Semester 1 Examinations 2019-20

Course Instance Code(s)	3BCT
Exam(s)	BSc (CS&IT)
Module Code(s)	CT3536
Module(s)	Games Programming
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	5
Discipline(s)	Computer Science
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 ☐

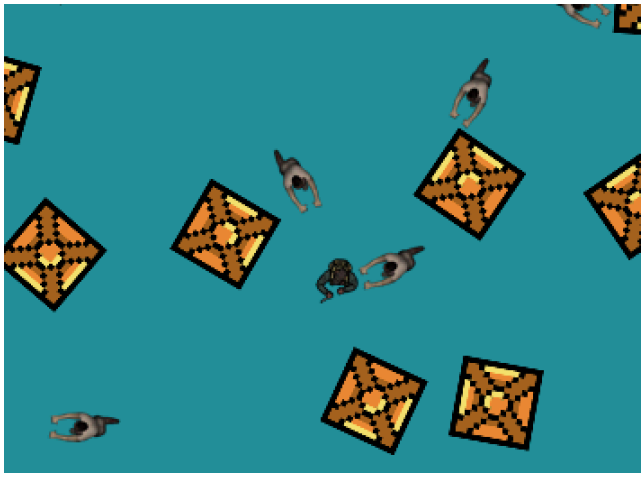
Q.1.

(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. [5]

(ii) What is a Coroutine in Unity, and how do Coroutines integrate with the Game Loop? [5]

(iii) In the two-dimensional game depicted in the image below, computer controlled 'zombies' chase the player. The zombie behaviour is controlled through a MonoBehaviour script called ZombieAI. Write Unity3D/C# code to rotate a zombie a little towards the player each frame, and indicate the appropriate method in the ZombieAI script into which this should be written. You can assume that zombies have access to the player game object via the static reference Player.activePlayer. Hint: use Vector3.RotateTowards() to calculate the result of rotating one vector towards another vector by a specified amount. [5]

(iv) Write additional code which will move the zombie in its forwards direction in each frame, but only if the zombie is facing straight at (or almost straight at) the player. Hint: use Vector3.Dot (vector dot product). [5]



Q.2.

Write technical notes on each of the following:

(i) How you would display (and update) a score on the screen while a game is being played, using the Unity GUI system. [5]
(ii) Garbage collection in Unity, including how to write low-garbage code. [5]
(iii) The use of State Machines to structure game logic. [5]
(iv) Screen space, viewport space and world space in Unity, including how to transform between them. [5]

Q.3.

(i) In the Unity3D game engine, define the terms: Collider, Trigger, and Rigidbody. Explain one typical use of each of these in a game. [6]

(ii) Under what circumstances does the OnCollisionEnter method get executed in a MonoBehaviour-derived script? Explain in general terms (no precise code required) how you could make use of the Collision.contacts argument received by OnCollisionEnter, to instantiate 'metallic spark' special effect prefabs at the points of contact, and have them correctly aligned so that the sparks move outwards from the surface of contact. [6]

(iii) 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 game objects are provided:

- rotate a game object 5 degrees around the world's y axis [2]
- move a game object 7 units directly towards another game object [3]
- move a game object 10 units forward in whatever direction it is facing [3]

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 planet orbiting a sun? 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. [7]

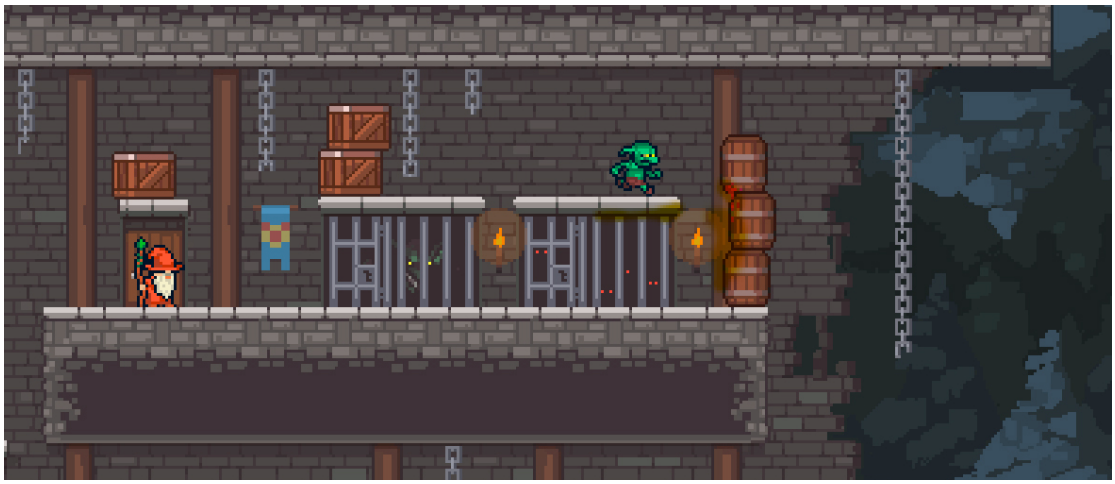
(ii) Extend your code so that, in addition to a planet orbiting a sun, there is also a moon orbiting the planet. [5]

(iii) Write a single Unity (MonoBehaviour) C# script which is suitable for attaching to any game object that you wish to orbit around another game object. The script should offer inspector settings (i.e. public variables and object references) for defining the object being rotated around, as well as the axis of rotation and the speed of rotation. [8]

Q.5.

(i) In games development, what does the term 'raycast' mean, as supported by various static methods of the Unity3D SDK's Physics class (and Physics2D class)? Explain, with illustrative C# code, how you could use a raycast to determine whether a character in a game is standing on something. [10]

(ii) In the two-dimensional game depicted below, the game characters have Rigidbody2D and Collider2D components, are moved left and right using the left/right arrow keys, and can jump upwards when the up arrow key is pressed – but only if they are standing on something. Their movement is controlled by the physics engine. Write suitable Unity3D/C# code to implement these movement behaviours for a character, indicating which methods the code is written in. [10]



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()			