KidzCoding Hour of Code - Pong

Activity:

In this activity, students will learn how to make pong in **Scratch**. They will learn how to use **variables, if statements, logic, coordinate/degree system,** and **loops.** The game will consist of the player playing pong against an AI opponent that the student will code themselves. It will keep track of the score between the player and the opponent over the course of the game.

Learning Objectives:

* Learn how to use keyboard inputs to control characters
* Develop a strong understanding of the coordinate plane and degrees
* Understand how to use logic with if statements

Materials:

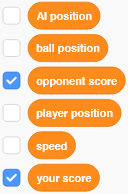
* Scratch Starter Project: <https://scratch.mit.edu/projects/877497456/>
* Finished Game: <https://scratch.mit.edu/projects/877468718/>

Coding Time!

Starting the project

Open the Scratch starter project, and you will have all of the included sprites and costumes.

Variables

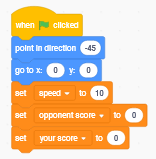


This game will require a lot of variables, but make sure that only “your score” and “opponent score” are checked. This will make them visible to the player as they play the game. “AI position” will keep track of the opponent AI’s paddle x-position, and “player position” will keep track of the player’s paddle x-position. “Ball position” will keep track of the ball’s x-position. “Speed” will be used to control how fast the ball is moving in the game.

**Coding the Ball**

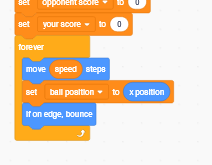
****

1. **Setting up the Ball**

****

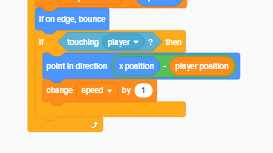
We want the ball to start in the middle of the screen, so we make it go to 0,0. It can point in any direction that you want, but -45 is a good starting direction. We also want to set the speed to 10, because it’s not too fast. The ball will speed up as the game progresses.

**2. Basic Movement**



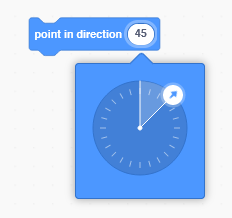
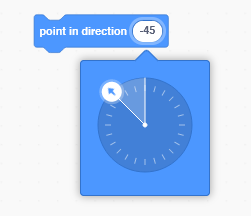
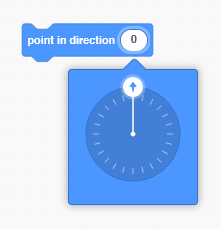
This step will control the basic movement of the ball. Since the speed variable controls how fast the ball moves, we will make it move the amount of steps equal to the speed variable. To keep track of the position of the ball, after every time the ball moves, we set the ball position variable equal to the x position of the ball. We also want the ball to bounce when it hits the sides of the screen, so we will use the “if on edge, bounce” block.

**3. Hitting the Player’s Paddle**

****

We’ll now tell the ball what to do when it hits the player’s paddle. What we want the ball to do is to bounce in a direction that matches where it hits the player’s paddle. For example, if the ball hits the left side of the paddle, it should bounce more to the left, and if it hits the right side of the paddle it should go right. If it hits in the middle, it should bounce straight upwards.

To do this, we need to understand how position works. In Scratch, position keeps track of the center of each sprite. Using this, we can find out which side of the paddle the ball is hitting. If the x position of the ball is less than the player position, this means that the ball is hitting to the left of the center of the paddle, so it should bounce left.



As you can see here, pointing in the direction 0 means that the ball is pointing straight upwards. Negative numbers mean that the ball is pointing more left, and positive numbers mean that its pointing more right. So, what we need to do is make the direction of the ball more negative the more left on the paddle the ball hits. To do this, we do (x position - player position).

As an example, if the ball’s x position was 120, and the player’s position was 130, we know that the ball would be on the left side of the player’s paddle. Now, if we did (x position - player position), we would get -10. As we showed before, a negative number makes the ball bounce to the left, which is what we want when the ball hits the left side of the player’s paddle.

Note:

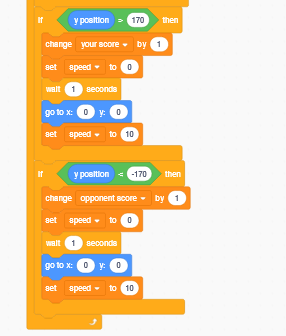
The paddle is 90 pixels wide. This means that each half is 45 pixels, so the bounce angle will never exceed 45 degrees on either side. This might be hard to understand, but this is because when you subtract the position of the player from the position of the ball, it will never be greater than 45. Therefore, the angle can’t be greater than 45 since we are using that subtraction value to control the angle of the ball.

**4. Hitting the Opponent’s Paddle**



When the ball hits the opponent's paddle, we’re going to do a similar process to what we did when the ball hits the player’s paddle, except the ball needs to bounce at a downwards angle. Try to understand why this code creates the right bounce angle for the ball. However, don’t worry if you can’t quite figure it out - this is using some very complex mathematical ideas.

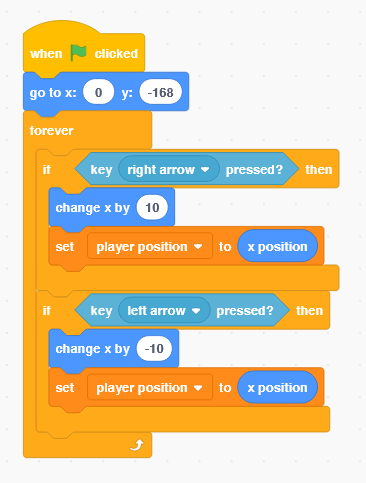
**5. Scoring Points**



This code is relatively simple; if the ball hits the top of the screen (y coordinate of 170), then the player gets a point. If the ball hits the bottom of the screen (y coordinate of -170), then the opponent gets a point. In both cases, the game will reset.

Coding the Player

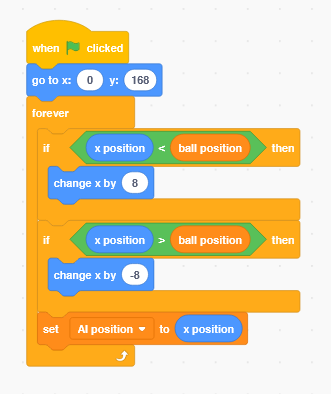
1. The Player



We’re going to be using keyboard inputs to move the player’s paddle. Everytime the player’s paddle moves, we need to update the player position variable because we’re using it for the ball to calculate the bounce direction.

Coding the AI Opponent

1. Detecting a Hit

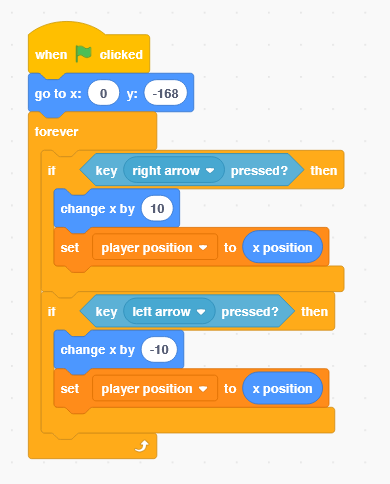


If the ball is to the right of the opponent’s paddle, the opponent will automatically move to the right. The same thing will happen on the left. We will also update the AI position variable because we’re using it to calculate the bounce direction for the ball.

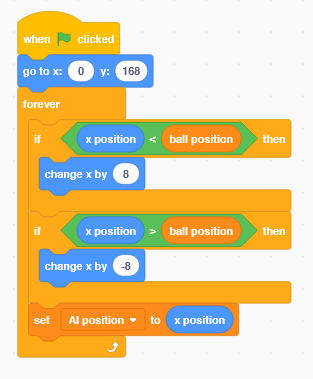
Congratulations, you have just completed an Hour of Code!

Complete Code:

Player:



Opponent:



Ball:

