School of Engineering and Information Sciences

MIDDLESEX UNIVERSITY

EXAMINATION Solutions PAPER

2011/2012 Winter/Spring term

MODULE TITLE Advanced Topics in Games Development

MODULE NUMBER CMT 3325

MODULE LEADER'S NAME Chris Huyck

Time allowed: 3 hours

Total number of questions: 4 questions

Instructions to candidates: Answer all questions. Each question carries 25 marks.

Materials provided: The book Programming Game AI by Example (by Buckland) is allowed, along with one A4 page marked as notes on the top. Please submit that page with the exam (it will not be marked).

Equipment permitted: none

Total number of pages: 3

No books, papers or electronic device is to be brought into the examination room other than any specified above.

Candidates are warned that illegible scripts will not be marked

1. AI and State Spaces

(a) In the labs, we made an 8x8 maze. How many states can the agent be in? Describe your assumptions.

(6 marks)

Marking scheme:
2 points a reasonable number.
2 points for assumptions
2 points for correct answer given assumptions

Sample answer:

Assuming that the only thing that is relevant is the room that the agent is in, there are 64 states. In the crystalspace mazing games, the agent could also be positioned in a range of places in a room, and be facing in different directions. For that matter, other factors such as walls and the balls could be involved. Each of these increases the size of the space. Assuming 10 position per direction, and 10 possible orientations per dimension, the states go to 10^6*64.

(b) Describe the difference between an exhaustive search and one that is not exhaustive. Give an example of both types. (If you can't name an algorithm, feel free to describe it.) Why would you use an exhaustive search?

(9 marks)

Marking scheme:
2 points for correct answer to exhaustive
2 points for correct exhaustive search
2 points for correct heuristic search.
3 points for why

Sample answer:

An exhaustive search goes through all of the possible states, while heuristic searches only typically go through some of the states. Two common exhaustive search techniques are depth first and bread first. The simplest heuristic search is hill climbing, though beam search and most others are also heuristic. Exhaustive search is useful when the best possible answer is needed, or when the state space is small.

(c) Give an example of a game and when the use of state space search might be useful. Explain the game, state space, and method for searching it.

(10 marks)

Marking scheme:
2 points game description
4 points state space description.
4 points search method

Sample answer:

Let's try Mario Kart, and here we'll be developing an agent. I think a good way to divide the problem space is into a large game space, and a direct track space. Mario Kart is a racing game with up to 12 participants. Each races around different tracks at the same time and get points based on how they finish. They can gain special powers by running into items, and

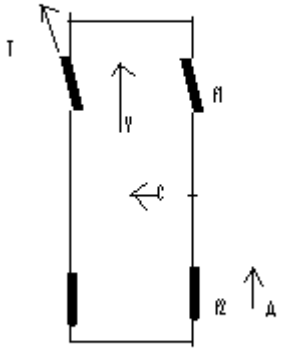
can slow others and speed themselves. Racers can play in teams or solo. The game space would be overall position, points desired, team mates etc. This would enable the agent to be more aggressive if he was behind, or behave differently if playing for a team. For that matter, in some circumstances, the agent could stop and catch the opponents coming around the track the next time. The large space could also include non-local context in the race. That is, Mario is way in front and Luigi is way behind, so they don't affect the local context. The local context would be the track features the agent could see and the position (and powers) of the other agents (including the user). I think the best way to search the large state would be a rule based mechanism to set goals. The goals could then be used to set search criterion at the lower level. At this level a best first search would be good trying to find good options rapidly; exhaustive search wouldn't work because there aren't enough cycles and the local state space is pretty big.

2. Physics

(a) An automobile is accelerating while it turns a corner. In turning a corner too fast, it may skid. Draw a force diagram including the relevant forces.

(8 marks)

Marking scheme:
2 points friction of tires and road
2 points centripetal force
2 points velocity
2 points other forces



Sample answer:
A is acceleration due to the wheels turning.
f1 and f2 are friction on the tyres.
V = velocity of the car.
C = centrifical force from the turn
T = the angle of the wheels turning (forcing the slow turn)

Note, my answer is not perfect. There may be forces that are omitted, and in different directions.

(b) The coefficient of friction for rubber on concrete is .7. If the automobile ways 1000kg, how much force will be needed before it skids?

(9 marks)

Marking scheme:
3 points F=umg
3 points reasonable equation
3 points correct answer
Sample answer:
F = umg. g=9.81 m = 1000kg and u = .7 So F= 9.81*.7*1000 ~=7000 Newtons. Note this is the equation for static friction.

(c) What will happen if the automobile skids? Which of Newton's laws are relevant?

(8 marks)

Marking scheme:
4 points a reasonable comment about the friction between the tires and the road not being enough to overcome forces
2 points one law (1 or 2)
2 points both law
Sample answer:
When a car skids, the friction between the tires and the road is not enough to compensate for the forces on those tires. The may be rotational motion (when you accelerate too fast) or centripetal force from the turn. Newton's first law is relevant because the car wants to travel in a straight line from its velocity, and for that matter it wants to keep rotating (since its already got some rotational velocity). However, the second law is also relevant because the forces are acting in (somewhat opposing) right angles. One overcomes another, and the systems moves from a slow turn into a skid.

3. Software

(a) Games engines like XNA and Crystal Space are a form of middle ware. What is middle ware and what's its use?

(8 marks)

Marking scheme:
3 points reasonable definition.
3 points use is to reuse the code that is standard across games
2 points other notes such as API and fewer bugs.
Sample answer:
Middleware is in between a product and the operating system. In the case of games, the game (like Mario Kart) is the endware. Middle ware is used to reduce the effort to make a product. Libraries are typical examples of middle ware. Game engines make many of the functions commonly used in games available via an API. So rendering, motion, collision detection, and even AI can be made available in the engine. Making a new game then involves calling the API and the effort to develop the game is reduced.

Moreover, the game itself has fewer bugs, because the engine has had more effort put into removing the bugs (because it's used for more than one game).

(b) We've used both XNA and Crystal Space in labs. Compare and contrast these engines for the development of games.

(9 marks)

Marking scheme:
3 points product vs. shareware
3 points compatibility with other software.
3 points other reasonable points.
Sample answer:
XNA is a product made available by Microsoft. Consequently, it has fewer bugs, but you can't actually look at the source. This means you can't customise it as well as CS, but it is more likely that any game you write will be widely used. Another simple difference is that XNA is called from C# while CS is called from C++; they're probably both written largely in those languages too. XNA has some 3D functionality but not much. CS has a range of functionality that you can plug in (such as physics). CS can run on a range of platforms as well as link to a number of packages. CS also has a scripting language that comes along with it.

(c) Both XNA and Crystal Space use DLLs (Dynamic Link Libraries). What are DLLs?

(8 marks)

Marking scheme:
3 points Stores the engine
3 points
2 points
Sample answer:
DLLs are libraries that are used to store standard software. In both cases, the DLLs contain the game engine, and perhaps other external sources (e.g. XNA may use mingw for graphics). The coder doesn't typically use the source code for these libraries, but instead calls that code via relatively well defined APIs. The libraries are loaded at run time when necessary. The game may be distributed with or without the DLLs. If the game only uses some of the dlls, only those dlls are needed to play. Moreover, if a DLL is only rarely used, the game may be playable without it since it isn't loaded until it's needed.

4. AI

(a) Write a set of rules that explain a driver's behaviour at a stop light. (There should be around five.)

(10 marks)

Marking scheme:
3 points reasonable rule format

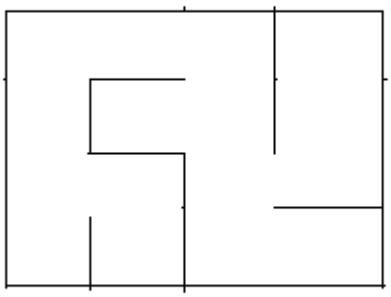
3 points reasonable red green stop go behaviour
2 points reasonable amber behaviour
2 points really sound rule set
Sample answer:

1. **If (light red) -> (stop)**
2. **If (light green) & (stop) -> (remove (stop)) (go)**
3. **If (light green) -> (go)**
4. **If (light amber) & (near light) -> (go)**
5. **If (light amber) -> (stop)**
6. **If (light amber) & (light red) -> (put in gear)**

(b) Draw a 4x4 maze. Draw a connectivity graph for that maze.

(6 marks)

(Marking scheme:
2 points maze
2 points some connectivity
2 points for correct connectivity graph
Sample answer: see attached



1,1-2,1-2,2-1,2
|- 3,1 -4,1-4,2-4,3-3,3-3,2
|- 2,3 -1,3-1,4
|- 2,4-3,4-4,4

c) Starting in the top left corner of the maze, perform a breadth first search. (Write out the steps.)

(9 marks)

Marking scheme:
3 points expanding multiple paths at a time
3 points describing paths
3 points correct
Sample answer:
4,1 ->
4,1-4,2;4,1-3,1
4-1-4,2-4,3 ; 4,1-3,1-2,1

4-1-4,2-4,3-3,3 ; 4,1-3,1-2,1-2,2 ; 4,1-3,1-2,1-1,1
4-1-4,2-4,3-3,3-3,2;4-1-4,2-4,3-3,3-2,3;4,1-3,1-2,1-2,2-1,2;4,1-3,1-2,1-1,1 (p1)
4-1-4,2-4,3-3,3-3,2(p3); 4-1-4,2-4,3-3,3-2,3-2,4; 4-1-4,2-4,3-3,3-2,3-1,3;
4,1-3,1-2,1-2,2-1,2(p2);(p1)
(p3); 4-1-4,2-4,3-3,3-2,3-2,4,3-4; 4-1-4,2-4,3-3,3-2,3-1,3-1,4;(p2);(p1)
(p3); 4-1-4,2-4,3-3,3-2,3-2,4,3-4,4;4-1-4,2-4,3-3,3-2,3-1,3-1,4(p4);(p2);(p1)