1 Route Planning

Planning an agent's route from its current point to a given point is acheived by a collaboration between a Person and PersonNavmeshRoutePlanner instance. Routes are stored using the jMonkeyEngine class MotionPath. This contains a set of waypoints and methods to animate the agent's movement between these waypoints. The Mmaximum distance between each waypoint is roughly constant and can be set as a parameter to the following procedure (it is not constant when the agent must turn at a point closer to its current position that the maximum distance between points). The distance used is an important decision in the implementation. If it is too large the accuracy of the animation tends to degrade. If it is too low then a large number of waypoints will be stored needlessly.After extensive experimentation 0.5 was found to be an acceptable value for the maximum distance between waypoints. In practice this provided a balance between quality and efficiency.
A route is calculated in two stages:

1. A path is calculated on the navigation mesh using a modified A* algorithm to traverse the mesh like a graph. This returns a small set of points. These points illustrate the lines of motion an agent must take to reach their goal. This is performed in the constructor for a PersonNavmeshRoutePlanner.

2. Using these points as guidance, the path is 'fleshed out' by moving along the path and placing MotionPath waypoints no further apart than the defined maximum distance. This terminates with a waypoint being placed on the goal location the agent must reach.

Note that a fresh PersonNavmeshRoutePlanner instance must be instantiated to calculate a route. The relationship between the Person and PersonNavmeshRoutePlanner instances is expressed in

2 Implementing the Perceive, Decide, Act Process

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

2.1 Perception

Agent perception is achieved using the following simple algorithm:

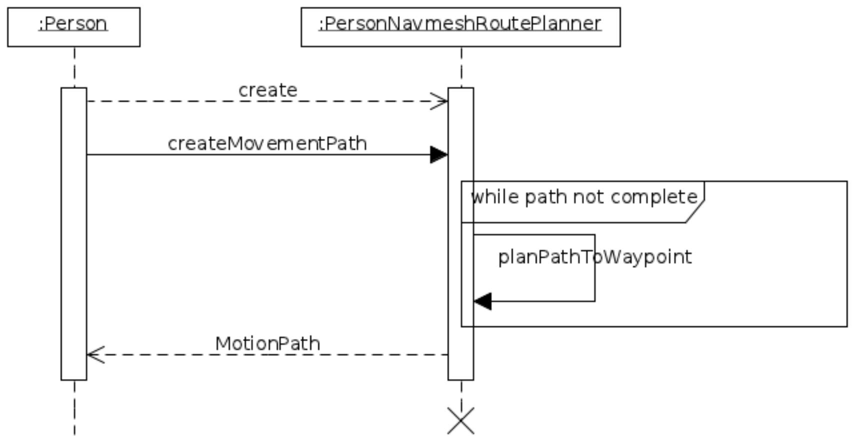

Figure 1: Generation of an agent route as a MotionPath

```
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

2.2 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
```

2.3 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