Q: Why did the pedestrian cross the road?

A: To get to the other side, of course.

This world is full of pedestrians trying to get to the other side of the road at a 4-way intersection. Your job is to make their actions look as human as possible. In this world there are 4 corners, crosswalks connecting those corners, and traffic signals for each of the crosswalks. Each actor (pedestrian) has a target corner that it is trying to reach. When it gets there it will receive a new target corner. This world should run forever with the actors constantly trying to get across the road.

The default_actor that is included acts in the most basic way possible. It walks straight towards its target corner, at a constant speed, not paying attention to traffic controls, not paying attention to other actors. You can certainly do better.

You have control over the heading and the speed of the actor. The goal is to create realistic behavior while achieving the goal of reaching the target corner. Some of the problems that you may want to tackle:

- Collision avoidance. Humans don't like running in to each other.
- Traffic signals. Humans don't like being run over by cars.
- Queuing. Humans generally gather in a bunch while waiting for the light to change.
- …. Humans have lots of other behaviors that you could try to model

Rules of the world:

- Time always moves forward (even while you are processing)
- You can change the heading and speed of the actor
- You can sense the world at the same rate that you act
- All actors are circles with a diameter of 1 unit
- The traffic signals give you 20 seconds to cross. At 10 seconds the signal will go from green (GO) to yellow (BLINK).
- The pedestrians will start in a random spot at one of the corners with a random target corner

Your actor will run against a simulation of the world implemented in a library you will use within your actor. Simulation time starts once you initialize the library. When you call step to advance the simulation state you all of the actors will be asked to act.

An example actor (default_actor) is provided and can be built by calling 'make all'. While C++ is preferred you can write in the language of your choice as long as you interface with the provided library. Don’t spend time optimizing for our specific hardware, but to give you an idea we will be testing on a machine with a Core i7 processor (4 physical cores) and at least 16GB of memory.

There is one initialization parameter of the simulation you can control, the number of actors in the space. If you find the problem too easy increase this parameter. If

it is too hard reduce this parameter. Let us know the value you think showcases your program the best. We're more concerned with you writing a good implementation than it scaling to huge numbers of actors;

The code for the graphical display is provided. If you want to add debugging information, make it prettier, etc… go ahead.

Your code should be well designed and well commented.

You should provide a document explaining your approach. In addition to explaining how your solution works, also include anything you tried that didn't end up in your final solution, alternative solutions you considered, and how you might expand on your solution given more time. There are several aspects to creating realistic human behavior. You will likely not be able to tackle all of them. Tell us what you did attempt and what other behaviors exist that you didn't work on.

The provided code will build and link on Ubuntu 14.04 64 bit. If you do not have an installation of this distribution available, it can easily be downloaded and run from a bootable USB drive or installed in a virtual machine. You can also download pre-installed VMs from a variety of sources. If you are using a freshly downloaded 14.04 image, you'll need to install (with sudo apt-get install) at least the following libraries: g++, libgl1-mesa-dev, libglu1-mesa-dev, freeglut3-dev, and libqt4-dev. Feel free to use any additional libraries that are free and easy to install on Ubuntu 14.04, but remember that one of the goals of this exercise is to evaluate ***your*** coding ability and practices, so make sure that can be judged from your submission. Example: using a matrix math library is fine, but using a motion planning library is probably a bad idea.

Deliverables:

- Source code
- Instructions to build and execute your source code
    - Include instructions for installing libraries your solution requires
- Explanatory document