Name: Sanad Masanant CCID: sanad Date: 9/24/2021

Instructions: Each question is worth the given amount of points, and the whole quiz is worth 9 points total. Answer each question to the best of your ability. Submit your quiz to eClass before Sunday at 11am. You may submit your quiz as a pdf, a docx, or as a zip file of images. **It is your responsibility to ensure the TAs and instructor can read your answers**, if you’re concerned about that please type your answers when possible.

This quiz is open everything except one another, meaning please do not collaborate with other students on the quiz. Cheating is unacceptable and will be reported.

**Multiple Choice (0.75 points each)**. Circle, bold, or mark the answer closest to the one you would give.

**Q1**) Which of the following describes the way most modern AAA games distribute computational power?

a. 50/50 split between graphics and AI b. Majority of computation to AI

c. Majority of computation to graphics d. None of the above

**Q2**) What is one benefit of path networks over nav meshes?

a. Nav Meshes are more restrictive b. Easier for designers to constrain AI behaviour

c. They have a much cheaper runtime cost d. All of the above

**Q3**) Nav Clues (the special generated path network nodes used in *Sunset Overdrive*) can be created based on which of the following?

a. The player’s position b. Objects in the world

c. Interesting locations d. All of the above

**Q4**) Assume we’re in an empty 8x8 grid with no obstacles, hazards, or other game entities, which of these three rankings represent the runtime computational cost from least to most of these pathing methods?

a. APSP < Greedy < A\* b. APSP < A\* < Greedy

c. Greedy <APSP < A\* d. All three would be roughly equivalent

**Q5 (2.5 points)**. Answer each part below to the best of your ability.

**Q5.A (0.5 points).** Consider the 8x3 (width by height) table below as a potential grid, where each cell with a letter in it is a potential grid cell. Mark between 4 and 6 of the cells (with letters in them) as obstructed. These five cells must be separated into at least two groups, as if they are covered by at least 2 obstacles. There must be at least one way to get from A to F.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | **B** | **C** |  |  | **D** | **E** | **F** |
| **H** | **I** | **J** | **K** |  | **L** | **M** | **N** |
| **O** | **P** | **Q** | **R** | **S** | **T** | **U** | **V** |

To mark the cells obstructed by obstacles you can fill them with a different colour. As in:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | **B** | **C** |  |  | **D** | **E** | **F** |
| **H** | **I** | **J** | **K** |  | **L** | **M** | **N** |
| **O** | **P** | **Q** | **R** | **S** | **T** | **U** | **V** |

You may instead draw the obstacles digitally or physically, As in:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **A** | **B** | **C** |  |  | **D** | **E** | **F** |
| **H** | **I** | **J** | **K** |  | **L** | **M** | **N** |
| **O** | **P** | **Q** | **R** | **S** | **T** | **U** | **V** |

You may instead list the cells obstructed by obstacles, As in:   
Obstructed: (H, I, O, P) and (V)

Do not use the same set of obstacles as in these examples or one that differs by **3 cells changes (obstructed/unobstructed) or fewer**. Do not cut off the path from A to F. You will receive 0 points for Q5 if you do not follow these instructions. There are hundreds of possible answers, so it shouldn’t be the case that you share an answer with anyone else.

**Q5.B. (1 point)**. Given a start location of A and a goal location of F, assume the Manhattan distance (abs(x2-x1)+abs(y2-y1)) as a heuristic. Give the sequence of current node values (as in the example on Monday) of the current node variable when running A\* on the environment you created in **Q5.A**. You do not have to list the open set, or the g, h, or f scores but can if you like. In the case where there are multiple possible next equally good current nodes you may choose any of them.

A

B

H

I

J

P

Q

R

S

T

L

D

E

F

**Q5.C. (0.5 points)**. Give the final path determined by your answer to **Q5.B**.

A🡪 B🡪H🡪I🡪J🡪P🡪Q🡪R🡪S🡪T🡪L🡪D🡪 E🡪F

**Q5.D. (0.5 point)**. Give a path smoothed version of the path from **Q5.C** using the path smoothing algorithm given in class (run only once). Assume that the pathing agent sits in the center of a grid cell and has a radius equal to 1/2 of a grid cell’s width (it takes up nearly the whole cell). Assume the empty centre area cannot be navigated or seen through.

A🡪P🡪Q🡪R🡪S🡪 E🡪F

**Q6 (1 point).** Given the environment you defined in **Q5.A.**, would it be best to make use of grids, path networks, or nav meshes (for the latter two, replacing the grid representation you’ve used thus far)? Give two reasons why your chosen spatial representation would be better for this environment (**0.5 points each**). Be as specific as you can (e.g. not “Nav Meshes are always better.”)

For my designed environment, I think nav meshes would be the better special representation. My first reason for choosing so is because while the end path is locked to a specific sequence (without smoothing S🡪T🡪L🡪D🡪 E🡪F🡪N🡪V), there is quite a lot of free space prior to node S so a nav mesh can allow one to traverse freely within that open area. My second reason that during runtime, it would be the cheapest given our current environment. As mentioned earlier, we have a set sequence of nodes we must visit due to obstacles being at nodes M & N so we would want to minimize cost prior to reaching our locked sequence and nav meshes are definitely cheaper than grids and in this case, are cheaper than using path networks .

**Q7 (1 point).** Given the below environment, give a valid final nav mesh with coverage of all open space that minimizes the total number of polygons in the mesh. Either draw the lines of the nav mesh (if you can be clear enough) or write them out (e.g. (A,B,F), (F, I, K) would define two triangles). Assume all lines that appear parallel are parallel and that all enclosed shapes are obstacles. There are multiple correct answers to this question.

Chart, scatter chart

Description automatically generated

**Q8 (2.5 points).** Answer the below parts to the best of your ability.

**Q8.A (0.5 points)**. Using the lists below create a type of NPC for a game.

**Choose one of the following:** Friendly, Antisocial, Lonely, Pack-based, Swarm

**Choose one of the following:** Intelligent, Mindless

**Choose one of the following:** Fast, Slow

**Choose one of the following**: Player companion, Enemy, Passive

For example, choosing the first in each list would get you a: “Friendly, Intelligent, Fast Player Companion”. Describe your character and invent any extra details you’d like. Do not use my same example.

Design: Antisocial, Intelligent, slow enemy.

**Q8.B (2 points)**. Describe what: (1) spatial representation (**0.5 points**), (2) path planning approach (**0.5 points**), and (3) steering/smooth /method(s) to modify given paths (**1 point**) you would use to communicate the character you generated in **Q8.A.** Fill in any extra details you would need to decide between different options here. Make sure to give brief (1-2 sentences) justifications for each answer.

Spatial Representation: Nav Meshes: Due to the nature of being antisocial, the character would rather be away from most people so a nav mesh which allows free movement would fit perfectly here. Others would be too robotic.

Path Planning Approach: As our enemy is slow yet intelligent, we will need the most efficient path planning for its AI as it would want to reach the player quicker. For that reason, I would choose A\*. Not only does A\* give us an efficient path to the goal, it could be assumed that the player would not be staying still and the area would be dynamic, A\* would definitely be the better options.

Steering/smooth /methods: Path smoothing would be needed due to the enemy’s intelligence yet slow movement to quickly get to the player. We would want to the enemy AI to look at where it’s going as it is not mindless, seek and purse the enemy as it is an enemy and its objective is to kill/defeat the player. We would want it to separate from other enemies due to it’s antisocial nature as well.

**EXTRA CREDIT (0.5 points)** Steering is a powerful tool for conveying the personality of AI entities in games (as in Question 8). However, when poorly designed, it can lead to entities that are caught in place as their steering functions cancel out to no movement. (EX.A) Give an example of a non-trivial reasonable steering/boids-like setup with at least 3 rules/functions that fire in order could lead to this scenario (**0.25 points**), and (EX.B) give an example with at least 3 entities caught in this scenario (**0.25 points**). For EX.A “appears reasonable” means that it cannot be too obvious that the three rules will cancel each other out.

For example:   
  
Rule 1: Go towards the closest entity 1 step

Rule 2: Go away from the closest entity 0.25 steps

Rule 3: Go away from the closest entity 0.75 steps

Would not be reasonable, but if it referenced more variables besides just “closest entity”…

You may assume entity locations are in two or three dimensions. You may make use of alignment if you choose but do not have to. You may invent any other details or variables you like, just state them clearly. You may write your rules in text or a pseudocode format of your choice. Hint: Make use of both other entities in your rules. **For even attempting this extra credit you will get 0.1 extra credit points**.

Rule 1: If player is in same y/x dimension, move 1 step within the x/y dimension towards the player respectively.

Rule 2: If other allied units are nearby, move one step in the same direction

Rule 3: If there is an allied unit near the next node, move back one step.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| E |  | E |  | E |
|  |  |  |  |  |
|  | E | P | E |  |
|  |  |  |  |  |
|  |  |  |  |  |