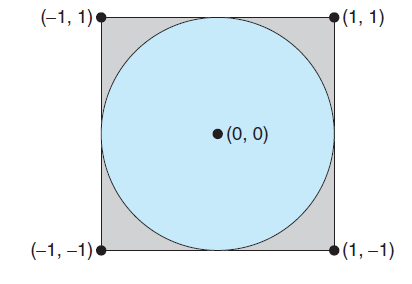
**Programming Assignment 3**

# Objective

The objective of this assignment is to understand the use of threads and how threads can be used in multithreaded programs.

# Assignment: Estimating Pi (π) using Monte Carlo Simulations

An interesting way of calculating π is to use a technique known as Monte Carlo, which involves randomization. This technique works as follows: Suppose you have a circle inscribed within a square, as shown in Figure below



(Assume that the radius of this circle is 1.) First, generate a series of random points as simple (x, y) coordinates. These points must fall within the Cartesian coordinates that bound the square. Of the total number of random points that are generated, some will occur within the circle. Next, estimate π by performing the following calculation:

Write a multithreaded version of this algorithm in C/C++ that creates a separate thread to generate a number of random points. The thread will count the number of points that occur within the circle and store that result in a global variable. When this thread has exited, the parent thread will calculate and output the estimated value of π. It is worth experimenting with the number of random points generated. As a general rule, the greater the number of points, the closer the approximation to π.

A code segment for generating random numbers, as well as determining if the random (x, y) point occurs within the circle is provided below:

/\* Generates a double precision random number \*/

double random\_double()

{

return random() / ((double)RAND\_MAX + 1);

}

/\* Check for points inside circle

for (i = 0; i < npoints; i++) {

/\* generate random numbers between -1.0 and +1.0 (exclusive) \*/

x = random\_double() \* 2.0 - 1.0;

y = random\_double() \* 2.0 - 1.0;

if (sqrt(x\*x + y\*y) < 1.0 )

++hit\_count;

}

Name the program *mcarlo.c* or *mcarlo.cpp* and provide the number of points of the command line:

* *./mcarlo <number of points>*

Use at least two worker threads to calculate the total number of points as well and the number of points inside the circle.

**Error Handling**

Perform the necessary error checking to ensure the correct number of command-line parameters. Verify the number of points is a valid number and during test vary significantly the number of points to see the difference in the accuracy of the estimation of π.

# Grading

The program will be graded on the basic functionality, error handling and how well the implementation description was followed. Be sure to name your program ***mcarlo.c (mcarlo.cpp)*** (no extra characters, capitals) Note that documentation and style are worth 10% of the assignment's grade!

# Submission

The program source code should be posted to Blackboard.