# 159.735 Assignment 5

You have been commissioned to help design a new quantum warp drive engine. Since this engine is designed to drive spaceships through hyperspace, you will need to deal with spatial dimensions greater than 3.

Given an $N$-dimensional sphere of radius, $r$, centred at the origin of an $N$-dimensional coordinate system, first complete a sequential function to count the number of integer coordinate points inside the sphere. Here are some examples to allow you to check your work:

1. A 1-dimensional sphere of radius 25.5 has 51 integer coordinate points: $\{(\pm 25, \pm 24, \pm 23, \cdots, \pm 1, 0)\}$
2. A 2-dimensional sphere of radius 2.05 has 13 integer coordinate points within the sphere: $\{(0,0), (-1,0), (-2,0), (1,0), (2,0), (-1,-1), (-1,1), (1,-1), (1,1), (0,-2), (0,-1), (0,1), (0,2)\}$
3. A 3-dimensional sphere of radius 1.5 has 19 integer coordinate points inside the sphere: $\{(-1,0,0), (-1,0,1), (-1,0,-1), (-1,-1,0), (-1,1,0), (1,0,0), (1,0,1), (1,0,-1), (1,-1,0), (1,1,0), (0,0,0), (0,1,0), (0,1,1), (0,1,-1), (0,-1,0), (0,-1,1), (0,-1,-1), (0,0,-1), (0,0,1)\}$

A small start up program will be provided, where you are asked to complete the sequential function `count_inside`. This function will be invoked for a number of different values of $r$ and $N$. The pseudo-code for the sequential implementation goes like:

```
for (n=0; n < ntrials; ++n) {
  radius = selected_at_random()
  ndim = select_at_random()
  // Invoke function to count inside integer coordinates
  count = count_inside(radius, ndim)
}
```

**Having completed the sequential program, then develop an implementation to run on your GPU.**

You will need to decide what your kernel function will do and how to organize your threads. It is also likely you will encounter some limitations on your graphics card that will affect performance and the values for $r$ and $N$ that will actually work.

## Submission

Please submit your C or C++ source code together with a brief report. In your report you should address at least the following. Marks will be awarded for discussions that demonstrate clear understanding on GPU architectures.

- State the graphics card you used together with any relevant specs.
- Briefly describe your algorithm for counting the number of integer coordinates within an $N$-sphere
- Provide a description on how you parallelized the above problem.
- Present appropriate performance data, and comment on your result.

Due date: TBD

This assignment is worth 20% of your final grade

## Appendix: Coding Hints

To convert a decimal integer number, **`num`**, into a number in base **`B`**, you can use a number of possible algorithms, for example:

```
idx = 0;
while (num != 0) {
  rem = num % B;
  num = num / B;
  digit[idx] = rem;
  ++idx;
}
```

Examples: $1208_{10} = 10111000_2$, $1208_{10} = 1122202_3$, $1208_{10} = 2270_8$, $\cdots$.

You can use **`cudaMalloc()`** to create a single data word, not just arrays, on the GPU device. Also, you can use **`cudaMemset()`** to initialize a device value and you can use **`cudaMemcpy()`** to copy single data words from the device to the host.

You will probably encounter a situation where a number of threads can modify a single value. Clearly, this would have to be done atomically. The CUDA library provides a (overloaded) function **`atomicAdd()`** that makes this easy.