National University of Computer and Emerging Sciences

Lahore Campus

Game Design & Development (CS4046)

Date: June 05, 2025

Course Instructor(s)

Ms. Saba Ghani

Final Exam

Total Time (Hrs):	**2.5**
Total Marks:	**90**
Total Questions:	**3**

_____ _____ _____

Roll No Section Student Signature

CLO # 1: Understand the gaming industry and product lifecycle of different types of games.
CLO # 2: Understand fundamental principles of game design, including gameplay mechanics, player motivation, player narrative and storytelling, pacing, and balance.
CLO # 3: Implement various game mechanics, such as character movement, controls, physics, data persistence, and player feedback

Cutting and overwriting is not allowed.
Write your answers in the order on the answer sheet. 05 Marks shall be reduced otherwise.

Do not write below this line.

Q1 (CLO-1): Select the best possible option from MCQs (30 Marks)

1. What is the primary purpose of a game development postmortem?
A) To celebrate the game's success without criticism.
B) To analyze what went right and wrong for future improvements.
C) To assign blame for project failures.
D) To replace the need for playtesting.

2. Which of the following correctly orders the layers of game technology from foundational to application-specific?
A) Hardware → Game-specific code → Middleware → Operating System
B) Hardware → Operating System → Middleware → Game-specific code
C) Middleware → Hardware → Game-specific code → Operating System
D) Operating System → Game-specific code → Middleware → Hardware

3. In the core gameplay loop of *Clash of Clans*, which action directly contributes to *progression*?
A) Collecting resources like Gold and Elixir.
B) Tapping the screen to view the base.
C) Watching replays of other players' attacks.
D) Sending chat messages to clan members.

4. Which characteristic is *most* associated with hyper-casual games?
A) Complex resource management systems.
B) Long play sessions requiring deep strategy.

C) Single core mechanic and snackable content.
D) High retention rates over months.

5. What is the primary purpose of A/B testing in game design?
A) To replace player feedback entirely.
B) To compare two versions of a feature to determine which performs better.
C) To ensure all players receive identical experiences.
D) To eliminate the need for analytics tools.

6. In Unity, if a GameObject's position is (10, 5, 0) and it moves by a vector (2, -3, 4), what is its new position?
A) (12, 2, 4)
B) (8, 8, -4)
C) (10, 5, 0)
D) (12, -15, 0)

7. Which camera projection type is *best* suited for a 2D side-scrolling game where objects must maintain consistent size regardless of distance?
A) Perspective projection.
B) Orthographic projection.
C) Isometric projection.
D) Fisheye projection.

8. What is the key advantage of using prefabs in Unity?
A) Prefabs cannot be modified once created.
B) Prefabs allow reusable templates with synchronized edits across instances.
C) Prefabs are only useful for 2D games.
D) Prefabs reduce the need for game logic scripting.

9. Which Rigidbody property would you adjust to make a GameObject simulate a bouncy rubber ball?
A) Mass.
B) Drag.
C) Physics Material
D) Is Kinematic.

10. Which input method is *most* suitable for physics-based movement in Unity?
A) Input.GetKey() in the Update() function.
B) Input.GetAxis() in the FixedUpdate() function.
C) Mouse clicks in the OnGUI() function.
D) Input.GetButtonDown() in the Start() function.

11. In the DPE framework, which component focuses on *player emotions and immersion*?
A) Design.
B) Play.
C) Experience.
D) Development.

12. What is the primary role of *boundaries* in a game world?
A) To limit the player's imagination.
B) To define the playable area and enforce game rules.
C) To eliminate the need for storytelling.
D) To reduce the workload for artists.

13. Which era of game development introduced *voice acting and motion capture* for characters?
A) Primordial Ooze (Pre-1985).
B) Early Ages (1985–1994).
C) Middle Ages (1995–2001).
D) Modern Ages (2002+).

14. Which monetization model is *most* likely used by a hyper-casual game?
A) Premium (pay-to-play).
B) Subscription-based.
C) In-app advertising.
D) Blockchain (Play-to-Earn).

15. Which Unity method is *best* for initializing variables that depend on other GameObjects?
A) Awake().
B) Start().
C) Update().
D) OnDestroy().

16. What is the primary advantage of using the *Action-based* input handling in Unity's New Input System over the legacy Input.GetAxis() method?
A) It requires fewer lines of code for basic movement.
B) It allows customizable control schemes and runtime remapping for multiple devices.
C) It only works with keyboard and mouse inputs.
D) It eliminates the need for scripting.

17. In a mobile game, how would you detect a *swipe gesture* using Input.touch?
A) Compare TouchPhase.Moved with TouchPhase.Stationary.
B) Store touch.position in TouchPhase.Began and compare it with touch.position in TouchPhase.Ended.
C) Use Input.mousePosition for touch simulations.
D) Ignore TouchPhase.Canceled to avoid false swipes.

18. Which Canvas Scaler mode ensures UI elements *maintain consistent physical size* (e.g., millimeters) across devices with different DPIs?
A) Constant Pixel Size.
B) Scale With Screen Size.
C) Constant Physical Size.
D) World Space.

19. Why is a *Sorting Layer* critical for 2D games in Unity?
A) It defines the collision boundaries for sprites.
B) It controls the render order of sprites independently of their Z-position.
C) It replaces the need for Rigidbody2D components.
D) It automatically generates sprite animations.

20. What is the purpose of the *Any State* in an Animator Controller?
A) To blend animations smoothly during transitions.
B) To trigger a transition to a specific state *regardless* of the current state (e.g., a death animation).
C) To pause all animations temporarily.
D) To synchronize animations with audio clips.

21. Which statement about PlayerPrefs is *false*?
A) It can store integers, floats, and strings.
B) Data is automatically encrypted for security.

C) Values persist across game sessions.
D) PlayerPrefs.Save() forces immediate disk writes.

22. Why use an *event* instead of a public delegate for score updates in a ScoreManager class?
A) Events allow external classes to invoke the delegate directly.
B) Events restrict invocation to the declaring class, ensuring safer encapsulation.
C) Delegates cannot handle method references.
D) Events are slower but more flexible.

23. What is the key difference between Invoke() and a *Coroutine*?
A) Coroutines cannot use WaitForSeconds.
B) Invoke() supports dynamic pauses and resumptions.
C) Coroutines allow complex sequences with yields (e.g., waits, frames).
D) Invoke() is only for UI animations.

24. When is the Singleton pattern *not* recommended?
A) For global access to a GameManager.
B) When multiple instances of a class might be needed later.
C) For managing audio across scenes.
D) When tight coupling simplifies code readability.

25. How does the Observer pattern *decouple* systems like a ScoreManager and UIManager?
A) The ScoreManager directly calls UIManager.UpdateScore().
B) The UIManager subscribes to a ScoreChanged event and reacts independently.
C) Both classes must inherit from the same base class.
D) It uses a Singleton to mediate communication.

26. Which game mechanic would *best* leverage the Command pattern?
A) A static menu with fixed buttons.
B) A replay system that records and re-executes player actions.
C) A particle effect triggered by a collision.
D) A singleton managing scene transitions.

27. Why use a Dictionary<string, int> over a List<string> for tracking item quantities in an inventory?
A) Lists support faster iteration through all elements.
B) Dictionaries allow O(1) lookups by key (e.g., item name).
C) Lists automatically sort items alphabetically.
D) Dictionaries cannot store integers.

28. You're designing a multiplayer game where:
- **Player 1 uses a keyboard/mouse, Player 2 uses a gamepad.**
- **Both need to remap controls in-game.**

How would you implement this with Unity's New Input System?
A) Use separate InputActionAssets for each player and enable runtime rebinding
B) Hardcode key/button checks in Update() for both players.

C) Use PlayerPrefs to store control schemes.
D) Create a Singleton InputManager with fixed keybindings.

29. For the OnTriggerEnter function to invoke successfully on a collision between two objects, at least one must have which two components?

A) A Collider and a MeshRenderer
B) A Collider and a MeshFilter
C) A Rigidbody and a Collider
D) A MeshRenderer and a MeshFilter

30. Which of the following is NOT a valid method for moving a GameObject in Unity3D using C#?

A) transform.Translate()
B) Rigidbody.AddForce()
C) GameObject.MoveTo()
D) CharacterController.Move()

MCQs Solution:

1	**B**	16	**B**
2	**B**	17	**B**
3	**A**	18	**C**
4	**C**	19	**B**
5	**B**	20	**B**
6	**A**	21	**B**
7	**B**	22	**B**
8	**B**	23	**C**
9	**C**	24	**B**
10	**B**	25	**B**
11	**C**	26	**B**
12	**B**	27	**B**
13	**C**	28	**A**
14	**C**	29	**C**
15	**B**	30	**C**

Q2 (CLO-2): Write short answers of the following: **(30 Marks)**

1. Define model layer of graphics display model in detail. (2)

Internal representation of graphics. The way computers store, process, and render graphical data internally before displaying it on a screen. Types are **Vector Graphics, Bitmap (aka, pixel map)/ Raster Graphics, 3D graphics and textures.**

2. Write four principles of data driven game design (2)

P1: Player behavior analysis: Identifying where players frequently quit or struggle can help adjust difficulty curves.

P2: A/B Testing (Split Testing): A/B testing allows developers to test different versions
P3: Engagement metrics
P4: Performance optimization
P5: Dynamic Content Adjustments: Some games use real-time data to adjust mechanics dynamically

3. Differentiate between player conversion rate and churn rate (2)

Player Conversion Rate: The percentage of players who complete a desired action (e.g., make a purchase, register, or subscribe).
Churn Rate: The percentage of players who stop playing the game over a given time period.

4. How many lighting modes are available in Unity? Categorize all light types as per lighting mode. (2)

There are 2 lighting modes (**Real time, Baked**) in Unity. Some versions have 3rd mode (mixed) as well.
Directional, point, spot (**real time + baked**), area light (**baked only**).

5. Differentiate between continuous collision detection and continuous dynamic collision detection (2)

Continuous: Collision detection of **fast-moving objects** colliding with **static objects**.
Continuous Dynamic: Collision detection of **fast-moving objects** colliding with **other moving objects**.

6. Explain hierarchy of User Interface in Unity (2)

1.Canvas: The top-level container for all UI elements**.**
2. Panel: Acts as a container or background for organizing groups of UI elements.
3. UI Elements (children of Panel or Canvas): Includes components like Text, Image, Button, Toggle, Slider, etc.

7. Map the game "Minecraft" on DPE framework. (3)

Design: Sandbox mechanics, survival mode, creative building tools.
Play: Players mine resources, craft tools, build structures, and survive against enemies**.**
Experience: A limitless sense of creativity, self-expression, and progression.

8. Differentiate between three scale modes of canvas scaler (3)

Constant Pixel Size: Makes UI elements retain the same pixel size, regardless of the screen size.
Scale With Screen Size: Sizes and positions UI elements according to a referenced resolution.
Constant Physical Size: Positions of the UI elements are specified in physical units such as millimeters or points.

9. Explain what anchors are in Unity UI with example and how they affect UI elements when the screen size changes. (3)

In Unity UI, **anchors** define the position and resizing behavior of a UI element **relative to its parent (usually the Canvas or a Panel)**.
Example:

- If you anchor a Button to the **bottom-right corner**, then resize the screen:
  - The button will **stay "anchored" to that corner**.
  - Its position relative to the bottom and right edges will remain constant.

Effect on Screen Size Changes: Anchors help make your UI responsive across different screen sizes by controlling how UI elements move or scale with their parent.

10. You're designing the Animator Controller for a player character in a 3D adventure game. The character can Idle, Walk, Run, Jump, and also instantly perform a "Hurt" animation whenever hit by an enemy, regardless of what animation is currently playing. Explain how you would use **Animator Controller** to play the "Hurt" animation instantly, and how you would return to normal gameplay animations afterward. (3)

This complete scenario can be optimally implemented in 3 steps:
Step 1: **Create a Trigger Parameter:**

- In the Animator Controller, add a **Trigger** parameter named Hurt.

Step 2: **Use "Any State":**

- Create a transition from **Any State** to the **Hurt** animation state.
- Set the **condition** to Hurt trigger.

Step 3: **Transition Back to Gameplay:**

- After the Hurt animation, add transitions back to Idle or other gameplay animations.
- Use a **bool**, **animation end event**, or **Exit Time** to smoothly return to normal flow.

11. Write at least three differences between events and delegates (3)

Aspect	Delegate	Event
Definition	A delegate is a **type** that holds references to methods.	An event is a **wrapper around a delegate** to restrict external access.
Who Can Invoke	Can be invoked by **any class** that has access to it.	Can only be invoked from **within the class** where it is declared.
Encapsulation Level	Less controlled — exposes **add, remove, and invoke** functionality.	More controlled — allows only **subscription (+=)** and **unsubscription (-=)** from outside.

12. Explain core components of command pattern (3)

Command Interface – Declares Execute() (and maybe Undo())
ConcreteCommand – Implements the command
Receiver – Does the actual work (e.g., Player)
Invoker – Triggers the command

Q3 (CLO-3): Write C# scripts for the following: **(30 Marks)**

1. Code to hide a 3d object (1)

1. gameObject.SetActive(false);
2. GetComponent<MeshRenderer>().enabled = false;

2. Write “GetComponent” command syntax (1)

Rigidbody rb = GetComponent<Rigidbody>();
Returns a component of the specified type attached to the same GameObject.

3. Code to disable the collider component of a GameObject using GetComponent (1)

1. GetComponent<Collider>().enabled = false;
2. GetComponent<BoxCollider>().enabled = false;

4. Code to instantiate a prefab named "EnemyPrefab" at the position (0, 0, 0) in Unity (1)

1. Instantiate(EnemyPrefab, new Vector3(0, 0, 0), Quaternion.identity);
2. Instantiate(EnemyPrefab, Vector3.zero, EnemyPrefab.transform.rotation);

5. Code to move a GameObject forward using Transform.Translate at a speed of 5 units/sec. (1)

1. transform.Translate(Vector3.forward * 5f * Time.deltaTime);
2. rb.MovePosition(rb.position + transform.forward * 5f * Time.fixedDeltaTime);

6. Code to rotate a 3D object clockwise around the Y-axis by 90 degrees in Unity. (2)

1. transform.Rotate(0f, 90f, 0f);
2. transform.rotation = Quaternion.Euler(0f, 90f, 0f);
3. transform.Rotate(Vector3.up * 90f);

7. Check if “UserName” key exists in playerprefs, if yes then display the value on console otherwise save your name with specified key. (2)

```
if (PlayerPrefs.HasKey("UserName")) {
      Debug.Log("User Name: " + PlayerPrefs.GetString("UserName")); }
   else {
      PlayerPrefs.SetString("UserName", "Saba Ghani");
      Debug.Log("User Name saved."); }
```

8. Write a coroutine that waits for 5 seconds before printing “Game Start!” (2)

```
IEnumerator StartGameCoroutine()
{
   yield return new WaitForSeconds(5f);
   Debug.Log("Game Start!");
}
```

9. Code to Detect Collision with a Tagged Object ("Coin") and Destroy It (2)

Version 1. void OnCollisionEnter(Collision collision) {
   if (collision.gameObject.CompareTag("Coin"))

```
{
   Destroy(collision.gameObject);  } }
```

Version 2. void OnCollisionEnter(Collision collision)

```
{
   if (collision.gameObject.tag == "Coin")
   {
      Destroy(collision.gameObject);  } }
```

Version 3. void OnTriggerEnter(Collider other)

```
{
   if (other.CompareTag("Coin"))
   {
      Destroy(other.gameObject);    } }
```

10. Code to Move a 3D Object to Touch Position (World Space) (2)

```
if (Input.touchCount > 0)
{
    Touch touch = Input.GetTouch(0);
    Vector3 touchPosition = Camera.main.ScreenToWorldPoint(touch.position);
    touchPosition.z = 0;
    transform.position = touchPosition;
}
```

11. Code to Implement Pinch-to-Zoom (Scale a GameObject) (3)

```
void Update(){
   if (Input.touchCount == 2)
   {
      var t0 = Input.GetTouch(0);
      var t1 = Input.GetTouch(1);

      var prevDist = (t0.position - t0.deltaPosition - (t1.position - t1.deltaPosition)).magnitude;
      var currDist = (t0.position - t1.position).magnitude;

      var scaleChange = (currDist - prevDist) * 0.01f;
      transform.localScale += Vector3.one * scaleChange; } }
```

12. Blend between **Walk**, **Run**, and **Sprint** animations using a float parameter speed based on the character's movement velocity. (3)

```
public Animator animator;
public Rigidbody rb;
void Update(){
   float currentSpeed = rb.velocity.magnitude;
   animator.SetFloat("Speed", currentSpeed); }
```

13. On login, save the player's email in PlayerPrefs. Next time the game starts, check and auto-fill it in the email field. (3)

```
public InputField emailInput;
   void Start()  {
      if (PlayerPrefs.HasKey("PlayerEmail"))  // Auto-fill email if saved previously
      {
         emailInput.text = PlayerPrefs.GetString("PlayerEmail"); }   }
   public void OnLoginButtonClicked() {
      string email = emailInput.text;
      PlayerPrefs.SetString("PlayerEmail", email);
      PlayerPrefs.Save(); // Optional but good practice  } }
```

14. You have three sprite GameObjects: **Player**, **Enemy**, and **Background**. Assign appropriate sorting layers and orders so the player appears **above** the enemy, and both appear **above** the background. Write the code to set their sorting layers and order in layer at runtime. (03)

```
public SpriteRenderer player, enemy, background;
void Start() {
      // Background behind everything
      background.GetComponent<SpriteRenderer>().sortingOrder = 0;

      // Enemy above background
      enemy.GetComponent<SpriteRenderer>().sortingOrder = 1;

      // Player above enemy
      player.GetComponent<SpriteRenderer>().sortingOrder = 2;  }
```

15. Write a Singleton class LevelManager that initializes a public variable currentLevel = 1, and contains a method LoadNextLevel() that increments the level. (3)

```
public class LevelManager : MonoBehaviour {
   public static LevelManager Instance;
   public int currentLevel = 1;

   void Awake() {
      if (Instance == null) Instance = this;
      else Destroy(gameObject); }

   public void LoadNextLevel() {
      currentLevel++;
      Debug.Log("Level: " + currentLevel); } }
```