National University of Computer and Emerging Sciences

Lahore Campus

Game Design & Development (CS4046)

Date: April 10, 2025

Course Instructor(s)

Ms. Saba Ghani

Sessional- II Exam

Total Time (Hrs): 1

Total Marks: 50

Total Questions: 3

_____ _____ _____

Roll No Section Student Signature

Cutting and overwriting is not allowed. Marks shall be reduced otherwise.
Write your answers in the order on the answer sheet. Attach question paper with answer sheet.
Fill out the table provided for MCQs. No MCQ will be evaluated otherwise.

CLO # 3: Implement various game mechanics, such as character movement, controls, physics, data persistence, and player feedback.

Q1: Select the best possible option from MCQs and write option name in this table [15]

1. c	**2. c**	**3. b**	**4. b**	**5. c**	**6. b**	**7. c**	**8. b**
9. b	**10. c**	**11. c**	**12.c**	**13. b**	**14. a**	**15. b**	

1. What is the primary difference between Update() and FixedUpdate() in Unity?

a) Update() is frame-rate dependent; FixedUpdate() is called every second

b) Update() handles physics; FixedUpdate() handles animations

c) Update() is frame-rate dependent; FixedUpdate() is used for physics calculations

d) FixedUpdate() is faster than Update()

2. What is the function of Raycast in Unity?

a) Detects UI clicks

b) Detects light direction

c) Detects objects in a certain direction

d) Detects animation speed

3. What is the purpose of using Tags in Unity?

a) To apply physics

b) To filter objects in collision and scripting logic

c) To change object color

d) To store animations

4. What does Random.Range(-5f, 5f) return?

a) A constant float value

b) A random float between -5 and 5 (inclusive)

c) A random float between -5 and 5 (exclusive)

d) A random integer between -5 and 5

5. Which Unity component ensures a UI element scales properly across devices?

a) Canvas Renderer

b) Layout Group

c) Anchor

d) Event System

6. What is the main difference between a BoxCollider and a MeshCollider?

a) MeshCollider is for UI; BoxCollider is for 3D models
b) BoxCollider is more optimized, MeshCollider is based on the object's shape
c) MeshCollider works only on 2D objects
d) BoxCollider can't detect collisions

7. Which of the following best describes the behavior of a Rigidbody component when using MovePosition inside FixedUpdate() instead of applying force?

a) It behaves the same as applying force and interacts with collisions identically.
b) It teleports the object, ignoring physics and collisions.
c) It ensures smooth, physics-based movement that respects collision detection.
d) It only affects the object's velocity but does not change position directly.

8. You implemented UI buttons using Unity's EventSystem and Canvas. However, the buttons don't respond to clicks during gameplay. What could be a possible reason?

a) The Canvas Render Mode is set to World Space.
b) There is no EventSystem in the scene.
c) The button is inside a Scroll View.
d) The Text component is overlapping the button.

9. A developer wants to create a touch-based character movement system that allows flick gestures to apply a force in the direction of the swipe. Which approach will likely give the most accurate and scalable input detection across platforms?

a) Using Input.GetMouseButtonDown and GetMouseButtonUp
b) Checking Input.touchCount and using TouchPhase states
c) Using Raycast on every frame to check input position
d) Adding an EventTrigger component on the character

10. Which of the following use cases best demonstrates the need for LateUpdate over Update or FixedUpdate?

a) Applying physics-based jump forces to a Rigidbody
b) Moving an object based on keyboard input
c) Smoothly following a camera target after all movements are applied
d) Checking for ground collision using Raycast

11. What does the deltaTime variable ensure when used in character movement?

a) The character only moves during the FixedUpdate loop.
b) Movement is dependent on screen resolution.
c) Movement is consistent across different frame rates.
d) The player speed remains constant only in physics-based games.

12. In Unity’s new Input System, what is the primary purpose of the InputAction asset?

a) To store all player preferences like volume and screen resolution
b) To track only button presses for menus
c) To define a centralized map of all input actions and bindings
d) To disable traditional input handling like Input.GetAxis

13. You want to display a responsive health bar UI that works across various screen sizes. Which setup is most appropriate?

a) Use World Space Canvas and place the bar manually
b) Set anchors and pivot correctly in Screen Space - Overlay Canvas
c) Use an Image with a fixed pixel size only
d) Scale the health bar manually with screen resolution

14. A prefab is instantiated every few seconds at a random horizontal position. What code snippet would correctly calculate that random position?

a) new Vector3(Random.Range(-5f, 5f), 0, 0)
b) new Vector3(0, Random.Range(5f, -5f), 0)
c) Vector3.up * Random.Range(-5, 5)
d) new Vector3(0, 0, Random.Range(5f))

15. What will happen if a UI element has anchors set to center but the pivot is set to (0, 0)?

a) The element stretches beyond the canvas
b) The element moves unexpectedly when screen resolution changes
c) The element rotates from its center
d) The element cannot be resized manually

Do not write below this line.

Q2: Write short answers of the following, make sure to justify the marks distribution while answering the questions. (05 x 3 = 15 Marks)

Q2.1. What is a prefab in Unity, and why is it useful?

Ans: A Prefab in Unity is a reusable GameObject template that stores a configured GameObject with components, settings, and child objects. It's useful because it allows for consistent spawning of objects at runtime and simplifies scene management, especially in complex games with repeated elements (e.g., enemies, bullets, collectibles).

Q2.2. Why do we use deltaTime in movement scripts?

Ans: deltaTime represents the time passed between the current and previous frame. It's used in movement scripts to ensure smooth, frame-rate-independent motion. For example, transform.Translate(Vector3.forward * speed * Time.deltaTime) ensures consistent speed regardless of FPS.

Q2.3. How can you detect which GameObject was hit using a raycast?

Ans: To detect which GameObject was hit using a Raycast, you use:

```
RaycastHit hit;
if (Physics.Raycast(ray, out hit)) {
    GameObject hitObject = hit.collider.gameObject;
}
```

This hitObject is the GameObject that was intersected by the ray.

Q2.4. How do you prevent a Rigidbody object from rotating in Unity?

Ans: To prevent a Rigidbody from rotating, you can freeze its rotation using:rigidbody.freezeRotation = true;Alternatively, in the Rigidbody component in the Inspector, you can check the "Freeze Rotation" constraints.

Q2.5. What is the difference between OnEnable() and Awake() in Unity?

Ans: Awake() is called when the script instance is loaded, even before the GameObject is enabled. OnEnable() is called each time the GameObject becomes active. Use Awake() for initialization that should occur once, and OnEnable() for setup needed whenever the object is re-enabled.

Q3: Write C# scripts for the following scenarios: (04 x 05 = 20 Marks)

Scenario 1: Player Health System

You are tasked with implementing a health system for a player character in Unity. The player starts with 100 health points. When the player takes damage, their health should decrease by a specified amount (damage =5), but the health should never fall below zero. Create a C# script that implements the following (Health points and messages shall be displayed using UI elements):

- A method to reduce health when damage is taken.
- A method to check if the player's health is zero or below and trigger a "Game Over" event if so.

Ans:

```
using UnityEngine;
using UnityEngine.UI;public class PlayerHealth : MonoBehaviour {
   public int health = 100;
   public Text healthText;
   public Text gameOverText;    void Start() {
      UpdateUI();
      gameOverText.gameObject.SetActive(false);
   }    public void TakeDamage(int damage) {
      health = Mathf.Max(0, health - damage);
      UpdateUI();
      if (health <= 0) {
         GameOver();
      }
   }    void UpdateUI() {
      healthText.text = "Health: " + health;
   }    void GameOver() {
      gameOverText.gameObject.SetActive(true);
      gameOverText.text = "Game Over";
   }
}
```

Scenario 2: Door Opening Mechanism
You need to assume a door that opens when the player is close enough and presses a button (e.g., "E"). Write a C# script that:

- Detects when the player is within a certain distance from the door.
- Opens the door when the "E" key is pressed.

Ans:

```
using UnityEngine;public class DoorOpener : MonoBehaviour {
   public Transform player;
   public float openDistance = 3f;
   private bool isOpen = false;    void Update() {
      if (!isOpen && Vector3.Distance(transform.position, player.position) < openDistance) {
         if (Input.GetKeyDown(KeyCode.E)) {
            OpenDoor();
         }
      }
   }    void OpenDoor() {
      // Example animation/rotation
      transform.Rotate(0, 90, 0);
      isOpen = true;
   }
}
```

Scenario 3: Countdown Timer
You are building a game where the player has a limited amount of time to complete a level. The level has a timer that starts at 60 seconds and counts down. When the timer reaches zero, the game should display a "Game Over" message and stop the level. Write a C# script that:

- Starts a countdown timer at 60 seconds when the level begins.
- Updates the timer every second and displays the remaining time on the screen.
- Ends the game when the timer reaches zero, displaying a "Game Over" message and pausing the game.

Ans:

```
using UnityEngine;
using UnityEngine.UI;

public class CountdownTimer : MonoBehaviour
{
   public float startingTime = 60f;      // Total time in seconds
   private float currentTime;           // Time left
   public Text timerText;              // UI Text to display time
   public Text gameOverText;            // UI Text to show Game Over

   private float timerInterval = 1f;     // Time between updates
   private float timer;                // Keeps track of 1 second intervals
```

```
    private bool isGameOver = false;

    void Start()
    {
        currentTime = startingTime;
        timer = timerInterval;
        gameOverText.gameObject.SetActive(false); // Hide Game Over at start
        UpdateTimerUI();                          // Show initial time
    }

    void Update()
    {
        if (isGameOver) return;

        timer -= Time.deltaTime;

        if (timer <= 0f)
        {
            currentTime--;
            timer = timerInterval;
            UpdateTimerUI();

            if (currentTime <= 0)
            {
                EndGame();
            }
        }
    }

    void UpdateTimerUI()
    {
        timerText.text = "Time: " + currentTime.ToString("0");
    }

    void EndGame()
    {
        isGameOver = true;
        gameOverText.gameObject.SetActive(true);
        gameOverText.text = "Game Over";
        Time.timeScale = 0f; // Pause the game
    }
}
```

Scenario 4: Swipe Gesture for Character Movement

You are developing a mobile game where the player controls a character's movement by swiping on the screen. The character should move in the direction of the swipe. Write a C# script that:

- Detect a swipe gesture on the screen.

- Determines the direction of the swipe (up, down, left, or right).
- Moves the character accordingly based on the swipe direction.

Ans:

```
using UnityEngine;

public class SwipeMovement : MonoBehaviour {
   private Vector2 startTouchPos, endTouchPos;
   public float moveDistance = 2f;

   void Update() {
      if (Input.touchCount > 0) {
         Touch touch = Input.GetTouch(0);

         if (touch.phase == TouchPhase.Began) {
            startTouchPos = touch.position;
         }
         else if (touch.phase == TouchPhase.Ended) {
            endTouchPos = touch.position;
            Vector2 swipe = endTouchPos - startTouchPos;
            swipe.Normalize();

            if (Mathf.Abs(swipe.x) > Mathf.Abs(swipe.y)) {
               if (swipe.x > 0) Move(Vector3.right);
               else Move(Vector3.left);
            } else {
               if (swipe.y > 0) Move(Vector3.forward);
               else Move(Vector3.back);
            }
         }
      }
   }

   void Move(Vector3 direction) {
      transform.Translate(direction * moveDistance);
   }
}
```