Modern Computer Games

COMP 521, Fall 2025

Assignment 3

Due date: Wednesday, November 5, 2025, by 23:59pm

Note: Late assignments will only be accepted with prior **written** permission of the instructor. 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

In this assignment you will explore dynamic pathfinding through an implemention within the Unity game development environment.

You must implement the pathfinding entirely yourself; **do not use any built-in or external implementations, assets, or tools for pathfinding**. You may, however, take full advantage of Unity's built-in physics or other APIs to detect, prevent, or react to collisions.

1. First build a 3D level as a large, bounded rectangular area. This area will need to be populated by 8-12 random obstacles, each either T-shaped or U-shaped, proportioned more or less as shown in the overhead view below. The size of obstacles can be fixed, but the number, position, and orientation of the obstacles should be different on every play-through. Obstacles should not overlap or be too close to the boundary or other obstacles—ensure the largest-area agent (see next question) can move between all obstacles. **5**

   The game level is also divided into 6 sub-areas (shown by the grey dividers), representing different terrain types that make walking in that area easier or harder. Assign to each area a random walking cost between 0.5 and 5.0, and use textures or colours to indicate the relative scale of walking cost.

   Position the camera and use relatively even lighting to allow easy visual observation of all parts of the level.

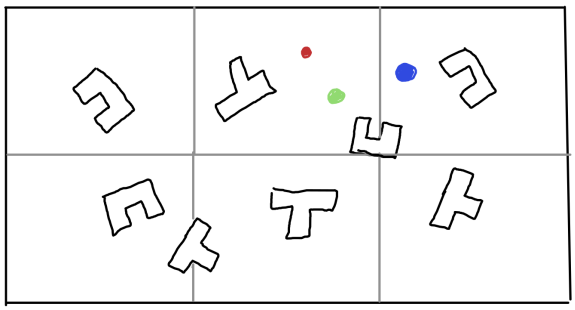

Figure 1: Sketch of the level design.

2. Define mobile agents that can move around within your simulation. An agent should have occupy a non-0 area, either small, medium, or large circles (shown as coloured dots in the sketch). Agents must never overlap with obstacles or the level boundary. Include an editor control to select the agent size, and spawn it at random, non-obstacle interior point in the level. A newly spawned agent should replace any existing agent in the simulation. **5**

   Once spawned an agent should attempt to move (via pathfinding, see next question) to random destination points within the level. Goals should be selected within the non-obstacle interior space and made visible in some fashion. For inspection, animate the agent motion, selecting an agent speed such that it would take them around 2s to get fully across the level length in a straight line if walking cost were 1.0 throughout. Once at their destination, the agent stops.

3. Design and implement a pathfinding system based on using a *reduced visibility graph* (RVG). Note that this means you will need to keep track of the abstract geometry of your level in order to generate the graph. **10**

   Your design should guarantee that an agent does not overlap with an obstacle or go outside the boundary, and your search should return an optimal path given the routes available in your RVG graph, taking into account the different movement costs of sub-areas.

4. Given the changing terrain cost, using an RVG will not necessarily compute truly optimal paths. Come up with a heuristic solution that improves over the naive RVG solution, but still includes the RVG graph. Include an editor control that lets you select between the naive RVG and your improved version. **5**

5. Provide *as a separate (.pdf) document* a concise explanation of how your approach deals with both the variable agent area, and varying sub-area costs. Compare your improved solution with your naive solution from question 3 in terms of path-length over multiple tests, giving data to show the average extent of improvement. **5**

What to hand in

You must submit your Unity source code, along with the document describing your approaches pathfinding solutions and performance data, and describing any missing features or limitations.
For the Unity questions, hand in a zip file of your project code containing all files needed in order to reconstruct and run your simulation. Please avoid excessive use of high-resolution assets, as it easily results in very large archives. Allow ample time to upload your assignment solution, and please follow the general Unity instructions given in MyCourses.

For the document(s) submit 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.

Assignments must be submitted on the due date **before midnight**. Submit your assignment to *MyCourses*. Note that upload times can be slow and clock accuracy varies: **do not wait until the last minute**.

This assignment is worth 15% of your final grade. **30**

Programming assessment

	Mastery	Proficient	Developing	Beginning
Correctness	The solution works correctly on all inputs and meets all specifications.	The solution meets most of the specifications; minor errors exist.	The solution is incorrect in many instances.	The solution does not run or is mostly incorrect.
Readability	Well organized according to course expectations and easy to follow without additional context.	Mostly organized according to course expectations and easy to follow for someone with context.	Readable only by someone who knows what it is supposed to be doing.	Poorly organized and very difficult to read.
Algorithm Design	The choice of algorithms, data structures, or implementation techniques is very appropriate to the problem.	The choice of algorithms, data structures, or implementation techniques is mostly appropriate to the problem.	The choice of algorithms, data structures, or implementation techniques is mostly inappropriate to the problem.	Fails to present a coherent algorithm or solution.
Documentation	The solution is well documented according to course expectations.	The solution is well documented according to course expectations.	The solution documentation lacks relevancy or disagrees with course expectations.	The solution lacks documentation.
Performance	The solution meets all performance expectations	The solution meets most performance expectations	The solution meets few performance expectations	The solution is not performant.