OLLSCOIL NA GAILLIMHE
UNIVERSITY OF GALWAY

Autumn Examinations 2022-2023 ***MARKING SCHEME***

Course Instance Code(s)	3BCT, 3BDA
Exam(s)	B.Sc. (CS&IT), B.A. (Digital Arts & Technology)
Module Code(s)	CT3536
Module(s)	Games Programming
Paper No.	1
External Examiner(s)	Dr. R. Trestian
Internal Examiner(s)	Prof. M. Madden *Dr. S. Redfern

Instructions: Answer any three questions. All questions carry equal marks. Note that the final page of this exam paper lists useful classes from the Unity3D SDK.

Duration	2 hours
No. of Pages	4
School	Computer Science
Course Co-ordinator(s)	Dr. C. O'Riordan, Dr. P. Killeen

Requirements:

Release in Exam Venue	Yes [x]	No [ ]
MCQ Answersheet	Yes [ ]	No [x]
Handout	None	
Statistical/ Log Tables	None	
Cambridge Tables	None	
Graphic material in colour	Yes [x] No [ ]	
Graph Paper	None	
Log Graph Paper	None	
Other Materials	None	

PTO

Q.1.

Making appropriate use of local and global co-ordinates, write Unity3D/C# code to perform the following transformations. You may assume that references to the runtime gameobjects are provided:

- rotate a gameobject 5 degrees around its own x axis [2]
- Half marks if rotation is applied via the world coordinate system

- move a gameobject 6 units downwards in the world's co-ordinate system [2]
- Half marks if translation is applied via the object's own coordinate system

- move a gameobject 7 units directly towards another gameobject [3]
- Calculation of difference between object positions [1]
- Normalization and difference vector, and multiplication of this by 7 [1]
- Translation of 1st game object [1]

- move a gameobject 10 units forward in whatever direction it is facing [3]
- Translation by 10 units [1]
- Correct use of transform.forward or similar [2]

(ii) Write code for the following method, which considers the supplied list of objects and returns the one which is furthest away from the specified 3D point: [10]

```
public static GameObject GetFurthestObject(List<GameObject> objects, Vector3 pos) {

}
```

- Iteration through list [2]
- Calculation of distance between each list object and 'pos' [4]
- Correct identification of maximal distance [2]
- Returning furthest object [2]

Q.2.

Write technical notes (approx. 150 words) on *each* of the following:

(i) The use of State Machines to structure game logic. [5]

Definition of State Machine. [2]
Clear separation of logic at different times [2]
Example(s). [1]

(ii) Screen space, viewport space and world space in Unity, including how to transform between them. [5]

Screen space: on-screen pixels (2D). [1.5]
Viewport space: normalized on-screen position (2D). [1.5]
World space: position in the virtual world (3D). [1]

Camera class transforms between these spaces, according to the viewpoint of the camera. [1]

(iii)The Object Pool pattern – why it's useful and how it operates [5]

Maintaining inactive objects in a data structure rather than destroying them [2]
How you set game objects inactive/active in Unity [1]
Importance of low-garbage code in games [2]

(iv) Coroutines in Unity, including two different situations for which Coroutines would be useful [5]

Coroutines are Monobehaviour methods which can be paused for varying time [2]
Unity internally maintains a list of paused Coroutines and the Game Loop processes this each frame and resumes those whose pause time has elapsed. [1]
Situations might include: gathering timed logic together into one method; animating an object's properties over times; waiting for other coroutines to end before continuing; carrying out CPU-intensive operations over multiple frames; and others! [2x1]

Q.3.
(i) In 3D games development, what does the term **'raycast'** mean, as supported by various static methods of the Unity3D SDK's Physics class? Explain, with illustrative C# code, how you could use a raycast to allow the user to click with the mouse and select a game object from the scene. [10]

- Definition of raycast concept [2]
- Specific reference to raycasting against world geometry [1]
- Identifying mouseclick in Unity [1]
- Transforming 2D screen point to 3D world position [1]
- Obtaining raycast direction vector via Camera's forward vector [1]
- Identifying the world object that was hit [2]
- Illustrative C# code [2]

(ii) In a shooting game, assume you are using raycasts to determine what the player has hit when they fire their gun. You may assume that you are given a reference to the gun object in the 3D scene.

- Write appropriate Unity3D/C# code to perform a raycast when the gun is fired, to determine what is hit by the bullet. The gun should have a maximum range of 500 metres. [6]

- Construction of Ray struct (or separate Vector3 structs) for: source position, and raycast direction [3]
- Correct use of Physics.Raycast() with Ray and distance 500 [2]
- Identification of what is hit (or nothing hit) [1]

- Write appropriate Unity3D/C# code to instantiate an ‘explosion’ object at the position that the bullet hits. You may assume that a prefab exists for this explosion object. [4]

- Use of GameObject.Instantiate() [2]
- Correct position of resulting object using data returned by Physics.Raycast [2]

PTO

Q.4.

(i) Write a C# Monobehaviour script to attach to a Unity3D game object which automatically destroys the object as soon as it is either behind the camera or more than a defined distance away from it. This defined distance should be available as a value that can be edited in the inspector. [8]

• public float distance	1
• Calculates distance from camera to game object	2
• Camera.main	1
• Camera.main.WorldToViewport	2
• Destroy(gameObject)	1
• Code in Update( ) ` `	1

(ii) What are C# Coroutines? [2]

• Methods that can be paused and resumed	2

(iii) Write a Coroutine which carries out a sequence of actions over time: [8]

- Gradually (frame by frame) moves its local game object at a speed of 1 metre per second towards a Vector3 position.
- After the game object arrives at the position, waits 2 seconds.
- Then moves the game object in the same way to a second Vector3 position.

Your Coroutine should use the following signature:

```
IEnumerator MoveBetween(Vector3 pos1, Vector3 pos2)
{

}
```

- Loop until arrived at 1st point 1
    - Calculate distance to arrival 1
    - Avoid overshoot 1
    - Move 1m/s using time.deltaTime 1
    - Move in correct direction 1
    - yield return null 1
- yield return new WaitForSeconds(2f) 1
- Similar loop to 2nd point, etc. 1

(iv) Write a general-purpose version of your Coroutine which: [2]

- Moves the game object between each position in a List rather than just two positions.
- After arriving at each position, waits for the time defined in the Float, before continuing to the next Vector3 in the List.

Your Coroutine should use the following signature:

```
IEnumerator MoveBetween(List<Vector3> positions, float waitTime)
{

}
```

- Outer loop moving through List 1
- yield return new WaitForSeconds(waitTime) 0.5
- Cached use of a WaitForSeconds object 0.5

Some Useful Unity3D SDK Classes

GameObject: static methods

Instantiate() Destroy() DestroyImmediate() Find()

GameObject: methods

AddComponent() SendMessage() GetComponent() SetActive()

GameObject: data members

activeInHierarchy transform tag

MonoBehaviour: methods

Start() OnDestroy() Awake() Update()
FixedUpdate() LateUpdate() OnDisable() OnEnabled()
OnBecameInvisible() OnBecameVisible() OnCollisionEnter() OnCollisionExit()
OnCollisionStay() OnTriggerEnter() OnTriggerExit() OnTriggerStay()
SendMessage() BroadcastMessage() SendMessageUpwards() GetComponent()
GetComponentInChildren() GetComponentInParent() GetComponents() GetComponentsInChildren()
GetComponentsInParent() GetInstanceID() Invoke() StartCoroutine()

MonoBehaviour: data members

enabled gameObject transform name

Transform: methods

Rotate() Translate() TransformPoint() InverseTransformPoint()
LookAt() RotateAround() SetParent() TransformVector()
InverseTransformVector() TransformDirection() InverseTransformDirection()

Transform: data members

position localPosition rotation localRotation
lossyScale localScale parent right
up forward gameObject

Rigidbody: methods

AddForce() AddForceRelative() AddForceAtPosition() AddTorque()
AddRelativeTorque() MovePosition() MoveRotation()

Rigidbody: data members

drag angularDrag mass velocity
angularVelocity centerOfMass

Camera: methods

ScreenToWorldPoint() WorldToScreenPoint() ScreenToViewportPoint()
ViewportToScreenPoint() WorldToViewportPoint() ViewportToWorldPoint()
ViewportPointToRay() ScreenPointToRay()

Physics: static methods

Raycast() SphereCast() OverlapBox() BoxCast()

Input: static data members and methods

mousePosition GetKey() GetKeyDown() GetMouseButton()
GetMouseButtonDown()