NUI Galway
OÉ Gaillimh

Autumn Examinations 2019

Course Instance Code(s)	3BCT
Exam(s)	BSc (CS&IT)
Module Code(s)	CT3111
Module(s)	Next Generation Technologies
Paper No.	1
External Examiner(s)	Dr. Jacob Howe
Internal Examiner(s)	Prof. Michael Madden *Dr. Sam 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
Discipline(s)	Information Technology
Course Co-ordinator(s)	Dr. Des Chambers

Requirements:

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

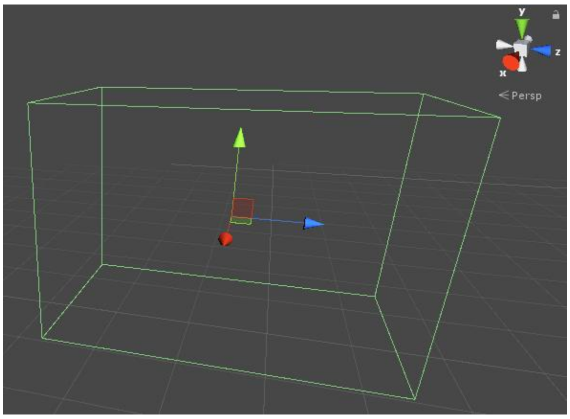
Q.1.

(i) Explain how Unity's MonoBehaviour class provides tight integration with the Game Loop. Refer to appropriate methods of the MonoBehaviour class in your answer. [6]

- Definition of the game loop and its essential steps [3]
- Identification of at least 2 methods of MonoBehaviour which match these steps [3]

(ii) What is a Coroutine in Unity, and how do Coroutines integrate with the Game Loop? [4]

- Definition of Coroutines as methods which may pause their operation in a number of ways [2]
- Explanation that this is not multi-threading, but rather it is registration of a method with the Unity engine such that the engine pauses and resumes the method as it performs the game loop, all within a single execution context [2]



(iii) The Game Object depicted has a Box Collider component, whose 'isTrigger' property is true. A script on the game object contains a reference to the Box Collider and to a prefab of a ball.

```
public BoxCollider bc;
public GameObject ball;

public IEnumerator SpawnBallsInBox(){

}
```

Write code for the SpawnBallsInBox() coroutine, so that it continually instantiates balls, at a rate of one ball every two seconds. The balls should be initialised to a random position somewhere inside the Box Collider. (Hint: use the 'bounds' property of the Box Collider, which has 'min' and 'max' properties, each of which are of type Vector3). [10]

- Infinite loop [1]
- WaitForSeconds() within the loop [2]
- Instantiation of ball object via use of the 'ball' class member [2]
- Use of the bounds property of the 'bc' class member to identify correct range of x,y,z values [3]

- Correct use of Random.Range to construct a Vector3 which is then used to position the ball [2]

Q.2.

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.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 gameobject 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]

Q.4.

(i) Bearing in mind that, in Unity's physics engine, gravity only operates along a fixed world vector, how could you simulate a moon orbiting a planet? Write Unity3D/C# code to achieve this, identifying the appropriate methods in which it should be written, as well as identifying the appropriate component(s) which have been added to the game objects. [10]

- Use of programmatic movement rather than via the physics engine [2]
- Periodically applying small movements (rotations) [2]
- Unity C# code written in the Update() or FixedUpdate() method [2]
- Unity C# code to perform the small movement, making use of Time.deltaTime

or Time.fixedDeltaTime [3]

- No components added to the objects [1]

(ii) Write Unity3D/C# code to accomplish the following:

- instantiate a gameobject at runtime, from a prefab [2]
- obtain a reference to the Rigidbody component which is assumed to be attached to it [2]
- attach a new Rigidbody to the gameobject, if it did not have one already [3]
- set the gameobject moving in a straight line using the physics engine [3]

- Each of these is one line of code, apart from the 3rd for which 1.5 marks are given for checking for a null result from getComponent<Rigidbody>() and 1.5 marks for .addComponent<RigidBody>()

Q.5.

Write technical notes on each of the following [5 x 4]

(i) How you would display (and update) a score on the screen while a game is being played, using the Unity GUI system.

- Create Text (or TextMesh) UI component in a Canvas [2]
- Obtain reference to this Text object at runtime, or via design-time referencing [2]
- Update the .text property when the score changes [1]

(ii) Garbage collection in Unity, including how to write low-garbage code.

- Definition of Garbage collection [1]
- Heap versus Stack and Objects versus simple values/structs [1]
- Problem with mark+sweep garbage collection in realtime systems [1]
- Low-garbage code via careful Object instantiation and avoiding of string manipulation [2]

(iii)Triggers and Colliders in Unity – how to use them and why they’re useful for games development.

- Definition of Colliders with reference to the Physics engine [2]
- Difference between Trigger and Colliders [1]
- How to create a collider and why it’s useful in games (at least one use-case) [1]
- How to create a trigger and why it’s useful in games (at least one use-case) [1]

(iv)Screen space, viewport space and world space in Unity.

- Definition of screen space [1]
- Definition of viewport space [1]

- Definition of world space [1]
- Some indication of how and why you would translate between these spaces [2]

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()