README.md

9. Shape Collisions

All of the code from this guide is sourced from Jeffrey Thompson's awesome online book which you should absolutely use for any collision you need. I've pulled out the most common cases to highlight here. The examples are in Java, and I'm working on converting them all over to p5.js.

DISCLAIMER: You are allowed to copy and use any of the code in this guide and on Thompson's website without it being considered plagiarism.

9.1 Point Inside Rectangle

This is straight-forward: a point inside of a rectangle is between all of its lines. So if we can prove a point is between the top and bottom sides (above the bottom, below the top) and between the left and ride sides (right of the left side, left of the right side), we're good.

```
// POINT/RECTANGLE         point   rectangle
function isPointInsideRect(px, py, rx, ry, rw, rh) {

  // is the point inside the rectangle's bounds?
  if (px >= rx &&        // right of the left edge AND
      px <= rx + rw &&   // left of the right edge AND
      py >= ry &&        // below the top AND
      py <= ry + rh) {   // above the bottom
        return true;
  }
  return false;
}
```

9.2 Point Inside Circle

A circle is, by definition, all of the points within a certain distance (radius) of the circle's center. Using that definition, we can test if a point is inside a circle by measuring the distance from our point to the circle and checking if that distance is less-than-or-equal-to the radius of the circle.

```
// POINT/CIRCLE             point   circle
function isPointInsideCircle(px, py, cx, cy, r) {

  // get distance between the point and circle's center
  var distance = dist(px, py, cx, cy);

  // if the distance is less than the circle's
  // radius the point is inside!
  if (distance <= r) {
    return true;
  }
  return false;
}
```

9.3 Circle Touching Circle

Seeing if two circles are touching is very similar to seeing if a point is inside a circle. Here, we're measuring the distance between the center of both circles and seeing if their radii overlap (distance is less than the sum of the radii). This would result in shared points, by definition.

```
// CIRCLE/CIRCLE               circle 1       circle 2
function isCircleTouchingCircle(c1x, c1y, c1r, c2x, c2y, c2r) {

  // get distance between the circle's centers
  var distance = dist(c1x, c1y, c2x, c2y);

  // if the distance is less than the sum of the circle's
  // radii, the circles are touching!
  if (distance <= c1r + c2r) {
    return true;
  }
  return false;
}
```

9.4 Rectangle Touching Rectangle

We have similar checks to a point, but their description is a little more... complicated:

```
Is the RIGHT edge of r1 to the RIGHT of the LEFT edge of r2?
Is the LEFT edge of r1 to the LEFT of the RIGHT edge of r2?
Is the BOTTOM edge of r1 BELOW the TOP edge of r2?
Is the TOP edge of r1 ABOVE the BOTTOM edge of r2?
```

Yikes.

```
// RECTANGLE/RECTANGLE      rectangle 1         rectangle 2
function isRectTouchingRect(r1x, r1y, r1w, r1h, r2x, r2y, r2w, r2h) {

  // are the sides of one rectangle touching the other?

  if (r1x + r1w >= r2x &&    // r1 right edge past r2 left AND
      r1x <= r2x + r2w &&    // r1 left edge past r2 right AND
      r1y + r1h >= r2y &&    // r1 top edge past r2 bottom AND
      r1y <= r2y + r2h) {    // r1 bottom edge past r2 top
        return true;
  }
  return false;
}
```