Modern Computer Games
COMP 521, Fall 2021

Assignment 3

Due date: Friday, November 12, 2021, by 6:00pm
Extension: no penalty if handed in by Monday, November 15, 2021 8:00am

Note: Late assignments (beyond the automatic extension) 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**.

1. You will first need a basic game environment. Build a 3D space as shown in the sketch below. The area is roughly rectangular, with an upper mezzanine level. It is divided into two sections, with the two sections of each level connected by 3 bridges (2 sloped) as shown. These bridges should be not quite wide enough to allow 2 characters to pass each other, so only one character can cross a given bridge at a time. **10**

   Position the camera and use relatively even lighting to allow easy visual observation of as much of the entire level as possible.

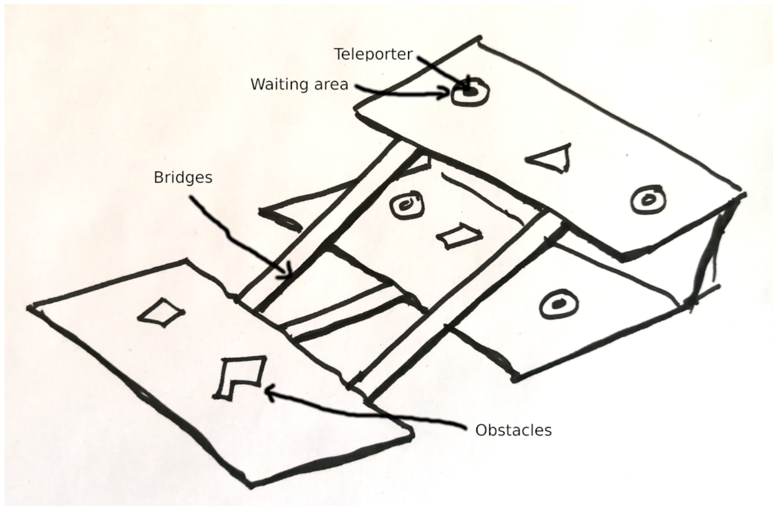

Figure 1: Sketch of the base design.

2. Connection between the lower and upper levels is also possible through 2 transporters. Each can transport up to 3 NPCs at a time. You do not need to model their appearance in detail, but can simulate the effect by teleporting characters from the lower to upper or vice versa. Define a perimeter/waiting area in front of the transporter in both the lower and upper parts. Each waiting area should easily hold at least 3 NPCs. Transporters then operate following a fixed schedule that cycles between moving NPCs from the lower level to the upper and then reversing direction: (1) up to 3 waiting NPCs are selected and removed from the lower (upper) waiting area; (2) transportation between levels takes 1s; (3) the NPCs being transported are re-spawned in the waiting area of the upper (lower) level as other NPCs are removed from the upper (lower) waiting area to repeat the process in the opposite direction. **5**

NPCs that exit the transporter should endeavour to leave the waiting area immediately, and NPCs not intending to use the transporter should not enter the waiting areas. NPCs should never overlap, and transporter capacity should never be exceeded.

Note that the NPCs moved by the transporter are selected from all NPCs in the waiting area other than the ones who just arrived from the other level. In case of congestion, an NPC that lingers in the waiting area may end up being transported again.

3. The walkable, area should have 10 small, randomly placed (different on each gameplay) and non-overlapping obstacles. Obstacle boundaries should be simple geometric shapes, but with some variation in either shape or orientation. Do not put obstacles on the bridges or waiting areas, and avoid blocking access to a bridge or waiting area. **5**

4. Populate the area with n NPCs, where n is a simulation (editor) parameter. NPCs should occupy a small but non-0 area (see bridge requirements above), and should never overlap with other NPCs or obstacles. NPCs are spawned at random, non-overlapping, locations anywhere in the non-obstacle game space, but not in the waiting areas. Each NPC follows the same rote behaviour, consisting of repeatedly doing the following: (1) choose a random destination (any non-obstacle location other than the waiting areas, and ignoring other NPCs) (2) make a plan to move to it and follow it, (3) pause for 200–1000ms once they arrive. Scale movement speed so NPC motion is moderately quick but easily trackable: it should take at least 200ms to traverse a sloped bridge. **5**

   Try to make individual agents visually distinguishable (e.g., different colours).

5. Implement two different pathfinding algorithms based on the algorithms discussed in class (other than Dijkstra). The choice of algorithm is up to you, but **you must develop your own implementations in their entirety**. All NPCs will use the same algorithm, but for testing (and debugging) you will need to easily switch between your implementations, so provide an editor control that allows for the choice. Your implementations should allow for NPCs to use both the bridges and the teleporters. **20**

   An NPC that cannot make progress along its current path should immediately attempt to re-path to the same destination as soon as it is blocked. If it cannot make any progress at all then it should pause and choose a new destination.

   Ensure that with just a few agents (e.g., $n = 4$) your agents are (almost) always able to efficiently get to their destinations.

   In a separate document briefly describe which algorithms you implemented and your general implementation strategy, especially in relation to how you accommodated the transporters.

   Up to 5 bonus points (limited to max 100% on the assignment) are possible for particularly advanced implementations.

6. Compare your two implementations. Keep track of the total number of pathings, the average time spent by the pathfinding algorithm itself, the number of re-pathings done, and the number of paths abandoned over a 2min simulation, averaged over a few simulations. Start with 4 agents and keep doubling the number until performance or behaviour becomes unacceptable. **10**

   In a separate document present your data (graph and/or tables) and briefly discuss characteristics of your data. How many agents can each of your algorithms effectively support?

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**.

Include all source code necessary to run your simulation, as well as the two documents describing your design and your

This assignment is worth 15% of your final grade. **$\overline{55}$**