Click Game part 1

Processing Practice Project

Let’s make a simple game to practice what we’ve learned so far.

Game Concept

The concept of this game is simple.

- A red circle will appear on the screen in a random location for one second, and then disappear for one second.
- You need to click on the circle as many times as you can before it disappears.
- You will get one point for each click, but you will lose one point for each time your click misses the circle.
- This will repeat 10 times.
- Your final score will be displayed.

Step by Step

The most important thing when doing a project like this, is to break it down into small, simple steps. Then, write code for each step.

Let’s begin by writing a program that displays a red circle. To keep things clean, we’ll create a method to display the red circle, called `showCircle()`

Step 1: A Red Circle

```
void setup() {
  size(800, 600);
}

void draw() {
  background(0);
  showCircle();
}

void showCircle() {
  fill(255, 0, 0);
  circle(width/2, height/2, 100);
}
```

Step 2: Register Clicks

Now let’s make our circle register a click. To do this, we will need variables for the circle’s x, y, and size values. (Later, these variables will be important to allow us to change the circle’s position and size.)

For this, we will use a built in Processing method `mouseReleased()`, which will detect when a mouse click is complete.

Inside of this method, we will use another Processing method, `dist()`. This method takes two x and y coordinates, and returns the distance between them. We will check the distance from our mouse to the center of the circle. As long as this distance is less than than radius of the circle (half of its size), that means our mouse is inside of the circle.

For now, we can test that this is working by printing “hit” and “miss” to our console.

To summarize:

1. Create three variables at the top for our circle’s x, y, and size.
2. Set the values for these variables is set in the `showCircle` method.
3. Create a new `mouseReleased` method.
4. Inside this method use and `if / else` to check if the mouse is inside the circle.
5. Print “hit” or “miss” to the console.

Our code now looks like this.

```
int circleX;
int circleY;
int circleSize;

void setup() {
  size(800, 600);
}

void draw() {
  background(0);
  showCircle();
}

void showCircle() {
  circleX = width/2;
  circleY = height/2;
  circleSize = 100;

  fill(255, 0, 0);
  circle(circleX, circleY, circleSize);
}

void mouseReleased() {
  if (dist(mouseX, mouseY, circleX, circleY) < circleSize/2) {
    println("hit");
  } else {
    println("miss");
  }
}
```

Challenges

1. Change the circle to a target.
2. Make different messages for each level of the target, like “bulls eye”, “nice shot”, “hit”, and miss. (You will need a few if/else if/else statements for this.)
3. Can you make a version of this game that uses squares instead of circles? (To do this you will need to check if the mouse’s position against all four sides of the square. Google Processing square collision detection, for help. You are looking for rectangle / point collisions.)