# Winter

## Q1

1. In a game, an Archer will hit a target, if he is facing the target, which in this case means the angle to target is less than 90 degrees.

Vector3 Target = new Vector3(38, 7, 100);

Vector3 ArcherPosition = new Vector3(10, 0, 156);

Vector3 ArcherForward = Vector3.normalise(new Vector3 (-2, 1, -2));

1. Find the vector from the Archer to the Target

(38, 7, 100) – (10, 0, 156)

**= (28, 7, -56)**

1. How far is the Archer from the Target?

√ (28² + 7² + 56²)

= √ (784 + 49 + 3136)

= √ (3969)

**= 63**

1. Calculate ArcherForward.

√ (2² + 1² + 2²)

= √ (4 + 1 + 4)

= √ (9)

= 3

**AcherForward = Vector3(-2/3, 1/3, -2/3)**

1. Use Scalar Products to determine if Archer is facing the target.

(-2/3, 1/3, -2/3) · (28/63, 7/63, -56/63)

= (-2/3, 1/3, -2/3) · (4/9, 1/9, -8/9)

= (-8/27, 1/27, 16/27)

= 9/27

= 1/3

**1/3 > 0, therefore Archer is facing target.**

1. A successful attack is permitted if the distance is less than 80m, and the Archer is facing the target. Will this attack be successful?

63m < 80m

1/3 > 0

**Attack is successful.**

1. What is the direction of for the arrow?

**(28/63, 7/63. -56/63)**

1. Given the arrow has a fixed speed of "ArrowSpeed" how would you assign the initial velocity to the arrow.

**( ((28/63)\*ArrowSpeed), ((7/63)\*ArrowSpeed), ((-56/63)\*ArrowSpeed) )**

1. Frame rates are a key consideration when implementing movement, and in particular keeping movement Frame Rate Independent
2. What are the physical rules governing motions that are used to ensure Frame Rate Independence?

**Time.deltaTime**

1. Illustrate (code or pseudo code) how Frame Rate Independent motion could be implemented.

**transform.position += transform.forward \* Time.deltaTime**

1. Illustrate how forces could be applied to an object, giving justification with reference to the appropriate physics formula

## Q2

1. BIOT was used to determine collisions between sprites.
2. What was BIOT?

**Binary Image Overlap Test**

1. The following sprite would have been stored in numeric form, derive these numbers.

128

64

32

16

8

4

2

1

24

231

60

231

255

102

126

195

1. Draw the sprite represented by the following bytes.

110

100

105

110

105

100

1. Axis Aligned Bounding Boxes (AABB) are not an accurate a means of detecting collisions.
2. Describe why AABB's are not accurate and outline their use in collision detection within a modern games engine.

**Eg. A rotating arrow would have a large AABB but a very small OBB**

**AABB are used in the broad phase of collision detection**

**AABBs are used in the AABB Intersection Test (Sort and Sweep) : Which produces collision islands to be passed on to the Narrow phase. This reduces the overall number of testes required which can grow very large (Handshake problem)**

1. The following table outlines the positions and ranges of the AABB's for 3 objects, a player and 2 enemies. Apply the Sort and sweep algorithm to determine if a collision has occurred.

|  |  |  |  |
| --- | --- | --- | --- |
|  | Player | Enemy 1 | Enemy 2 |
| X | 105 | 100 | 148 |
| +- | 14 | 4 | 12 |
| Y | 249 | 231 | 223 |
| +- | 23 | 13 | 6 |
| Z | 33 | 39 | 21 |
| +- | 8 | 3 | 6 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| X-Axis | Ps | E1s | E1e | Pe | E2s | E2e |
| A | 91 | 96 | 104 | 119 | 136 | 160 |
| B | P | P, E1 | P | NULL | E2 | Null |
| C |  | **(P,E1)** |  |  |  |  |
|  |  |  |  |  |  |  |
| Y-Axis | E2s | E1s | Ps | E2e | E1e | Pe |
| A | 217 | 218 | 226 | 229 | 244 | 272 |
| B | E2 | E2, E1 | E2, E1, P | E1, P | P | NULL |
| C |  | (E2, E1) | (E2, P) **(E1, P)** |  |  |  |
|  |  |  |  |  |  |  |
| Z-Axis | E2s | Ps | E2e | E1s | Pe | E1e |
| A | 15 | 25 | 27 | 36 | 41 | 42 |
| B | E2 | E2, P | P | P, E1 | E1 | NULL |
| C |  | (E2, P) |  | **(P, E1)** |  |  |