1 Implementing the Peceive, Decide, Act process

Here we discuss the techniques and algorithms used to realise the Perceive, Decide, Act process previously discussed [Reference to Research].

1.1 Perception

Agent perception is achieved using the following simple algorithm:

```
input  : The set of goals in the environment: goals
output: A set of goals visible to the user at the given instant in time:
        visibleGoals

for each goal g in goals do
    if g is in line of sight of agent then
        add g to visibleGoals;
    end
end
```

Algorithm 1: Agent Perception Algorithm

1.2 Decision Making

In the Research Chapter [Reference] the three classes of human behaviour considered in the scope of this project were defined. In practice, only herding behaviour implemented as part of the Decide step; queueing and competetive behaviour must be handled asynchronously using collision avoidance techniques [Reference to Dan's work].
To decide on a new action, an agent must select one of the visible goals that were found in the Perceive step (if any). Otherwise it must continue on its current path. The decision making process is realised using an extendable algorithm, which sequentially considers sets of different classes of goal according to their priority. Exits have the highest priority.

The final algorithm only makes decisions regarding exits, for reasons we will discuss later in the report. However it is easy to extend this process to include other goal types by adding further conditionals following the pattern laid out below.

```
input  : Person person, ExitGoal[ ] exits /*add further exit types here as
         arrays */
output: Target Goal for agent to move toward
if no of exits > 0 then
    ExitGoal currentExit = exits[0];
    if person is stressed then
        for each exit e in exits do
            if number of people queuing at e > no. of people queueing at
            targetExit then
                targetExit = e;
            end
        end
        return targetExit;
    else
        Vector3f position = person.location;
        for each exit e in exits do
            if distance to e < distance to targetExit then
                targetExit = e;
            end
        end
        return targetExit;
    end
else
    return null;
end
```

1.3 Act

This step's representation in the BehaviouralModel class is trivial since the Decide step already returns a target goal. It is left in as a place for performing any calculations which should be performed before returning the target goal to the agent.
Upon receiving a new target goal, an agent should perform the following:

- Use a PersonNavmeshRoutePlanner to calculate a route to this goal
- Set the returned MotionPath as the current MotionPath for the agent
- Begin moving down this path