the ball makes it past a paddle, increase the appropriate score (**`leftScore`** or **`rightScore`**), and generate a new ball in the centre of the canvas.

When one player reaches 11 points, the game should end. Use a **`boolean gameOver`** variable to control the end of the game. When this variable is set to **`false`** the game should run normally. When **`gameOver`** is set to **`true`**, the game should be frozen. Don't move the ball, or allow the players to move the paddles.

[Programming Standards are worth 5 marks]

Part 2: Written Questions

Save your answers to these questions as a **PDF document** named **LastnameFirstnameA2.pdf** and submit this file in the Assignment 2 submission folder on UM Learn.

Question 6: Exploring Beyond Earth [3 marks]

Often code does not work correctly on the first run, even after careful planning. Syntax errors are usually easy to find because they're caught by the computer, but runtime errors are harder to solve.

(a) Describe three different steps you can take to find and correct runtime errors. ("Ask someone" is not an appropriate answer – describe how **you** can find and correct the errors.)

(b) Computer scientists call the process of correcting errors in code "debugging". What is the name of the person who recorded the very first computer "bug"?