Introduction to AR VR - Test-I Solutions

Date: 7-10-2026

Time: 6:00 PM - 7:00 PM

Marks: 10

Instructions

- Figures in brackets on the right-hand side indicate full marks.
- Assume suitable data if necessary.
- Question 1 is compulsory.
- Answer any 2 from the remaining questions.

SET A Solutions

Q1a: Player Cube Moving Forward and Backward

CO-4, SO-2, B.L.-3, Marks: [1×2]

Steps:

1. Create a new 3D project named "ARVRTest_SetA_Q1a" in Unity Hub.
2. Add a Plane named "Ground", set Position to (0, 0, 0), Scale to (10, 1, 10), add Box Collider.
3. Add a Cube named "Player", set Position to (0, 0.5, 0), add Rigidbody and Box Collider.
4. Create a script named "PlayerMovement" in a "Scripts" folder.
5. Attach "PlayerMovement" to the Player GameObject.
6. Test with W/S for movement, ESC to stop in Editor or quit in build.
7. Build via File > Build Settings > Add Open Scenes > Build.

Code:

```csharp
```

```
using UnityEngine;

public class PlayerMovement : MonoBehaviour
{
    public float moveSpeed = 5f; // Speed of movement, adjustable in Inspector

    Rigidbody rb;

    void Awake()
    {
        rb = GetComponent<Rigidbody>(); // Get the Rigidbody component for physics
    }

    void Update()
    {
        float verticalInput = Input.GetAxisRaw("Vertical"); // Get W/S or Up/Down input (-1 for back, 1 for forward)
        Vector3 movement = new Vector3(0, 0, verticalInput) * moveSpeed * Time.deltaTime; // Frame-rate independent moven
        transform.Translate(movement, Space.Self); // Move forward/backward relative to cube's orientation

        if (Input.GetKeyDown(KeyCode.Escape)) // Check if ESC is pressed
        {
#if UNITY_EDITOR
            UnityEditor.EditorApplication.isPlaying = false; // Stop play mode in Editor
#else
            Application.Quit(); // Quit the application in build
#endif
        }
    }
}
```

Submission Requirements:

- Script screenshot showing movement + ESC logic.
- Hierarchy screenshot showing proper naming (e.g., "Player", "Ground").
- Game window screenshot showing cube movement.

Q1b: TextMeshPro Element Displaying "Ready to Move"

CO-4, SO-3, B.L.-3, Marks: [1×2]

Steps:

1. In Hierarchy, right-click > UI > Canvas, name it "UICanvas".

2. Add Text - TextMeshPro under UICanvas, name it "MovementText", set Position to (0, 2, 0) relative to Player.
3. In Inspector, set Text to "Ready to Move", Font Size to 24, Alignment to Center.
4. Create a script named "TextDisplay" in the Scripts folder.
5. Attach "TextDisplay" to UICanvas or MovementText.
6. Test to ensure text appears above Player.

Code:

```csharp
using UnityEngine;
using TMPro;

public class TextDisplay : MonoBehaviour
{
    public TextMeshProUGUI textDisplay; // Assign MovementText in Inspector

    void Start()
    {
        if (textDisplay != null) // Ensure text object is assigned
        {
            textDisplay.text = "Ready to Move"; // Set initial text
        }
    }
}
```

Submission Requirements:

- Hierarchy screenshot showing TMP above player.
- Game window screenshot showing visible text.
- Inspector screenshot showing TMP properties.

Q2: Obstacle Disappears on Collision

CO-3, SO-6, B.L.-5, Marks: [3]

Steps:

1. Add a Cube named "Obstacle", set Position to (2, 0.5, 0), add Box Collider.
2. Attach Rigidbody to Obstacle, uncheck "Use Gravity" if static.

3. Add TextMeshPro under UICanvas, name it "StatusText", position at top-center.
4. Create a script named "ObstacleCollision".
5. Attach "ObstacleCollision" to Obstacle, assign "StatusText".
6. Test by pushing Player into Obstacle.

Code:

```csharp
using UnityEngine;
using TMPro;

public class ObstacleCollision : MonoBehaviour
{
    public TextMeshProUGUI statusText; // Assign StatusText in Inspector

    void OnCollisionEnter(Collision collision)
    {
        if (collision.gameObject.CompareTag("Player")) // Check if Player collides
        {
            gameObject.SetActive(false); // Disable the obstacle (makes it disappear)
            if (statusText != null)
            {
                statusText.text = "Obstacle Cleared!"; // Update TMP text
            }
        }
    }
}
```

Submission Requirements:

- Script screenshot showing collision logic.
- Inspector screenshot of obstacle.
- Game window screenshot before and after collision.
- Hierarchy screenshot with clean naming.

Q3: Trigger Zone with Falling Cube

CO-3, SO-1, B.L.-4, Marks: [3]

Steps:

1. Create an Empty named "TriggerZone", add child Cube "TriggerArea", set Position to (0, 0, 5), scale (2, 1, 2), check "Is Trigger".
2. Add a Cube named "FallingCube", set Position to (0, 5, 5), add Rigidbody (uncheck initially).
3. Add TextMeshPro to UICanvas, name it "GravityText", position at top-center.
4. Create a script named "TriggerGravity".
5. Attach "TriggerGravity" to TriggerZone, assign "FallingCube" and "GravityText".
6. Test by moving Player into TriggerArea.

Code:

csharp

```
using UnityEngine;
using TMPro;

public class TriggerGravity : MonoBehaviour
{
    public GameObject fallingCube; // Assign FallingCube in Inspector
    public TextMeshProUGUI gravityText; // Assign GravityText in Inspector

    void OnTriggerEnter(Collider other)
    {
        if (other.CompareTag("Player")) // Check if Player enters
        {
            if (fallingCube != null)
            {
                Rigidbody rb = fallingCube.GetComponent<Rigidbody>();
                if (rb != null)
                {
                    rb.useGravity = true; // Enable gravity to make it fall
                }
            }
            if (gravityText != null)
            {
                gravityText.text = "Gravity Enabled"; // Display message
            }
        }
    }
}
```

Submission Requirements:

- Script screenshot showing trigger handling.

- Hierarchy showing trigger zone and affected object.
- Game window showing cube falling and TMP text.

Q4: Rotating Obstacle that Stops on Collision

CO-4, SO-2, B.L.-6, Marks: [3]

Steps:

1. Add a Cube named "RotatingObstacle", set Position to (2, 0.5, 2).
2. Attach Rigidbody, uncheck "Use Gravity".
3. Add TextMeshPro to UICanvas, name it "StopText", position at top-center.
4. Create a script named "RotatingObstacle".
5. Attach "RotatingObstacle" to the obstacle, assign "StopText".
6. Test to ensure rotation stops on collision.

Code:

csharp

```
using UnityEngine;
using TMPro;

public class RotatingObstacle : MonoBehaviour
{
    public float rotationSpeed = 50f; // Rotation speed in degrees per second
    public TextMeshProUGUI stopText; // Assign StopText in Inspector
    private bool isRotating = true; // Flag to control rotation

    void Update()
    {
        if (isRotating) // Rotate only if flag is true
        {
            transform.Rotate(0, rotationSpeed * Time.deltaTime, 0, Space.World); // Rotate around Y-axis
        }
    }

    void OnCollisionEnter(Collision collision)
    {
        if (collision.gameObject.CompareTag("Player")) // Check for Player collision
        {
            isRotating = false; // Stop rotation
            if (stopText != null)
            {
                stopText.text = "Stopped"; // Update TMP text
            }
        }
    }
}
```

Submission Requirements:

- Script screenshot showing rotation and stop logic.
- Inspector screenshot of rotating object.
- Game window (before and after collision).

SET B Solutions

Q1a: Player Cube Moving Left and Right

CO-4, SO-6, B.L.-3, Marks: [1×2]

Steps:

1. Create a new 3D project named "ARVRTest_SetB_Q1a".
2. Add a Plane named "Ground", Scale (10, 1, 10).
3. Add a Cube named "Player", Position (0, 0.5, 0), add Rigidbody and Box Collider.
4. Create a script named "PlayerSideMovement".
5. Attach "PlayerSideMovement" to Player.
6. Test with A/D for movement, ESC to stop/quit.
7. Build via File > Build Settings.

Code:

csharp

```
using UnityEngine;

public class PlayerSideMovement : MonoBehaviour
{
    public float moveSpeed = 5f; // Speed of side-to-side movement

    Rigidbody rb;

    void Awake()
    {
        rb = GetComponent<Rigidbody>(); // Get Rigidbody for physics
    }

    void Update()
    {
        float horizontalInput = Input.GetAxisRaw("Horizontal"); // Get A/D or Left/Right input
        Vector3 movement = new Vector3(horizontalInput, 0, 0) * moveSpeed * Time.deltaTime; // Frame-rate independent
        transform.Translate(movement, Space.Self); // Move left/right

        if (Input.GetKeyDown(KeyCode.Escape)) // ESC handling
        {
#if UNITY_EDITOR
            UnityEditor.EditorApplication.isPlaying = false; // Stop in Editor
#else
            Application.Quit(); // Quit in build
#endif
        }
    }
}
```

Submission Requirements:

- Screenshot of script logic showing motion and ESC handling.
- Game window screenshot showing cube movement.
- Hierarchy screenshot showing clean naming (e.g., "Player", "Ground").

Q1b: Display Player's Position

CO-4, SO-2, B.L.-3, Marks: [1×2]

Steps:

1. Create "UICanvas" in Hierarchy.
2. Add "PositionText" as child of Player, set Position to (0, 2, 0).
3. Create a script named "PositionDisplay".
4. Attach "PositionDisplay" to UICanvas or PositionText.
5. Test by moving Player to see position updates.

Code:

csharp

```
using UnityEngine;
using TMPro;

public class PositionDisplay : MonoBehaviour
{
    public TextMeshProUGUI positionText; // Assign PositionText in Inspector
    public Transform player; // Assign Player in Inspector

    void Update()
    {
        if (positionText != null && player != null)
        {
            Vector3 pos = player.position; // Get player's position
            positionText.text = $"Position: ({pos.x:F1}, {pos.y:F1}, {pos.z:F1})"; // Display with 1 decimal
        }
    }
}
```

Submission Requirements:

- Hierarchy screenshot showing TMP above player.
- Game window screenshot showing text visible.

- Inspector screenshot showing TMP component properties.

Q2: Obstacle Changes Color on Collision

CO-3, SO-6, B.L.-5, Marks: [3]

Steps:

1. Add a Cube named "ColorObstacle", Position (2, 0.5, 0).
2. Add TextMeshPro named "CollisionText" on UICanvas.
3. Create a script named "ColorChangeCollision".
4. Attach "ColorChangeCollision" to ColorObstacle, assign "CollisionText".
5. Test by colliding with Player.

Code:

```csharp
using UnityEngine;
using TMPro;

public class ColorChangeCollision : MonoBehaviour
{
    public TextMeshProUGUI collisionText; // Assign CollisionText in Inspector
    public Color newColor = Color.red; // Color to change to

    void OnCollisionEnter(Collision collision)
    {
        if (collision.gameObject.CompareTag("Player"))
        {
            GetComponent<Renderer>().material.color = newColor; // Change obstacle color
            if (collisionText != null)
            {
                collisionText.text = "Collision Detected!"; // Update TMP
            }
        }
    }
}
```

Submission Requirements:

- Script showing collision logic.
- Inspector of obstacle material.

- Game window before & after collision.
- Hierarchy showing clean naming.

Q3: Trigger Object Rotates Object on Collision

CO-3, SO-2, B.L.-5, Marks: [3]

Steps:

1. Create an Empty named "RotationTrigger", add child Cube "TriggerArea", Position (0, 0, 5), scale (2, 1, 2), check "Is Trigger".
2. Add a Cube named "RotatableObject", Position (0, 0.5, 5).
3. Add TextMeshPro named "ZoneText" on UICanvas.
4. Create a script named "RotationTrigger".
5. Attach "RotationTrigger" to TriggerZone, assign "RotatableObject" and "ZoneText".
6. Test by entering trigger.

Code:

csharp

```
using UnityEngine;
using TMPro;

public class RotationTrigger : MonoBehaviour
{
    public GameObject rotatableObject; // Assign RotatableObject in Inspector
    public TextMeshProUGUI zoneText; // Assign ZoneText in Inspector
    public float rotationSpeed = 50f; // Speed of rotation
    private bool isActive = false; // Track if player is inside

    void OnTriggerEnter(Collider other)
    {
        if (other.CompareTag("Player"))
        {
            isActive = true; // Start rotation
            if (zoneText != null)
            {
                zoneText.text = "Active Zone"; // Display text
            }
        }
    }

    void OnTriggerExit(Collider other)
    {
        if (other.CompareTag("Player"))
        {
            isActive = false; // Stop rotation
            if (zoneText != null)
            {
                zoneText.text = ""; // Clear text
            }
        }
    }

    void Update()
    {
        if (isActive && rotatableObject != null)
        {
            rotatableObject.transform.Rotate(0, rotationSpeed * Time.deltaTime, 0, Space.World); // Rotate while active
        }
    }
}
```

Submission Requirements:

- Script showing trigger logic + rotation control.

- Hierarchy of trigger + platform.
- Game window with TMP visible.

Q4: Trigger Changes Color on Collision

CO-4, SO-2, B.L.-6, Marks: [3]

Steps:

1. Create an Empty named "ColorTrigger", add child Cube "TriggerArea", Position (0, 0, 5), scale (2, 1, 2), check "Is Trigger".
2. Add a Cube named "ColorObject", Position (0, 0.5, 5).
3. Add TextMeshPro named "ZoneText" on UICanvas.
4. Create a script named "ColorTrigger".
5. Attach "ColorTrigger" to ColorTrigger, assign "ColorObject" and "ZoneText".
6. Test by entering trigger.

Code:

```csharp
```

```
using UnityEngine;
using TMPro;

public class ColorTrigger : MonoBehaviour
{
    public GameObject colorObject; // Assign ColorObject in Inspector
    public TextMeshProUGUI zoneText; // Assign ZoneText in Inspector
    public Color activeColor = Color.blue; // Color when active
    private Color originalColor; // Store original color
    private bool isActive = false; // Track if player is inside

    void Start()
    {
        if (colorObject != null)
        {
            originalColor = colorObject.GetComponent<Renderer>().material.color; // Save original color
        }
    }

    void OnTriggerEnter(Collider other)
    {
        if (other.CompareTag("Player"))
        {
            isActive = true; // Activate color change
            if (colorObject != null)
            {
                colorObject.GetComponent<Renderer>().material.color = activeColor; // Change to active color
            }
            if (zoneText != null)
            {
                zoneText.text = "Active Zone"; // Display text
            }
        }
    }

    void OnTriggerExit(Collider other)
    {
        if (other.CompareTag("Player"))
        {
            isActive = false; // Deactivate
            if (colorObject != null)
            {
                colorObject.GetComponent<Renderer>().material.color = originalColor; // Restore original color
            }
            if (zoneText != null)
            {
```

```
            zoneText.text = ""; // Clear text
        }
    }
}
}
```

Submission Requirements:

- Script showing sliding logic.
- Hierarchy showing doors and trigger.
- Game window (before and after door movement).

Important Notes

1. **Player Tag:** Ensure the "Player" tag is added to the Player GameObject in the Inspector (Tag dropdown > Add Tag > Create "Player").
2. **TextMeshPro Setup:** Import TextMeshPro via Window > TextMeshPro > Import TMP Essential Resources if not already done.
3. **Building:** Build the project via File > Build Settings > Add Open Scenes > Build for ESC to work in .exe.
4. **Unity Version:** Use Unity 2021.3 LTS or newer for compatibility.

End of Test Solutions