Modern Computer Games
COMP 521, Winter 2015

Assignment 3

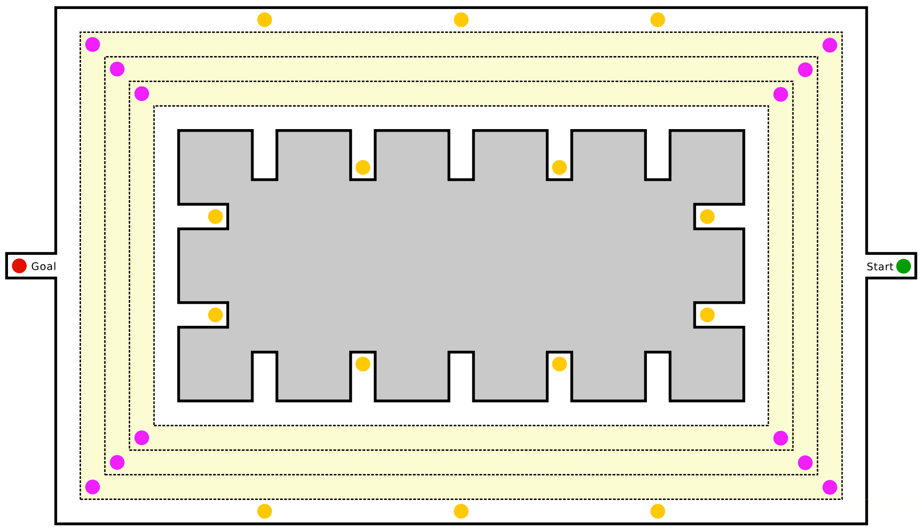
Due date: Thursday, March 26, 2015, by 6:00pm

Note: Late assignments will only be accepted with prior **written** permission of the instructor. You must **explain** all answers and **show all work** to get full marks! Please make sure your code is in a professional style: **well-commented,** properly structured, and appropriate symbol names. Marks will be very generously deducted if not!

Description

This assignment again requires you develop a solution in the Unity game development environment. Please continue to use the version you used for previous assignments. Note that you need to implement your AIs entirely on your own—please do not use the Behaviour Designer or other tools.

1. You first need an environment with zombies. Build a game level as shown by the top-down view shown below. It consists of a large space with a central obstacle, numerous alcoves, and 3 "tracks" (or lanes) that ring the obstacle. The goal of this is to observe the behaviour within, so you will need an unattached camera view that can be freely moved around irrespective of obstacles, and relatively even lighting. **10**



nb: You do not need to reproduce this plan perfectly, but you do need to ensure the same general proportions and features.

2. The zombies are intended to walk along single tracks of the multi-track course, with their behaviour entirely defined and controlled through a *finite state machine* (FSM) that describes suitable states and makes use of transitions based on (combinations of) time, probabilities, and game events. Zombies are of different types, so you will need several different FSMs. **15**

Classic These zombies all walk at the same speed (call it v). They never change lanes.

Shambler These zombies walk at speed $v/2$, frequently and arbitrarily changing lanes (to adjacent lanes).

Modern These zombies walk at speed $2v$, and are able to switch to adjacent lanes to overtake slower zombies ahead of them.

Zombie with a phone These zombies walk and interact with a smartphone at the same time. Thus they frequently stop for several seconds, erratically vary their speed (between $v/2$ and $2v$), occasionally changing direction, and sometimes change lanes.

The latter two zombies are difficult zombies, while the former two are easy ones.

Zombies occupy space: if lanes are 1 unit wide, then zombies can be represented by a unit square or unit circle. Movement is meant to be continuous (not discrete), although lane changes and direction changes may be instantaneous/discrete.

In addition, all zombies exhibit *collision avoidance* behaviour—slowing down, stopping, or changing lanes (to whatever extent their type allows) in order to guarantee not colliding with other zombies.

Include a clear, labelled drawing of each FSM as a separate document. You may use present your FSMs in a moduler way to reduce redundancy, as long as how they connect is made clear.

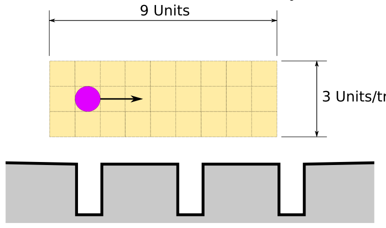
3. Now, create a visually distinct representation for each kind of zombie, and populate your level with n zombies, randomly choosing either classic or shambler types, where n is defined by a parameter. Initially, place your zombies randomly (but non-overlapping) in the various lanes of the course, all facing the same direction. Do not initialize zombies on the right-vertical part of the course (near the start point). **15**

   Implement your FSMs for each zombie type behaviour. Note that you must implement your own AI/FSM (do not use a predefined library or asset), although if you prefer you can hard-code the states and transition-checking code for your situation rather than implement a general FSM representation and interpreter.

   After initialization, at each lane-corner (purple dot) a zombie has a probability of instantaneously despawning, while simultaneously (re)spawning a new zombie of a random type at a different corner. This respawn probability, p, should be a parameter. The choice of zombie types induces another variable, r, the ratio of difficult to easy zombies.

4. Now, introduce the *survivor* into the simulation. This agent is spawned at the "start" dot shown on the right (green), and needs to eventually reach the "goal" dot shown on the left (red). Between, they need to visit each of the small (orange) dots distributed around the level. They occupy the same amount of space as a zombie, and move at speed $\approx 1.5v$. They can see any non-occluded zombie in a 360° FoV with infinite range. (nb: For this you will need to visually indicate which zombies are apparent to the survivor and which are not.) **15**

   A survivor must not be detected by any zombie, or the zombies will mass attack, and it is game over for them. Fortunately, zombies have limited awareness, with the ability to sense the survivor within one lane-width to either side of them, almost about 2 alcoves distance forward, but only a short distance behind them:



   The survivor is controlled entirely through a behaviour tree AI, such that they are able to complete their journey while also endeavouring to not be seen by any zombies. As with the zombie AIs, you should provide your own implementation, but may hard-code the tree structure rather than implement a generic behaviour tree interpreter if you wish.

   Verify your survivor can complete their task in the absence of any zombies, and multiply they time they take to do so by 3 in order to give them a maximum timeout for completion.

   Include a brief, textual explanation of your AI strategy for avoiding being seen, as well as a clear, labelled drawing of your behaviour tree.

5. Your survivor should certainly succeed in reaching the goal with only a few zombies. Now you need to make the game harder. Find 3 parameter settings that highlight (maximize) one of the parameters at the expense of the others so as to make it difficult, but still feasible for your AI. Your 3 parameter are n (total number of zombies), p (the probability of respawning), and r (ratio of difficult to easy zombies). **5**

   For each setting, your AI should be able to succeed approximately 50% of the time. Describe the exact settings and justify your choices.

What to hand in

Assignments must be submitted on the due date **before 6pm**. Submit your assignment to *MyCourses*. Note that clock accuracy varies, and late assignments will not be accepted without a medical note: **do not wait until the last minute**.

For the Unity questions, hand in an exported project containing all files needed in order to reconstruct and run your simulations. Note that Unity exports can be extremely large, and take non-trivial time to upload (another reason last-minute submission may not work out well).

For non-Unity questions, submit either an ASCII text document or a .pdf file *with all fonts embedded*. Do not submit .doc or .docx files. Images (plots or scans) are acceptable in all common graphic file formats.

This assignment is worth 15% of your final grade. **$\overline{60}$**