1 Minimum Deliverables

1.1 Prepare existing codebase to be expanded upon and fix existing issues

- Convert source files to adhere to current coding conventions
- Make use of .NET 4.5 features where applicable
- Overhaul world generation system to require less repetition of code
- Test and fix ray tracing of a moving collision hull

1.2 Implement new pathfinding system and threat avoidance

- Place path vertices at significant positions within each block of the city for use when pathing between two nearby positions
- Implement an abstracted higher level path finding system for finding general paths between two remote locations, using the intersections of roads in the city as vertices
- Allocate a limited amount of time for each agent to calculate paths per frame, and store the partial result to be continued next frame if required
- Add geometry *hint* nodes to mark features of the world that can be exploited when agents are avoiding threats, for example hints for doorways and room corners
- Rewrite agent threat avoidance to take into account geometry hints
- Test the new path finding system by instructing agents to path to randomly selected locations in the world

1.3 Analyse system scalability

- Evaluate frame time and memory allocated for a range of world sizes, from trivial to the largest possible
- For each world size, record the above metrics for varying numbers of agents, from none to the limit that increases frame time to be larger than some target, with path finding disabled
- With some configuration of the above two parameters, experiment with the amount of time allocated for agents to find paths to analyse frame times
- Experiment with limiting the range of agents path finding node exploration to currently or previously visible tiles, and then for currently occupied or previously occupied blocks, and compare performance and memory used

2 Intermediate Objectives

2.1 Improve and extend simulation environment

- Add mesh entity rendering component for entities with 3D models
- Implement barricade entities that block off a tile that may be destroyed if attacked enough
- Add at least one decorative entity that will produce a resource (barricade material) when destroyed
- Improve world generation to produce several building types with different layouts and that contain varying amounts of resource dropping decorative entities

2.2 Implement subsumptive agent AI architecture

- Build a wide variety of AI components to achieve different tasks or promote certain behaviours such as threat avoidance or exploration
- Organise the AI components into layers of abstraction, where more abstracted layers may choose between the decisions made by the preceding less abstracted one, or override them
- Compare survival rates and realism of different weightings for the subsumptive behaviour system, for example a configuration designed for flocking, and another where agents are solitary

2.3 Improve performance of path finding

- Implement path caching to improve path finding performance
- Limit the capacity of the path cache, and remove infrequently used paths if this is exceeded
- Investigate the resulting cache hit and fault rates for cached path discarding methods

3 Advanced Objectives

3.1 Expand user interaction with the simulation

- Add barricade material storage designations, and the ability for a player to place them
- Include method for a player to order a specified group of agents to travel to a given location
- Allow players to mark decorative entities that produce barricade material to be dismantled and scavenged from
- Implement barricade building designations that may be placed by a player
- Produce a method for players to toggle individual or groups of agents from being offensive or defensive
- Create new AI components to complete the new user designated tasks

3.2 Implement a BDI architecture for comparison

- Construct an alternative agent AI architecture using a beliefs- desires-intentions model to perform the same tasks as the existing subsumptive one
- Attempt to build a rudimentary planning system on top of the new BDI oriented architecture to improve agent performance
- Compare the two architectures in terms of memory usage, frame times, behaviour and survival rates for a variety of different initial simulation states
- Assess the efficiency at which each architecture completes user assigned tasks in similar circumstances