PAPER CODE NO.	EXAMINER : Xiaowei Huang
COMP222	DEPARTMENT : Computer Science Tel. No. 0151 795 4260

UNIVERSITY OF LIVERPOOL

Second Semester Examinations 2017/18

Principles of Computer Game Design and Implementation

TIME ALLOWED : Two Hours

INSTRUCTIONS TO CANDIDATES

Answer FOUR questions.

If you attempt to answer more questions than the required number of questions (in any section), the marks awarded for the excess questions answered will be discarded (starting with your lowest mark).

UNIVERSITY OF LIVERPOOL

Question 1

A. Describe the code structure of a modern computer game. Your answer should mention *game-specific* code, *game engine*, and *in-house tools*. You should also cover a typical game architecture to include *initialisation*, *main game loop* and *cleanup*. Give a diagrammatic representation of a typical game architecture. **6 marks**

Most games make a distinction between game-specific code and game-engine code. Game-specific code is, as the name implies, tailored to the current game being developed. It involves the implementation of specific parts of the game domain itself, such as the behaviour of zombies or spaceships, tactical reasoning for a set of units, or the logic for a front-end screen. This code is not intended to be generically reused in any other game in the future other than possibly direct sequels.
Game-engine code is the foundation on top of which the game-specific code is built. It has no concept of the specifics of the game being developed, and deals with generic concepts that apply to any project: rendering, message passing, sound playback, collision detection, or network communication.
In-house development tools are created to support artists and developers in creation of worlds and content.
Initialisation and shutdown of different systems and phases of the game is a very important step, yet it is often overlooked. Without a clean and robust way of initialising and shutting down different parts of the game, it becomes very difficult and error prone to switch between levels, to toggle back and forth between the game and the front end, or to even run the game for a few hours without crashes or slowdowns.

Main Game Loop: At their heart, games are driven by a game loop that performs a series of tasks every frame. By doing those tasks every frame, we put together the illusion of an animated, living world. Sometimes, games will have separate loops for the front end and the game itself, since the front end usually involves a smaller subset of tasks than the game. Other times, games are organised around a unified main loop.
A description of the code structure gives 3 marks, 1 mark for the picture; description so initialisation, game loop and cleanup is 1 mark each.

B. Describe the typical functionality of a physics engine, and name at least two advantages of using a third-party physics engine and at least two advantages of using an in-house physics routine. **6 marks**

A physics engine is computer software that provides an approximate simulation of certain simple physical systems, such as rigid body dynamics (including collision detection), soft body dynamics, and fluid dynamics,
Advantages of game engines:
- Complete solution from day 1
- Proven, robust code base (in theory)
- Lower costs
Advantages of home-grown solutions:
- Choose only the features you need
- Opportunity for more game-specific optimisations
- Greater opportunity to innovate

C. Discuss the difference between the traditional Artificial Intelligence discipline and Artificial Intelligence in computer games. **4 marks**

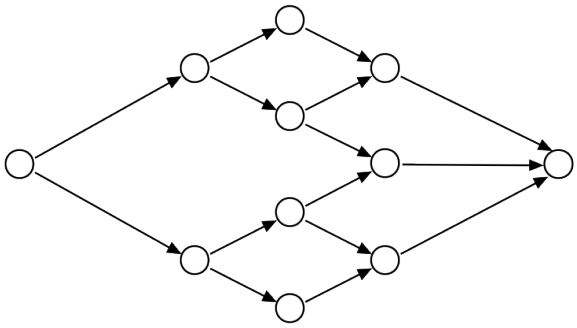
In academia, some AI researchers are motivated by philosophy: understanding the nature of thought and the nature of intelligence and building software to model how thinking might work. Some are motivated by psychology: understanding the mechanics of the human brain and mental processes. Others are motivated by engineering: building algorithms to perform human like tasks.
Games developers are primarily interested in only the engineering side: building algorithms that make game characters appear human or animal-like.
In academia, AI is about the matter. In games AI is about the appearance.

D. A computer game can be defined as a sequence of meaningful choices made by the player in pursuit of a clear and compelling goal. Justify such a definition and give a graphical representation of the classical game structure. **4 marks**

Justification:

- Must have choice, or it is not interactive
- Must be a series of choices or it is too simple to be a game
- Must have a goal or it is a software toy

Graphical representation:



Starts with a single choice, widens to many choices, returns to a single choice

E. Games are driven by a game loop that performs a series of tasks every frame. Traditionally,

games only featured one game loop. Some game architectures suggest running multiple game loops in different threads. What is the reason for doing this? Name at least one advantage and one disadvantage of such an approach to multithreading. **5 marks**

This architecture allows the developers to run different game subsistems in different update loops, e.g. Heads-up Display, AI, Rendering, Physics, etc all run in different threads.
Advantages:

- Better use of hardware resources;
- Clear separation between subsystems;
- Developers can run different subsystems at different speeds.

Disadvantages

- The cost of switching context on older architecture (where multithreading is software not hardware driven) – less important these days;
- Game subsystems are tightly coupled and making the execution thread safe might be tricky;
- Most of the time the threads will be waiting – e.g. the physics simulation thread will wait until new updates come from the user.

Every advantage / disadvantage is 1 mark. A comparison of a single loop / multiple loops is 2 marks. For 1 extra mark: a much better approach is to split the load into small independent units and run them on the available CPUs.

Question 2

A. Let $\mathbf{V} = (3, 2, 1)$ and $\mathbf{W} = (4, 2, -6)$ be 3D-vectors. Compute (and show your working)

(a) $\mathbf{V} + \mathbf{W}$ **1 marks**

$$\mathbf{V} + \mathbf{W} = (3 + 4, 2 + 2, 1 - 6) = (7, 4, -5)$$

(b) $2\mathbf{V}$ **1 marks**

$$2\mathbf{V} = (6, 4, 2)$$

(c) $\mathbf{V} - 2\mathbf{V}$ **1 marks**

$$\mathbf{V} - 2\mathbf{V} = -\mathbf{V} = (-3, -2, -1)$$

(d) $\mathbf{V} \cdot \mathbf{W}$ **1 marks**

$$(3, 2, 1) \cdot (4, 2, -6) = 3 \cdot 4 + 2 \cdot 2 - 1 \cdot 6 = 12 + 4 - 6 = 10$$

(e) $|\mathbf{V}|$ **1 marks**

$$|\mathbf{V}| = \sqrt{3^2 + 2^2 + 1^2} = \sqrt{14}$$

(f) $\text{proj}_{\mathbf{V}}\mathbf{W}$ **2 marks**

$$\text{proj}_{\mathbf{V}}\mathbf{W} = \frac{\mathbf{W} \cdot \mathbf{V}}{\|\mathbf{V}\|^2}\mathbf{V},$$

so

$$\text{proj}_{\mathbf{V}}\mathbf{W} = \frac{10}{14}(3, 2, 1) = \left(\frac{15}{7}, \frac{10}{7}, \frac{5}{7}\right)$$

(g) $\mathbf{V} \times \mathbf{W}$ **2 marks**

$$(3, 2, 1) \times (4, 2, -6) = (-2*6 - 1*2, 1*4 + 3*6, 3*2 - 2*4) = (-14, 22, -2)$$

B. Recall that the 2D rotation matrix representing the counter-clockwise rotation about the origin by an angle θ is as follows

$$\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

Give the 3D rotation matrix representing the counter-clockwise rotation about the Z-axis through an angle α combined with counter-clockwise rotation about the X-axis through an angle β. **8 marks**

Rotation about the Z-axis is given by

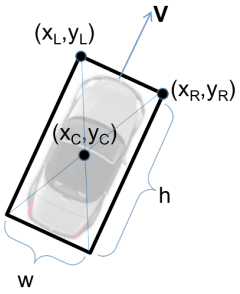
$$M_Z = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Rotation about the X-axis is given by

$$M_X = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\beta & -\sin\beta \\ 0 & \sin\beta & \cos\beta \end{bmatrix}$$

Their combination is given by the matrix produce $M_Z \times M_X$.

C. Assume you are implementing a simple 2D racing game with a top view of the track. In this game a user is in control of a vehicle, which is approximated as a rectangular shape. Collision detection is implemented by checking the positions of the front corners of the vehicle.



Given the dimensions h and w of the vehicle, coordinates (x_C, y_C) of its centre and its heading vector **V** determine the coordinates of the forward left corner (x_L, y_L) and forward right corner (x_R, y_R), as illustrated in the figure above.

8 marks

Let **U** be the unit vector obtained by normalisation of **V**, $\mathbf{U} = \frac{1}{|\mathbf{V}|} \cdot \mathbf{V}$. Let $\mathbf{U} = (x_U, y_U)$. Then $\mathbf{W} = (x_W, y_W) = (y_U, -x_U)$ is the (unit) vector obtained by rotation of **U** $90°$ counter-clockwise.
Then $(x_L, y_L) = (x_C, y_C) + \frac{1}{2}h\mathbf{U} - \frac{1}{2}w\mathbf{W}$; $(x_R, y_R) = (x_C, y_C) + \frac{1}{2}h\mathbf{U} + \frac{1}{2}w\mathbf{W}$.

Question 3

A. Modern computer games commonly use scene graphs to represent a graphical scene. Name at least three advantages of this form of data representation as compared with unstructured collections of geometries, light sources, textures, etc. **3 marks**

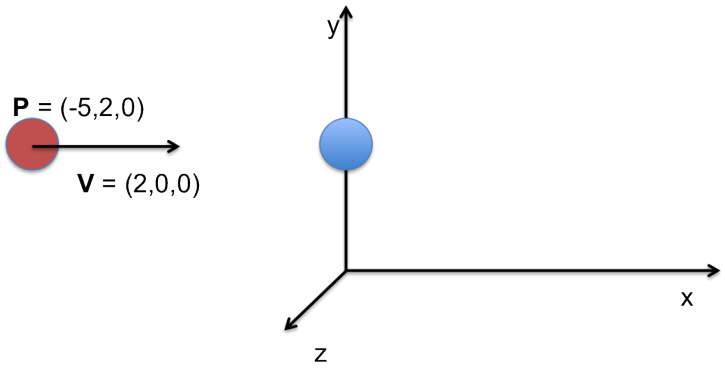
Some advantages:

- Scene graphs provide an intuitive way to manage large amounts of geometric and rendering data
- The data needed for rendering, which is associated with the scene graph nodes, can be kept separate from the rendering code.
- Hierarchical animated models are easier to deal with
- View frustum culling can be supported by using bounding volumes at the nodes.

B. In this module we studied two major approaches to collision detection: overlap testing and intersection testing. Define these approaches and discuss their advantages and disadvantages. **6 marks**

The main difference is that overlap testing detects whether a collision has already occurred, and intersection testing predicts if a collision will occur in the future.
The idea of overlap testing is that at every simulation step, each pair of objects will be tested to determine if they overlap with each other. If two objects overlap, they are in collision. This is known as a discrete test since only one particular point in time is being tested.
The biggest advantage of overlap testing is that it is easy to implement. It's biggest disadvantage is that it handles poorly objects travelling fast. For overlap testing to always work, the speed of the fastest object in the scene multiplied by the time step must be less than the size of the smallest collidable object in the scene. This implies a design constraint on the game to keep objects from moving too fast relative to the size of other objects.
Intersection testing tests the geometry of an object swept in the direction of travel against other swept geometry. Whatever geometry the object is composed of, it must be extruded over the distance of travel during the simulation step and tested against all other extruded geometry.
The disadvantages of overlap testing include a poor handling of networked games. The issue is that future predictions rely on knowing the exact state of the world at the present time. Due to packet latency in a networked game, the current state is not always coherent, and erroneous collisions might result. Therefore, predictive methods aren't very compatible with networked games because it isn't efficient to store enough history to deal with such changes and, in practise, running clocks backward to repair coherency issues rarely works well.
One more potential problem for intersection testing is that it assumes a constant velocity and zero acceleration over the simulation step. This might have implications for the physics model or the choice of integrator, as the predictor must match their behaviour for the approach to work.

C. A ball with coordinates $(-5, 2, 0)$ moves uniformly with a constant speed given by vector $(2, 0, 0)$.



(a) Describe the discrete motion of the ball as a sequence of its positions using Euler steps. **4 marks**

To use Euler steps, we need to upate the forces, acceleration, velocity and position of the cannonball at every frame as follows. In this specific case, no forces act on the ball, so Euler steps are as follows.

$$\begin{aligned} \mathbf{F}_{i+1} &= 0 \\ \mathbf{a}_{i+1} &= 0 \\ \mathbf{V}_{i+1} &= \mathbf{V}_i \\ \mathbf{P}_{i+1} &= \mathbf{P}_i + tpf \cdot \mathbf{V}_{i+1} \end{aligned}$$

(b) Sketch the main game loop to model the uniform motion of the ball. Your answer should take the frame rate into account. **4 marks**

- Let $pos := (-5, 2, 0)$
- Let $V := (2, 0, 0)$
- While (loop not finished)
  - $pos := pos + 5 * tpf * V$
- EndWhile

where tpf is the time per frame corresponding to the current frame rate.

(c) Suppose that there is a stationary ball of the same mass with coordinates $(0, 2, 0)$ as shown in the picture above.

- Compute the exact moment of time when the collision between the balls happens; **4 marks**
  The distance between balls is 5. Hence, the ball will collide with the plain in exactly 2.5 time units.
- What will be the outcome of the collision? **4 marks**
  After the collision, the first ball becomes stationary and the second starts moving with the speed given by vector $(2, 0, 0)$.

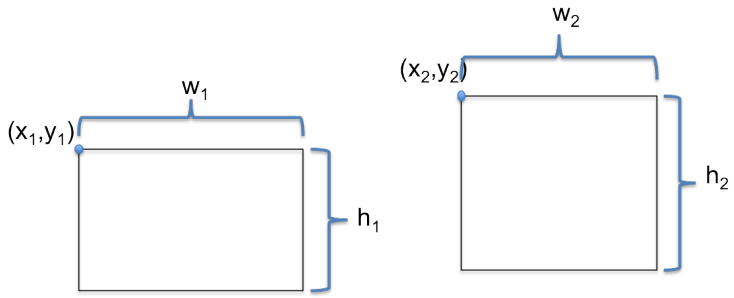
Question 4

A. Overlap testing in computer games is often approximated with the help of *bounding volumes*: a real shape is being embedded into a simplified geometry, and if two bounding volumes do not overlap, one does not perform an (expensive) triangle-level overlap test.

(a) Simple bounding volume shapes include Axis Aliened Bounding Boxes (AABBs) and Oriented Bounding Boxes (OBBs). What are the advantages of OBBs over AABBs? Are there any significant disadvantages? **5 marks**

OBBs fit the real geometry tighter. The main disadvantage is that it is harder to check if the bounding volumes overlap; however, this disadvantage is outweighed by better collision detection offered by OBBs.

(b) Sketch a method which, given the coordinates of *upper left* corners of two 2-dimensional axis-aligned boxes (x_1, y_1) and (x_2, y_2) and their widths w_1, w_2 and heights h_1, h_2, respectively, determines whether these boxes intersect.

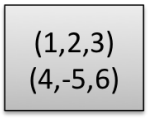


7 marks

There is an elegant solution to this problem based on the check of when boxes *do not* overlap.

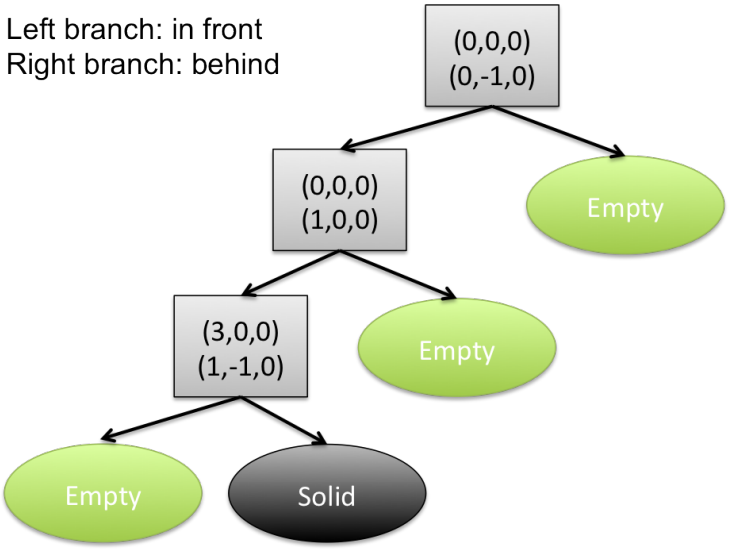
```
if((x1 + w1 < x2) || (x2 + w2 < x1) ||
   (y1 - h1 > y2) || (y2 - h2 > y1)) {
        return false;
}
else {
        return true;
}
```

B. Recall that a node of a solid-leaf BSP tree can be *solid*, *empty*, or it can be an internal node associated with the plain that partitions the space. In the diagram below, the plain associated with an internal (shown as a box) node is determined by a position vector (first three numbers) and a normal vector (the second line). For example, for the internal node



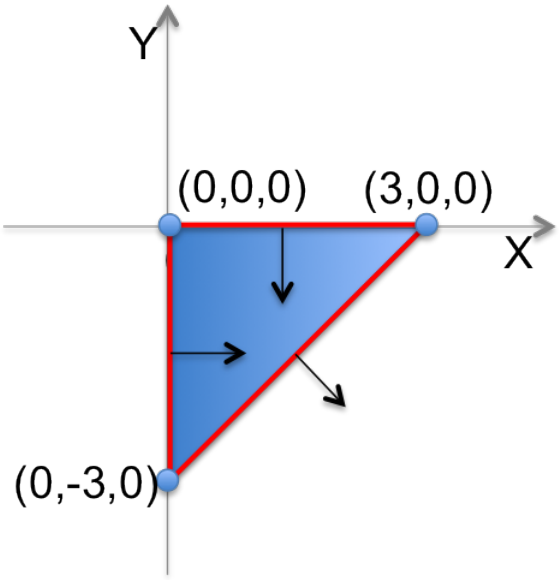
the position vector is (1,2,3) and the plain normal is (4,-5,6).

Sketch the geometrical shape defined by the solid-leaf BSP tree shown below.

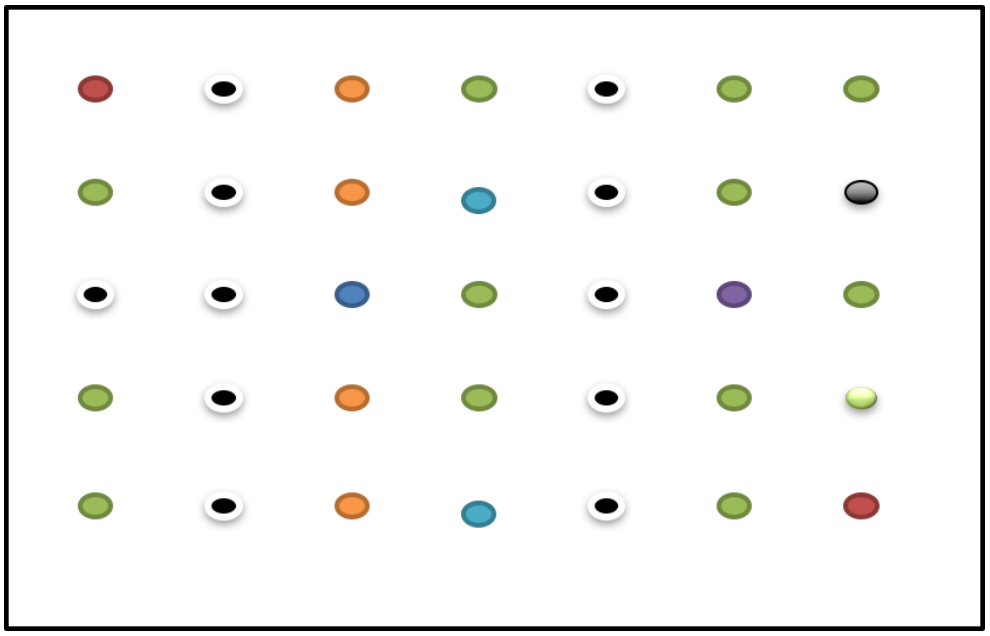


Mark clearly on your drawing the position and normal vectors for each plain. **7 marks**

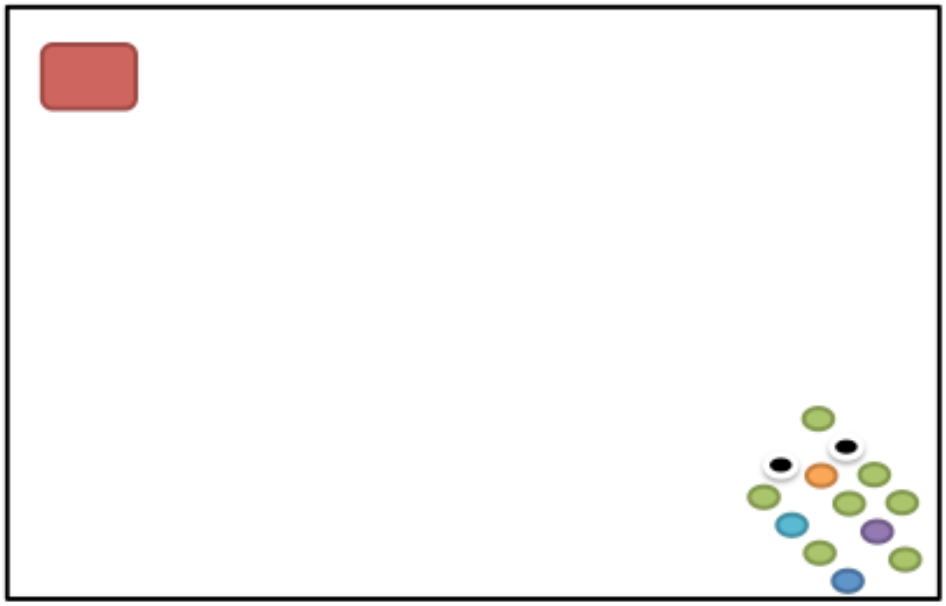
The BPS tree determines the following shape:



C. In your opinion, what data structure is most suitable to reduce the number of pairwise collision detection tests in the scene shown below? Explain your reasoning.



Moreover, what data structure is most suitable to reduce the number of pairwise collision detection tests in a scene where there is one large static object in one corner and a number of small static object in the other as shown below? Explain your reasoning



6 marks

In the first scen, objects are similar size and uniformly distributed. A uniform grid is the best choice, though a quad tree, a k-d tree of a BSP tree can also be used.
In the second scene, it's best to use non-uniform grid based data structure such as a quad tree, a k-d tree of a BSP tree.

Question 5

A. In agent-based computer game AI, intelligent agents continually step through the *sense-think-act* cycle. In your opinion, what is the necessity of the *sense* step of the cycle since the game world is always entirely represented inside the game and perfect information about the state of the world is always available? **5 marks**

Game agents are autonomous entities associated with game characters (enemies, companions, computer car drivers etc.).
While all of this rich information exists, it may be expensive or difficult to tease out useful and pertinent information. Additionally, at any time, the game agent can query the game world representation to locate the player or other enemies, but most players would consider this cheating. Therefore, it is necessary to endow the game agent with certain limitations in terms of what it can sense. For example, game agents should typically not be able to see through walls. Game agents are usually given human limitations. They are restricted to knowing only about events or entities they have seen, heard, or perhaps were told about by other agents. Therefore, it is necessary to create models for how an agent should be able to see into the world, hear the world, and communicate with other agents.

B. Consider the following behaviour of a fighter game agent. The agent can be in three possible states, *idle*, *patrol*, or *attack*. In the *idle* state the agent remains motionless, in the *patrol* state the agent moves to the next checkpoint, and in the *attack* state the agent attacks another player. If the agent sees the other player, it goes into the *attack* state; otherwise, from being idle it changes, on a timeout, to the *patrol* state and, once completed the move to the next checkpoint, returns to the *idle* state. If the enemy unit is destroyed, the agent goes from the *attack* state to the *idle* state.

(a) What AI technique is best suitable to represent the behaviour of such an agent? **2 marks**

Finite state machine

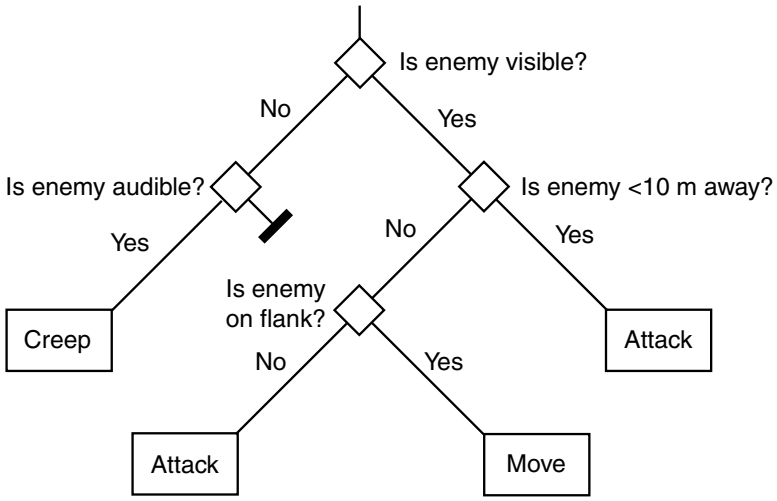
(b) Give a graphical representation of this model of agent behaviour. Indicate clearly conditions under which one state changes into another. 5 **marks**

(c) Assume now that you want the agent to show more complicated behaviour: in the *patrol* state the agent patrols four stations S_1,...,S_4 in the order $S_1 \rightarrow S_2 \rightarrow S_3 \rightarrow S_4 \rightarrow S_1 \rightarrow \ldots$ and in the *attack* state the agent goes through three consecutive stages: *approach*, *aim*, *fire*.

In your opinion, what is the best way to accommodate these modifications to the agent behaviour? Give a graphical representation of the new model of agent behaviour. **8 marks**

There are two options how this can be handled. Either to add more states to the FSM, or consider a hierarchical FSM in which the *patrol* state and *attack* state are FSMs as follows.

C. Describe in plain English the agent strategy given diagrammatically by the following decision tree.



5 marks

If the enemy is visible and within 10 meters away then attack. If the enemy is visible, further than 10 meters and is on flank then move. If the enemy is visible, further than 10 meters and is not on flank then attack. If the enemy is not visible but is audible then creep.

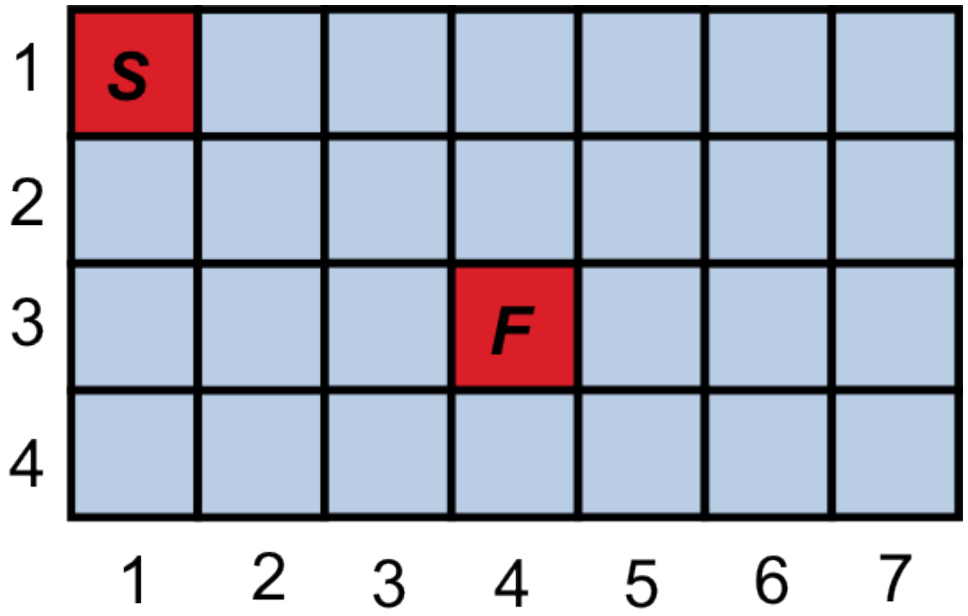
Question 6

A. Why in computer games is the character motion control routine often considered at two logical levels: steering and pathfinding? Name at least two advantages of such separation. **4 marks**

Steering techniques allow a computer character to navigate from one position into another provided there are no (or few simple) obstacle on the way. Pathfinding is used whenever a computer entity needs to find a way to a goal avoiding obstacles.
Advantages:

- steering is closer to game engine an often requires integration with the physics engine; pathfinding is closer to the decision taking level. Keeping them separate leads to a cleaner code and better task distribution.
- Pathfinders can be reused in a different kind of game even of another genre.

B. Consider the following tile-based map.



The only permitted movements are up-down and left-right (if the adjacent tile exists).

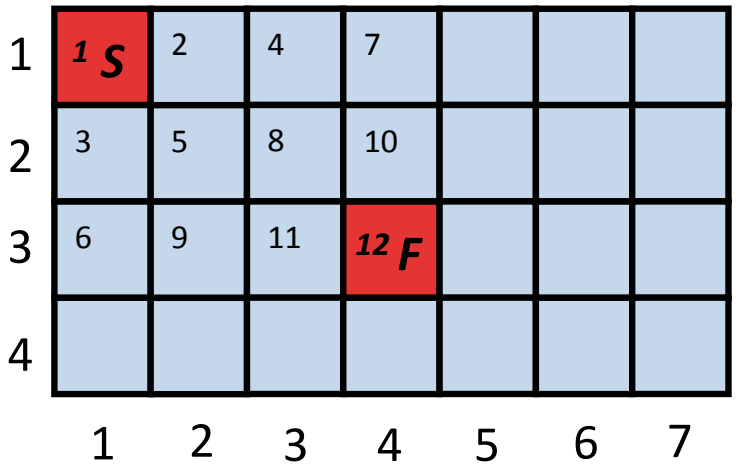
(a) Define class `Node` referring to a tile on the map, that could be used by a pathfinding algorithm. **3 marks**

Students get full marks for

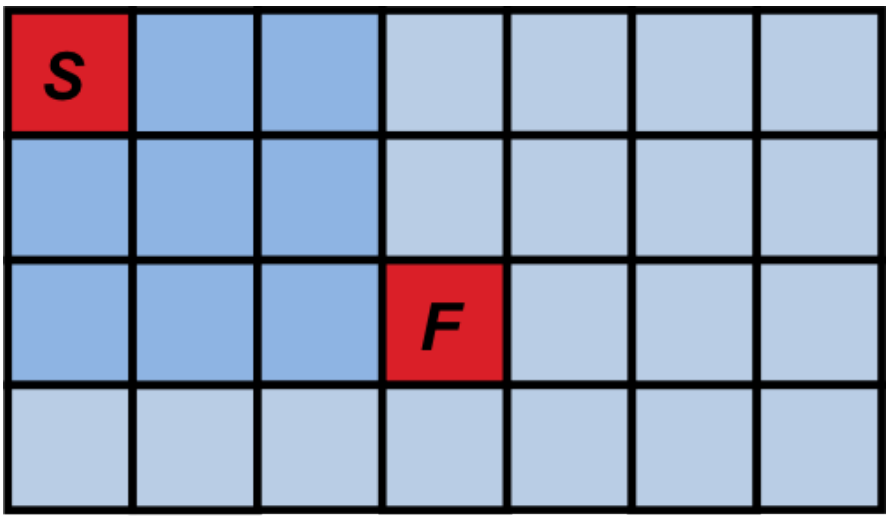
```
public class Node {
    int x,y;
}
```

(b) Using Manhattan block distance as heuristics

i. Apply the A* algorithm to find a path from the start tile (marked with ***S***) to the finish tile (marked with ***F***). **6 marks**
There is more than one way to explore the tiles. Here is one example



ii. Shade the tiles which will be considered by the A* algorithm with Manhattan block distance heuristic. **6 marks**



(c) Suggest a different heuristic to reduce the number of nodes explored by the A* algorithm. Will this heuristic be admissible? **6 marks**
Manhattan block distance multiplied by a small number. Although this heuristic is inadmissible, it is more goal-oriented and would make the A* search more greedy.