Sprite Lab (2 Days)

Description

In this two-day lesson, students will be introduced to the p5.play library and sprites.

On Day 1: Through a code-along, they will learn how to include the p5.play library, then create and modify their own sprites. During individual practice, students will create sprites with selected attributes.

On Day 2: Students will explore how sprites move and interact. Through a code-along students will learn how to control how sprites interact when they collide or overlap. During individual practice challenges, students will create and modify sprites, then determine how the sprites interact.

Objectives

- I can define what a sprite is.
- I can use the p5.play library to create sprites.
- I can set location, size, color, type, and motion attributes for my sprites.
- I can use gravity on my sprites.
- I can create dynamic and static colliders.
- I can control what two sprites do when they collide.
- I can control what two sprites do when they overlap.

Focus Question

How can we make and use sprites in p5.play?

Day 1: Brain-Starter + Hook (5 min)

Prompt:
"Think about some of your favorite video games, books, movies, etc.
Who is the main player or character?"

Turn && Talk: who is the main player or character in your favorite video game, book, or movie?

We're going to be using the p5.play library for this unit. ALL of the players in p5.play are represented with sprites.

Vocabulary:
sprite == characters, items, or anything else that moves above a background.

Let's make some sprites!

Day 1: Code-Along with Interspersed Practice (40 min)

Starter Code | Post-Code-Along Code | Challenges complete example

Using p5.play

Just like we need to include the p5 library to code with p5, we need to include the p5.play library to work with p5.play.

Let's look at how we're including the p5.play library for our use in the <head> of index.html

```
    <!-- NEEDED FOR p5 Play -->
    <!-- planck -->
    <script src="https://p5play.org/v3/planck.min.js"></script>
    <!-- Version 3 of p5.play -->
    <script src="https://p5play.org/v3/p5.play.js"></script>
    <!-- END: NEEDED FOR p5 Play -->
```

Starter code for this unit will always include these lines, but if you'd like to make additional p5.play sketches, make sure to include these lines.

Creating a sprite

Now that we have the library set, let's make a sprite. (New code highlighted in yellow.)

```
// Variables go here
let player;

function setup() {
  createCanvas(600, 600);
  player = new Sprite();  // creating a Sprite

function draw() {
  background(220);


  // helper code, feel free to comment out
      textSize(15);
      fill(255);
      stroke(0);
      strokeWeight(2);
      text("x: " + int(mouseX) + " y: " + int(mouseY), 30, 30);
 }
```

Note that we are creating the sprite in `setup()`. What happens if we create the sprite in `draw()` instead?

(Demonstrate for students and ask them to describe what happens and why: it will repeatedly create Sprites on top of each other, spilling out all over the screen. This happens because `draw()` is a loop.)

Walk students through determining the default settings for a sprite and comment this in your code.

- What stays the same each time we rerun the code? (shape: square, location: center of sketch, size: 50 pixels)
- What changes? (random color)

Modifying a Sprite

Walk students through how to do the following - in `setup()`:

- set x and y locations of a sprite

```
player.x = 100; // set x-value of center of Sprite
player.y = 120; // set y-value of center of Sprite
```

- make a circle sprite by setting diameter

```
player.diameter = 80; // set diameter of Sprite to make a circle
```

- comment out diameter and set width and height of a sprite to make a rectangle

```
player.w = 150; // set width of Sprite
player.h = 75; // set height of Sprite
```

- rotate a sprite (static)

```
player.rotation = 45; // rotate Sprite 45 degrees around center of Sprite
```

- set the color of a sprite

```
player.color = color(255, 255, 0); // set color of Sprite using rgb values
```

Teacher note: See https://p5play.org/learn/sprite.html?page=0 for more. (Wait to show students, as you'll cover movement soon!)

⭐ Student Practice #1 (slide for students)

Create two new sprites in the upper-right corner of your sketch: one red circle and one blue rectangle.

Creating Sprites Quicker with Constructors

Walk students through how to use the Sprite constructor to make:

- the rectangle you created earlier

```
player = new Sprite(100, 120, 150, 75);
// creates new Sprite with x: 100, y: 120, w: 200, h: 75
```

- a circle (what happens if we only include 3 numbers!)

```
player = new Sprite(100, 120, 150);
// creates new Sprite with x: 100, y: 120, diameter: 150
```

- a polygon (can use: triangle, square, pentagon, hexagon, septagon, octagon, enneagon, decagon, hendecagon, or dodecagon)

```
player = new Sprite(100, 120, 150, "triangle");
```

```
// creates new Sprite with x: 100, y: 120, sidelength: 150, as a triangle
```

Teacher note: See https://p5play.org/learn/sprite.html?page=4 for more about rectangle & circle constructor, then https://p5play.org/learn/sprite.html?page=8 about polygons.

⭐ Student Practice #2 (slide for students)

Rewrite your code for your rectangle and circle to use the Sprite constructor. If time, create a new polygon sprite.

Making Sprites Move by Themselves

Walk students through how to do the following in `draw()`, commenting out work as you go and taking care to not have sprites interact much!

- move in the x (or y direction - ask students how they might get the triangle to move up/down, left/right

```
player.vel.x = -5; // sets x-velocity; will move left at speed 5
```

- move back and forth

```
player.vel.y = cos(frame); // sets y-velocity; will move up and down with trig motion (cosine)
```

- set rotation speed

```
player.rotationSpeed = 1; // rotates around center at set speed
```

- set direction and speed:

```
player.direction = 45; // sets direction of Sprite
player.speed = cos(frame); // sets speed, will move back and forth in set direction
```

Teacher note: See https://p5play.org/learn/sprite.html

Day 1: Challenges (10 min)

Post-Code-Along Code | Challenges complete example

Write code in the same sketch as the code-along to:

- Make your blue rectangle move straight down in your sketch.
- Make your red circle move towards the bottom left corner of the sketch.
- Create two polygon sprites at the bottom left and right corners of your sketch. Make them rotate in opposite directions.
- Stretch goal: check out https://tinyurl.com/day1Stretch and make a sprite that moves using your keyboard.

Note: students will likely notice interactions between sprites! Note this and tell them we'll be working on this tomorrow.

Day 1: Closing (5 min)

On a sticky note, write your name and answer one of the following:

- What is something you learned today?
- What is something you made a connection to?
- What is something that is standing in the way of your learning?

Day 2: Brain-Starter + Hook (5 min)

Review:

- When making a sprite. Does it get created in the draw or setup function?
- What are some of the built-in properties of a sprite?

After think-time, have students "vote" for draw vs setup, then share out built-in properties.

Creating and modifying sprites has been fun, but let's get into the power of p5.play: how the sprites interact!

First, we'll need to talk about one other property that sprites have: their collider!

A sprite's collider is used to detect collisions with other sprites. (You may have noticed this yesterday when your sprites ran into each other!)

By default, sprites have a 'dynamic' physics collider that allows the sprite to move freely and be affected by gravity.

The other collider types are 'static' and 'none'. Sprites that have a 'static' collider can't be moved by other sprites and aren't affected by gravity. Setting a sprite's collider type to 'none' removes its collider from the physics simulation.

Day 2: Code-Along (25 min)

Starter code | Post-Code-Along Code

Review

Review (with student input) declaring variables:

```
// Variables here:
let player;
let ball;
let floor;
```

and work in `setup()`:

- How do we create a sprite? (Call this player.)

```
    player = new Sprite();  // square sprite as player
```

- How do we create a circular sprite? (Call this ball and make it near the top of the sketch, 50-100 pixels in diameter.)

```
ball = new Sprite(100, 120, 80); // circle sprite
```

- How do we create a rectangular sprite? (Call this floor and make it a long thin one near the bottom of the sketch.)

```
floor = new Sprite(width/2, 500, width, 10); // rectangular sprite near bottom of sketch
```

Introducing Gravity

We used different attributes to make sprites move yesterday. Since p5.play is a game engine and uses a physics simulator, we can also use gravity!

Walk students through using gravity:

- How do we set gravity? Let's make it simulate Earth's gravity!

```
world.gravity.y = 10; // simulate Earth's gravity
```

- What if we wanted gravity to work a little differently? ("Pull" upwards, left, or right? Simulate the moon or another planet? Moon: 1.625, see https://phys.org/news/2016-01-strong-gravity-planets.html for more!)
- However, what if I want my floor to stay put?
    - Set floor's collider to static

```
floor.collider = "static"; // floor can't be moved by sprites or affected by gravity
```

    - Set collider to static in floor's constructor (comment out work above)

```
floor = new Sprite(width/2, 500, width, 10, "static"); // rectangular sprite near bottom of sketch, floor can't be moved by sprites or affected by gravity
```

Teacher note: See https://p5play.org/learn/world.html for more.

Collisions

Now that we've got gravity, how can we control what happens when sprites collide? Let's use `collides()`!

If the player collides with the floor, let's make the player and the floor change:

```
// if the player collides with the floor
if(player.collides(floor)) {
    player.w = 100; // change the player's width
    floor.rotation = 15; // rotate the floor by 15 degrees
}
```

We can change any attributes about the player and the floor.

Teacher note: See https://p5play.org/learn/sprite.html?page=5 for more. Note: leaving collided() and colliding() for students to explore later.

Overlap

What if we don't want collisions? What if we want to gobble up shapes?

Let's create a small circle sprite that is a static collider in the path of our ball. Have students describe how to declare the variable:

```
let dot;
```

then create the sprite in `setup()`:

```
dot = new Sprite(100, 200, 10, "static");
```

The ball will just stop at the dot right now.

But we can use `overlaps()` to gobble up shapes!

```
if (ball.overlaps(dot)) {
   dot.remove();
}
```

Teacher note: See https://p5play.org/learn/sprite.html?page=6 for more.

Day 2: Challenges (25 min)

Starter code | Completed examples: Mild | Medium | Spicy

Mild

- Set world gravity to simulate Earth's gravity.
- Create a static floor sprite that takes up the bottom part of your sketch.
- Create a dynamic sprite that will collide with the floor sprite. When this sprite and the floor collide, change 1 attribute of either sprite.
- Create a static sprite that is between your dynamic sprite and your floor sprite. When the dynamic sprite overlaps the static sprite, change one attribute of either sprite.

Medium

- Set world gravity to simulate Earth's gravity.
- Create a dynamic circle sprite that will fall to collide with a static rectangle sprite. Make the sprites change 2 attributes each when they collide.

- Create a dynamic polygon sprite that will fall to overlap with a static circle sprite. Make the static circle sprite change or disappear when the polygon overlaps with it.

Spicy

Recreate challenge: https://01Sprite-LabPt3.kmaschm.repl.co (link in starter code)

Day 2: Closing (5 min)

Submit your challenge work in the post on Google classroom.