**Challenge**:

Create a simulation with a square containing a inscribed circle. If points are randomly chosen throughout the square, some will fall inside the circle. It turns out that the ratio of the number of points inside the circle to the total number of points generated is 1/4th of a very interesting number. Can you find the interesting number? And can you determine why that number is appearing in the simulation?



**Things you’ll need to know**:

* How to use playfields and stamping objects
* How to use variables

**Things to think about:**

* How can you randomize the location of points and leave a marker behind?
* How can you tell if the point is inside the circle versus just the square?

**How to do it:**

* See below!

**Extensions**:

* What would happen if you used a rectangle with an inscribed ellipse?

Geometric Probability Lesson

1. In the world, create three number variables: **inCircle**, **total**, and **fourTimesQuotient**. The first two can be integers, but the last variable should have as many decimal places as possible.
2. Drag a playfield to the world and name it **square**. Drag an ellipse to the playfield and name it **circle**. Drag another ellipse to the playfield and name it **point**.
3. For the square set the width and length to be some large number (say 400). For the point, set the width and length to be a very small number (say 3). You don’t need to change the circle’s width and length (yet). Make sure all three have different colors.
4. Create a World script named **randomPoint**. It should first randomize the position of the **point** to be somewhere in the playfield. The formula I used for the x-coordinate was: random(**square** width + 1) - 1. Do something analogous for the y-coordinate. Next, tell the **point** to show. Now, test if the **point** is overlapping with the **circle** - if so, the World’s **inCircle** variable should increment by 1. After the test, increase the World’s **total** variable by 1. Assign the World’s **fourTimesQuotient** variable to be 4 \* inCircle / total. Finally, have the **point** stamp and then hide itself.
5. Create a World script named **startSim**. It should tell the **randomPoint** script to start.
6. Create a World script named **stopSim**. It should tell the **randomPoint** script to stop.
7. Create a World script named **resetSim**. Set the **circle’**s width to be the same as the **square**’s width. Also set the **circle**’s length to be the same as the **square**’s length. Next, set the **circle**’s x to be the same as half the **square**’s width. Also set the **circle**’s y to be the same as half the **square**’s length. Set the World’s **inCircle**, **total**, and **fourTimesQuotient** variables to 0. Finally, tell the **square** and the **circle** to erase all pen trails.
8. Now, make buttons for the **startSim**, **stopSim**, and **resetSim** button... and you should be done!