

DUBLIN INSTITUTE OF TECHNOLOGY
KEVIN STREET, DUBLIN 8

BSc Computer Science

Year 4

SUPPLEMENTAL EXAMINATIONS 2006

GAMES PROGRAMMING

Mr. Bryan Duggan

Mr. Hugh McAtamney

Dr. Brendan O' Shea

Dr. K. Bechkoum

Time 3 Hours

Attempt any TWO Questions from Section A and any TWO questions from Section B

All questions carry equal marks

SECTION A

1. (a) What is meant by the following terms: *DirectX*, *anisotropic filtering*, *texturing* and *transform* in relation to 3D computer games.

(4 marks)

- (b) (i) List the basic operations required to implement a camera in a 3D FPS (First Person Shooter).

(2 marks)

- (ii) Given that a camera class for an FPS game engine contains the data members in Figure 1, write code to implement the operations you listed in your answer to part (i).

```
D3DXVECTOR3 _pos; \\ The camera position
D3DXVECTOR3 _up; \\ Up
D3DXVECTOR3 _look; \\ The direction the camera is looking
D3DXVECTOR3 _right; \\ A vector orthogonal to _up and _look
```

Figure 1

(7 marks)

- (iii) Describe how to generate the DirectX **view transform matrix** from the 4 vectors identified in Figure 1. In your answer, include **either** a description or code.

(7 marks)

- (c) Assuming a character in a computer game is represented by a C++ class holding the data members listed in Figure 2 and is drawn using a mesh, write the DirectX code to calculate the **world transform matrix** required to *position* and *orientate* the character in a 3D world.

```
D3DXVECTOR3 _pos; \\ The entities position
D3DXVECTOR3 _up; \\ Up
D3DXVECTOR3 _look; \\ The direction the entity is heading
D3DXVECTOR3 _right; \\ A vector orthogonal to _up and _look
float _yaw; \\ The angle to yaw in radians
float _pitch; \\ The angle to pitch in radians
```

Figure 2

(5 marks)

2. (a) What is meant by the terms *singleton* and *lazy loading* in relation to asset management in computer games. Include in your answer a simple example of how to implement these techniques in C++. (6 marks)

- (b) Figure 3 shows classes that exist in a computer game engine:

```
// Represents a rectangular wall
class Wall
{
    D3DXVECTOR3 _lowerBound;
    D3DXVECTOR3 _upperBound;
};

// Represents the player
class Player
{
    D3DXVECTOR3 _pos; // The players position in the world
};

// Represents the enemy
class Enemy
{
    D3DXVECTOR3 _look; // The direction the enemy is looking
    D3DXVECTOR3 _pos; // The enemy's position in the world
    World * _world; // The world
    float _fov; // the enemy's field of view in radians
};

class World
{
    std::vector<Wall*> _walls; // A vector of walls
    Player * _player;
    Enemy * _enemy;
};
```

Figure 3

Formulate an approach for detecting if the game character represented by the Enemy class can *see* the game character represented by the Player class. In your answer include:

- (i) A diagram illustrating a possible scenario. (3 marks)
 - (ii) A detailed description of any algorithms you propose implementing. (6 marks)
 - (iii) A listing of any new API's you propose implementing. (4 marks)
 - (iv) A critique of your approach. (1 marks)
- (c) In your opinion, what are the key issues surrounding the use of *physics engines* in commercial games. Give examples of *physics based gameplay* and *scripted gameplay* from commercial games.

(5 marks)

3. (a) “Using *hardware accelerated graphics* can speed up the performance of the *Euclidian distance heuristic* in an implementation of the *A* algorithm*.”

Explain the *bold italicised* terms in the above statement and justify the statement.

(5 marks)

- (b) Figure 4 presents an abstraction of a game world, where *S* represents the starting position and *D* represents the destination of a game character. An *X* represents non traversable nodes.

			S			
		X	X		X	X
					D	

Figure 4

Use the A* algorithm with the Manhattan distance heuristic to calculate the shortest path from *S* to *D*.

In your answer include:

- The *f*, *g* and *h* scores for each node considered at each iteration of the algorithm.
- Nodes that are expanded and nodes that are no longer to be considered.

(10 marks)

- (c) What are the access requirements of the *open* and *closed* lists in an implementation of the A* algorithm. What data structures from the STL would you use to optimally meet these requirements?

(5 marks)

- (d) What is meant by the term *edge relaxation* in relation to pathfinding. Include an example in your answer.

(5 marks)

SECTION B

4. (a) Give a brief synopsis of the history of video games. Illustrate your answer using examples of both current and early generation gaming systems. (8 marks)
- (b) Summarise the 6 elements described in Jesper Juul's Classic Game Definition. (6 marks)
- (c) "If a particular game unit is sufficiently over-effective for its cost, it could make another unit completely useless in most or all circumstances"
- Discuss this statement with reference to *game balance*. Illustrate your answer with examples from commercial games. (8 marks)
- (d) Identify the key issues in the debate between *Setting* versus *Gameplay* (3 marks)
5. (a) "Visual sensors play a major role in the in the Crytek engine's AI perception system"
- Discuss this statement. Illustrate your answer with examples from the game FarCry. (13 marks)
- (b) Describe briefly how the *alert status* in the FarCry Game engine works. List 3 properties that you might change in the sandbox editor to affect it. (3 marks)
- (c) What is the relationship between the different types of *entities* in the Crytek game engine? How might you go about creating a game entity in Crytek? (5 marks)
- (d) In FarCry, what is the *stealth o meter*? Describe briefly how it works. (4 marks)
6. (a) Describe in detail how the CryAI system calculates *pathfinding* using *graphs* and *triangulation*. (10 marks)
- (b) In the context of the CryAI system, what is *beautification* and why is it used? (4 marks)
- (c) Explain why *game events* are the nervous system of the game FarCry. (4 marks)
- (d) What is *game scripting*. Why is the LUA language used for scripting in FarCry. (7 marks)