Using Mathematica/Wolfram-Cloud to play with Bertrand's Paradox
https://www.youtube.com/watch?v=f2FOGLhQX_Q

http://www1.lasalle.edu/~blum/c152wks/Bertrand.pdf

Based on Bertrand's Paradox (with 3blue1brown) - Numberphile
https://www.youtube.com/watch?v=mZBwsm6B280

Version 1: Answer of 1/3
Version 2: Answer of 1/4
Version 3: Answer of 1/2

My "Version 4" simulation was producing a result about 0.6

In[1]:=
```
NumSamples = 10 000;
GTCount = 0;
MyPoints = {};
MyChords1 = {};
MyChords2 = {};
For[i = 1, i ≤ NumSamples, i++,
MyAngle = RandomVariate[UniformDistribution[{0, 2 Pi}]];
MyRadius = Sqrt[RandomVariate[UniformDistribution[{0, 1}]]]; (* Note the sqrt*)
MyAngle2 = RandomVariate[UniformDistribution[{0, 2 Pi}]];
root1 = -MyRadius * Cos[MyAngle2] + Sqrt[1 + MyRadius ^ 2 * ((Cos[MyAngle2]) ^ 2 - 1)];
root2 = -MyRadius * Cos[MyAngle2] - Sqrt[1 + MyRadius ^ 2 * ((Cos[MyAngle2]) ^ 2 - 1)];
AppendTo[MyPoints, Point[{MyRadius * Cos[MyAngle], MyRadius * Sin[MyAngle]}]];
x1 = (MyRadius + root1 * Cos[MyAngle2]) * Cos[MyAngle] - root1 * Sin[MyAngle2] * Sin[MyAngle];
y1 = (MyRadius + root1 * Cos[MyAngle2]) * Sin[MyAngle] + root1 * Sin[MyAngle2] * Cos[MyAngle];
x2 = (MyRadius + root2 * Cos[MyAngle2]) * Cos[MyAngle] - root2 * Sin[MyAngle2] * Sin[MyAngle];
y2 = (MyRadius + root2 * Cos[MyAngle2]) * Sin[MyAngle] + root2 * Sin[MyAngle2] * Cos[MyAngle];
r = Sqrt[(x2 - x1) ^ 2 + (y2 - y1) ^ 2];
If[r > Sqrt[3],
GTCount++;
AppendTo[MyChords1, Line[{{x1, y1}, {x2, y2}}]],
AppendTo[MyChords2, Line[{{x1, y1}, {x2, y2}}]]
];
]; Print[N[GTCount / NumSamples]];
```

```
0.6078
```

In[7]:=
```
Clear[r]
```

In[8]:=
```
Integrate[2 / Pi r, {r, 0, 1}, {theta, 0, Pi}]
```

Out[8]=
```
1
```

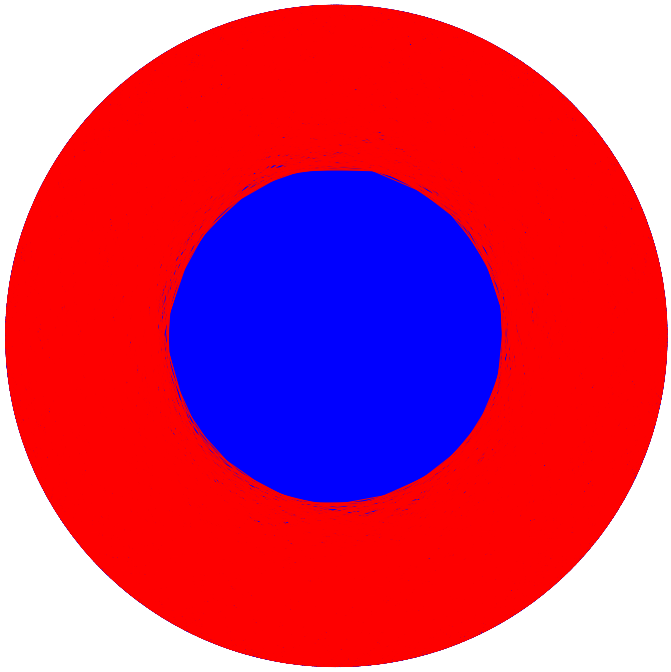
In[9]:=
```
ans = Integrate[2 / Pi r HeavisideTheta[1 / 2 - r * Sin[theta]], {r, 0, 1}, {theta, 0, Pi}]
```

Out[9]=
$$\frac{1}{3} + \frac{\sqrt{3}}{2\,\pi}$$

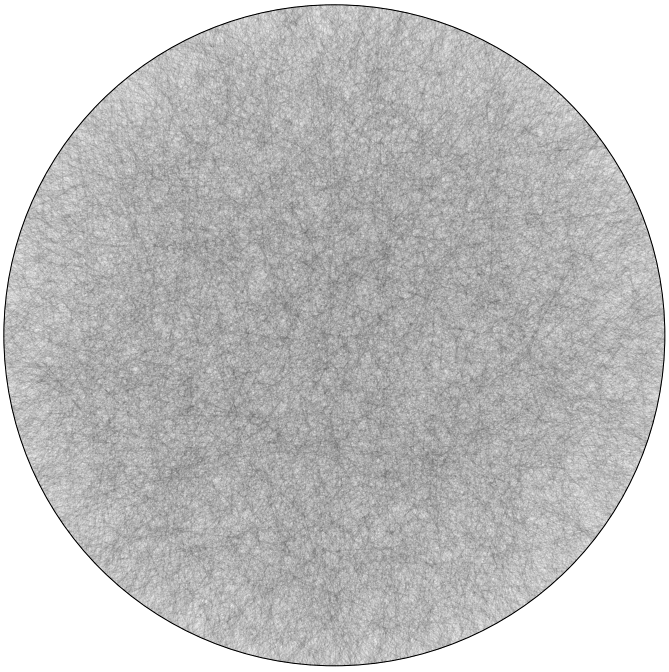
In[10]:=
```
N[ans]
```

Out[10]=
```
0.608998
```

In[11]:=
```
Show[Graphics[Circle[{0, 0}, 1]], Graphics[{Blue, MyChords1}], Graphics[{Red, MyChords2}]]
```

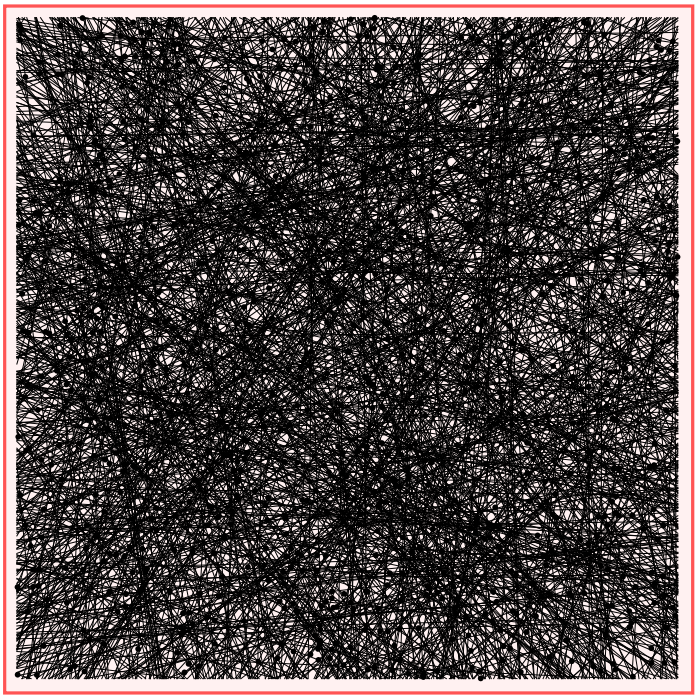


In[12]:=
```
Show[Graphics[Circle[{0, 0}, 1]],
 Graphics[{Opacity[0.025], MyChords1}], Graphics[{Opacity[0.025], MyChords2}]]
```



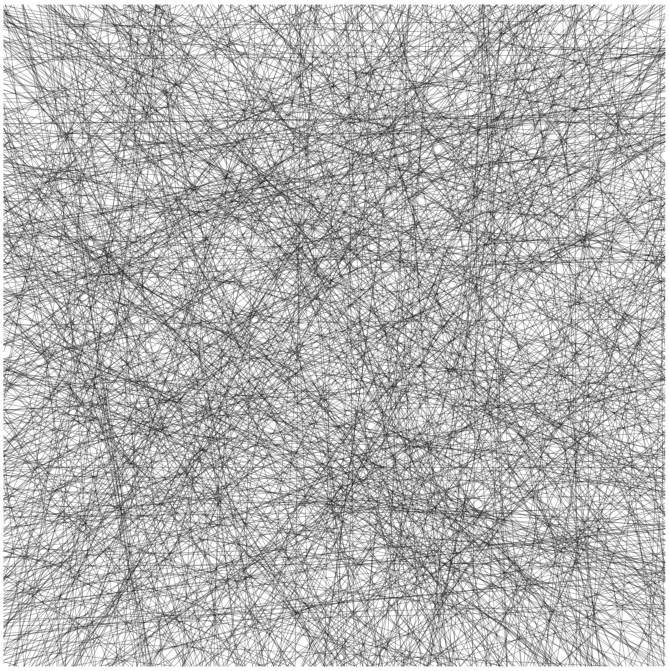
In[13]:=
```
NumSamples = 1000;
MyPoints = {};
MyChords = {};
For[i = 1, i ≤ NumSamples, i++,
(* Pick a point in the square that bounds the unit circle *)
X = RandomVariate[UniformDistribution[{-1, 1}]];
Y = RandomVariate[UniformDistribution[{-1, 1}]];
AppendTo[MyPoints, Point[{X, Y}]];

MyAngle = RandomVariate[UniformDistribution[{0, Pi}]];

MyPoints2 = {};
xa = 1; ya = Y + (1 - X) * Tan[MyAngle]; If[-1 ≤ ya ≤ 1, AppendTo[MyPoints2, {xa, ya}],];
xb = -1; yb = Y + (-1 - X) * Tan[MyAngle]; If[-1 ≤ yb ≤ 1, AppendTo[MyPoints2, {xb, yb}],];
yc = 1; xc = X + (1 - Y) * Cot[MyAngle]; If[-1 ≤ xc ≤ 1, AppendTo[MyPoints2, {xc, yc}],];
yd = -1; xd = X + (-1 - Y) * Cot[MyAngle]; If[-1 ≤ xd ≤ 1, AppendTo[MyPoints2, {xd, yd}],];
(*Print[MyPoints2]; *)
AppendTo[MyChords, Line[MyPoints2]];

](* end for loop over samples *);
Show[Graphics[MyPoints],
 Graphics[{Blue, Opacity[0.25], Square[{-1, -1}, {1, 1}]}], Graphics[MyChords]]
```

Out[17]=



In[18]:=
```
Graphics[{Opacity[0.25], MyChords}]
```

Out[18]=



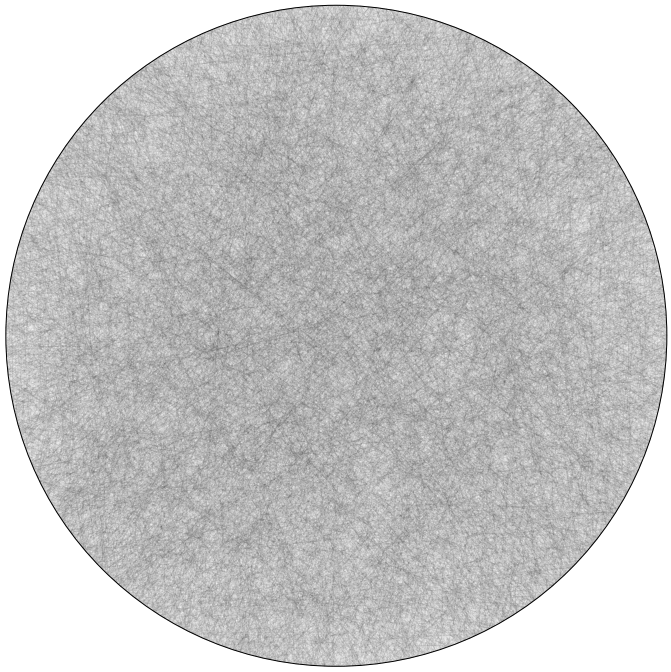
In[19]:=
```
NumSamples = 10 000;
MyPoints = {};
MyChords1 = {};
MyChords2 = {};
ChordCount = 0;
BigChordCount = 0;
For[i = 1, i ≤ NumSamples, i++,
(* Pick a point in the square that bounds the unit circle *)
X = RandomVariate[UniformDistribution[{-1, 1}]];
Y = RandomVariate[UniformDistribution[{-1, 1}]];
AppendTo[MyPoints, Point[{X, Y}]];

MyAngle = RandomVariate[UniformDistribution[{0, Pi}]];

a = 1 + (Tan[MyAngle]) ^ 2;
b = 2 * (X + Tan[MyAngle] * Y);
c = X ^ 2 + Y ^ 2 - 1;
If[b ^ 2 - 4 * a * c > 0,
ChordCount++;
root1 = (-b + Sqrt[b ^ 2 - 4 * a * c]) / 2 / a;
x1 = X + root1; y1 = Y + root1 * Tan[MyAngle];
root2 = (-b - Sqrt[b ^ 2 - 4 * a * c]) / 2 / a;
x2 = X + root2; y2 = Y + root2 * Tan[MyAngle];

r = Sqrt[(x1 - x2) ^ 2 + (y1 - y2) ^ 2];
If[r > Sqrt[3],
BigChordCount++; AppendTo[MyChords1, Line[{{x1, y1}, {x2, y2}}]];,
AppendTo[MyChords2, Line[{{x1, y1}, {x2, y2}}]] ;;
]
,(* nothing if no solution*)]

](* end for loop over samples *);
Print[BigChordCount / ChordCount];
Print[N[BigChordCount / ChordCount]];
```

$$\frac{2707}{4705}$$

0.575345

In[26]:=
```
Show[Graphics[{Opacity[0.025], MyChords1}],
 Graphics[{Opacity[0.025], MyChords2}], Graphics[Circle[{0, 0}, 1]]]
```

Out[26]=



In[49]:=
```
NumSamples = 10 000;
MyPoints = {};
MyChords1 = {};
MyChords2 = {};
ChordCount = 0;
BigChordCount = 0;
For[i = 1, i ≤ NumSamples, i++,
(* PICK A LARGER SQUARE TO CHOOSE POINTS *)
X = RandomVariate[UniformDistribution[{-3, 3}]];
Y = RandomVariate[UniformDistribution[{-3, 3}]];
AppendTo[MyPoints, Point[{X, Y}]];

MyAngle = RandomVariate[UniformDistribution[{0,  Pi}]];

a = 1 + (Tan[MyAngle]) ^ 2;
b = 2 * (X + Tan[MyAngle] * Y);
c = X ^ 2 + Y ^ 2 - 1;
If[b ^ 2 - 4 * a * c > 0,
ChordCount++;
root1 = (-b + Sqrt[b ^ 2 - 4 * a * c]) / 2 / a;
x1 = X + root1; y1 = Y + root1 * Tan[MyAngle];
root2 = (-b - Sqrt[b ^ 2 - 4 * a * c]) / 2 / a;
x2 = X + root2; y2 = Y + root2 * Tan[MyAngle];

r = Sqrt[(x1 - x2) ^ 2 + (y1 - y2) ^ 2];
If[r > Sqrt[3],
BigChordCount++; AppendTo[MyChords1, Line[{{x1, y1}, {x2, y2}}]];,
AppendTo[MyChords2, Line[{{x1, y1}, {x2, y2}}]] ;;
]
,(* nothing if no solution*)]

](* end for loop over samples *);
Print[BigChordCount / ChordCount];
Print[N[BigChordCount / ChordCount]];
```

$$\frac{1908}{3803}$$

0.501709

In[34]:=
```
Show[Graphics[{Opacity[0.025], MyChords1}], Graphics[
 {Opacity[0.025], MyChords2}], Graphics[Circle[{0, 0}, 1]]]
```

Out[34]=