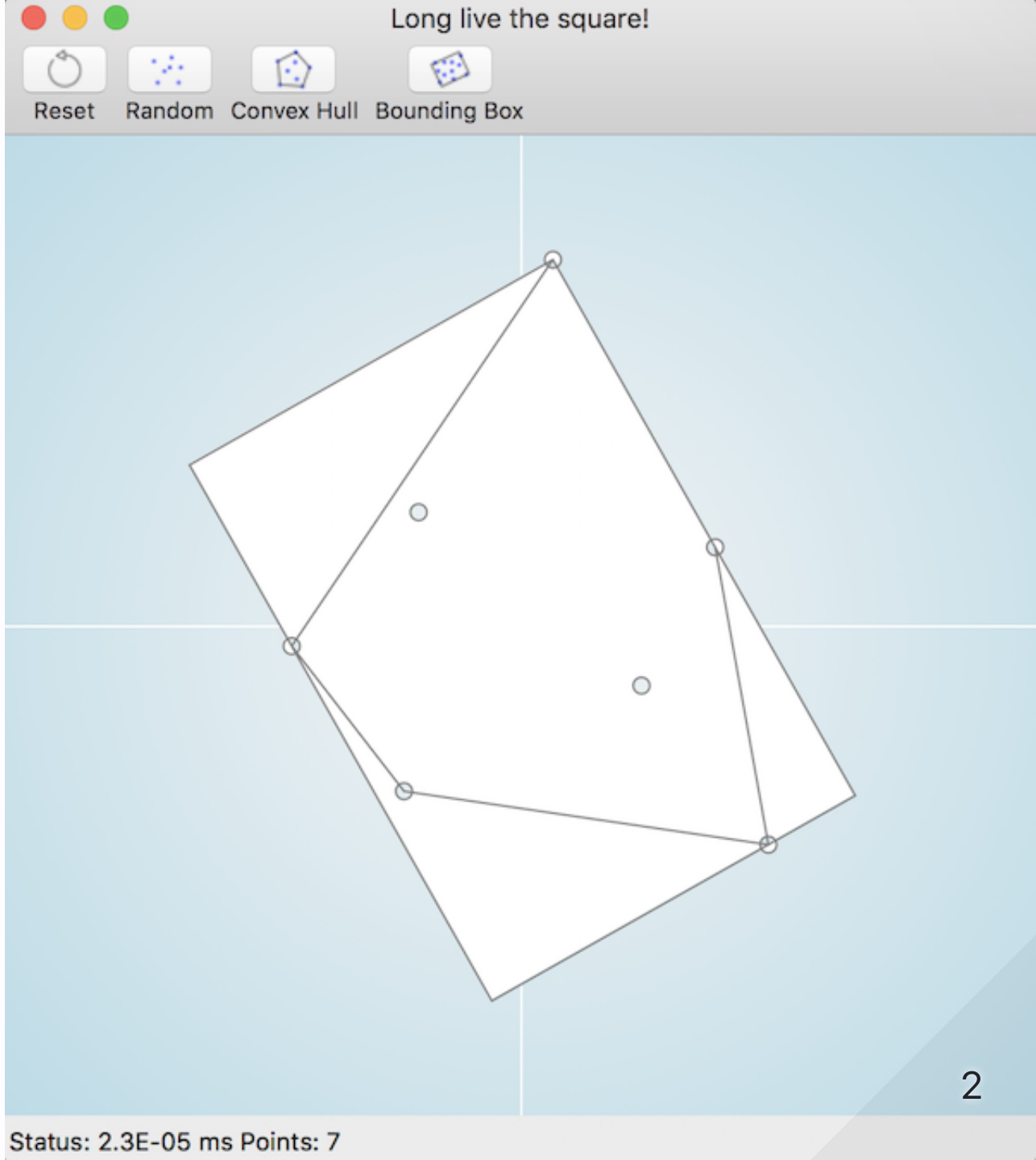
Collision Detection

Did we hit something? 🚗

Why?

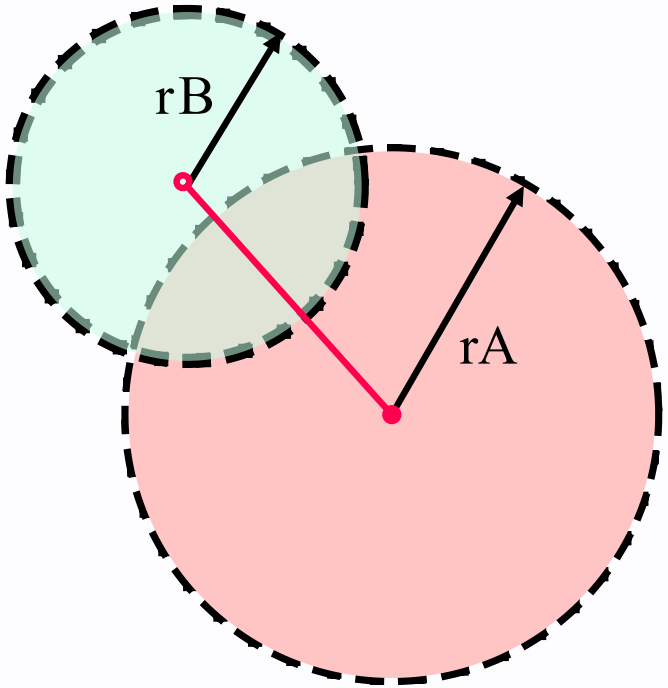
- Player / Obstacle
- Mouse / Button
- Trigger
- Computational geometry



Distance between point A and point B.

$$\sqrt{(x_B - x_A)^2 + (y_B - y_A)^2}$$

```
// helper method
dist(x1, y1, x2, y2)
```



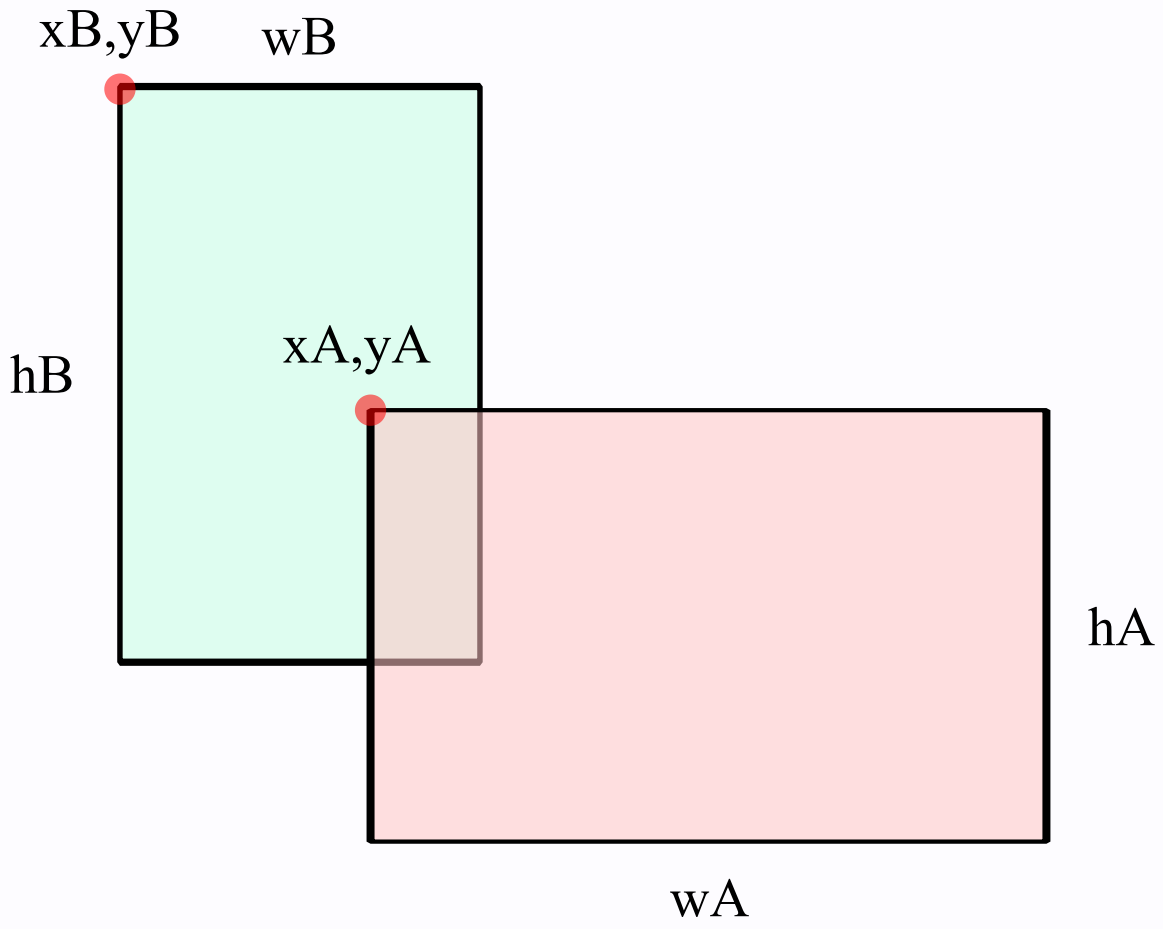
Circle Intersection

When do these two circles intersect?

- $dist(c_a, c_b) < r_a + r_b$

Circle Intersection

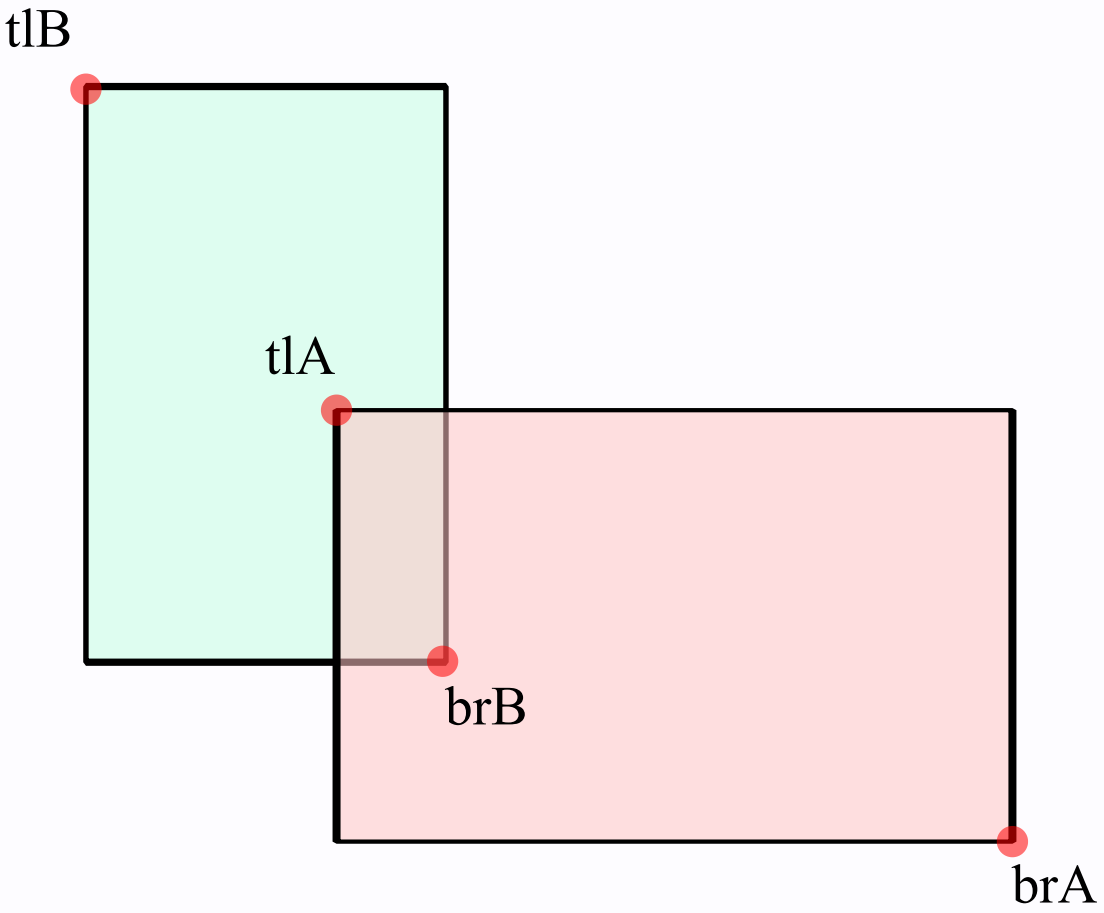
```
boolean checkCircle(
  float cxa, float cya, float ra,
  float cxb, float cyb, float rb) {

  return dist(cxa, cya, cxb, cyb) < ra + rb;

}
```



Box Intersection

When do these two boxes intersect?

- Calculate Top Left and Bottom Left positions



Box Intersection

Only if not...

One box is on left side of other

$tl_B x >= tl_A x$ or $tl_A x >= tl_B x$

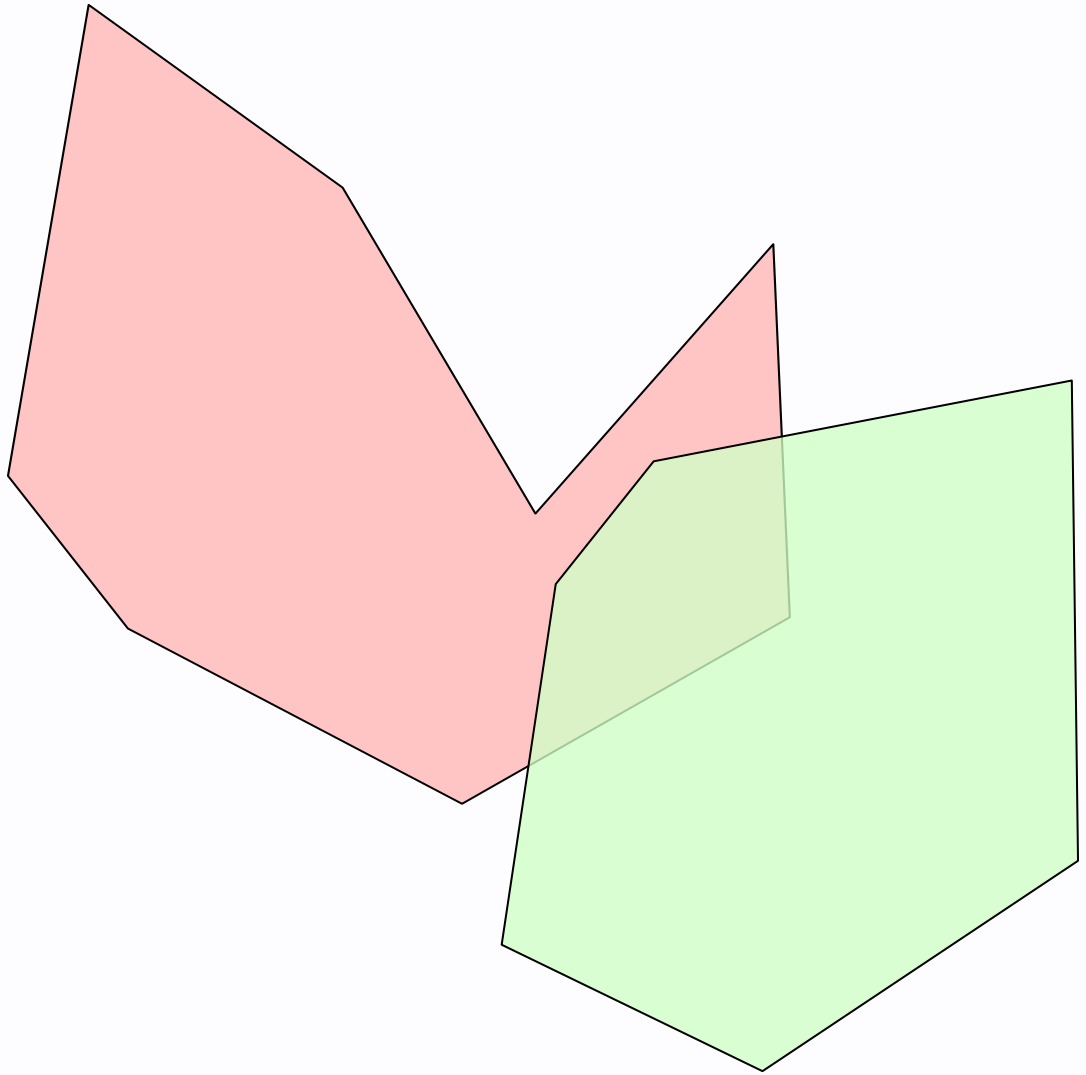
or

One box is above other

$tl_B y >= tl_A y$ or $tl_A y >= tl_B y$

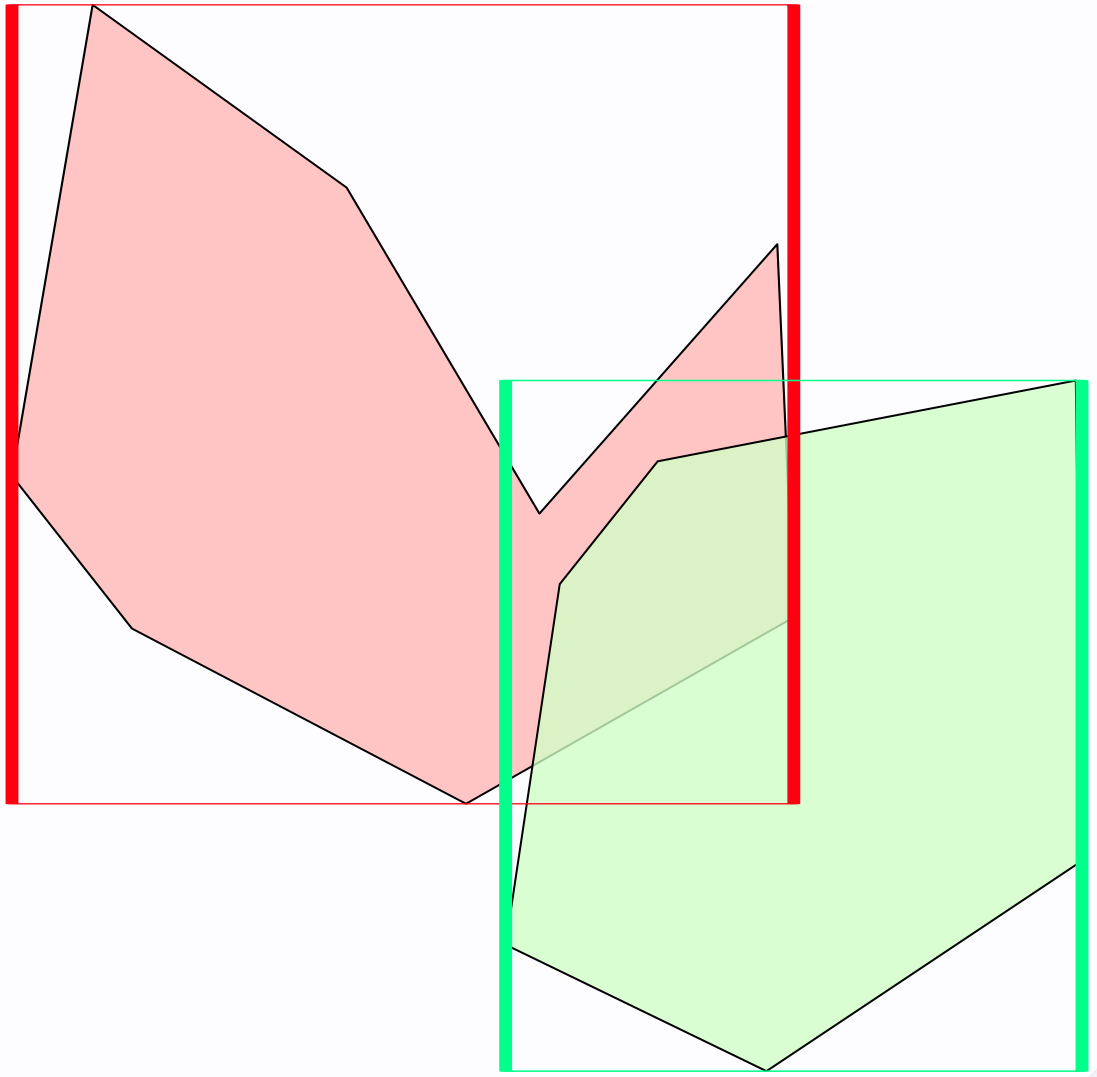
```
boolean checkRectangle(int r1x, int r1y, int r1w, int r1h,
                       int r2x, int r2y, int r2w, int r2h) {
  // store the locations of each rectangles outer borders
  int top1 = r1y-r1h/2;
  int bottom1 = r1y+r1h/2;
  int right1 = r1x+r1w/2;
  int left1 = r1x-r1w/2;
  int top2 = r2y-r2h/2;
  int bottom2 = r2y+r2h/2;
  int right2 = r2x+r2w/2;
  int left2 = r2x-r2w/2;

  if (top1>bottom2 || bottom1<top2 || right1<left2 || left1>right2) {
    return false;
  } else {
    return true;
  }
}
```

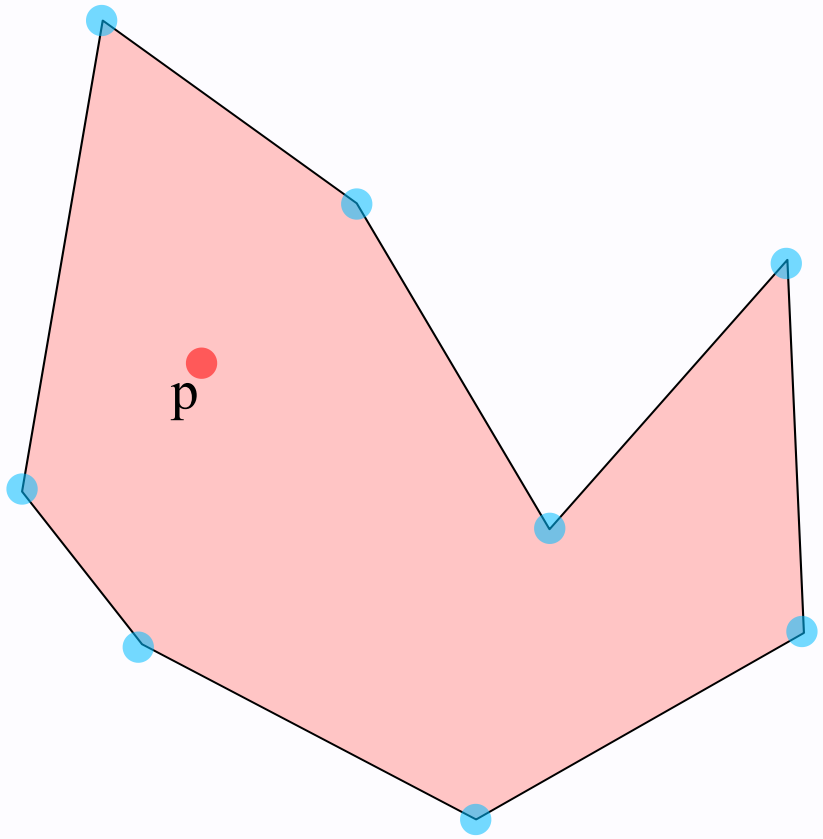
More Complex Shapes?



More Complex Shapes?

Create a bounding box or circle!



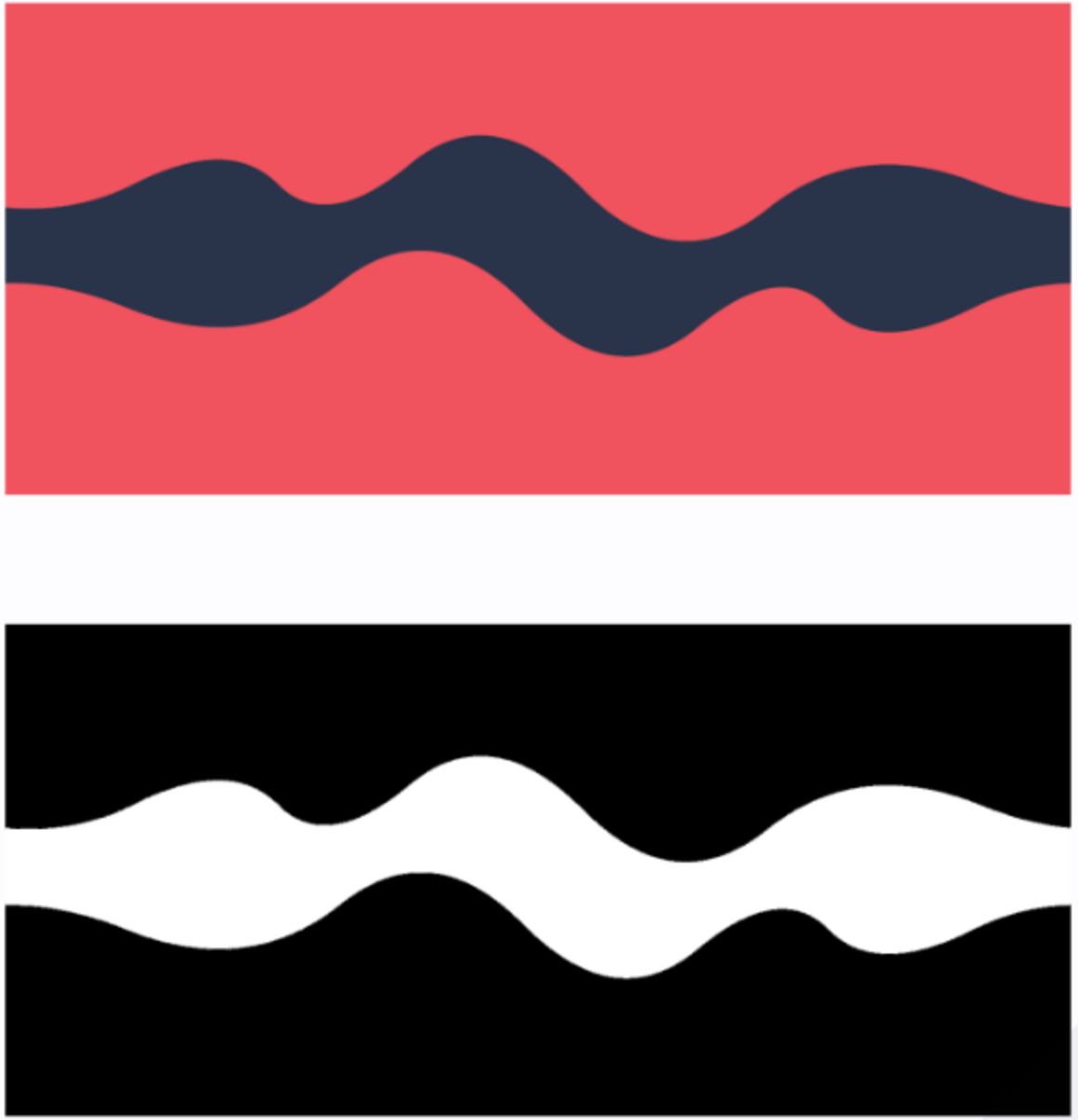


Polygon & Point Intersection

```
// W. Randolph Franklin
boolean pnpoly(
  PVector[] vertices,
  float testx, float testy) {
    ...
  }
```

Bitmap Collision Systems

- Read pixel color to determine if hit or miss.



Task Game Obstacles (30min)

Create an obstacle class which spawns on the right side of the screen and is able to collide with your player object. If this happenns, the player loses the game.

Task (30min)

Create a very simple **planetary system** in which the individual planets & moons circle around the sun. Use the matrix operations `translate()` & `rotate()`, and the transformation stack with `pushMatrix()` & `popMatrix()`.

Questions?