```
using UnityEngine;
using UnityEngine.SceneManagement;
using TMPro;
using UnityEngine.UI;

public class PlayerController : MonoBehaviour

{
    [Header("Movement")]
    public float forwardSpeed = 15f;
    public float sidewaysSpeed = 12f;
    public float leftLimit = -5f;
    public float rightLimit = 5f;

    [Header("UI & Game Over")]
    public GameObject gameOverPanel;              // Drag GameOverPanel here (in scene)
    public TextMeshProUGUI finalScoreText;        // Drag FinalScoreText (TMP) here
    public TextMeshProUGUI finalTimeText;         // Drag FinalTimeText (TMP) here
    public Button restartButton;                  // Drag Restart Button (optional)
    public Button mainMenuButton;                 // Drag Main Menu Button (optional)
    private int highScore = 0;
    public TextMeshProUGUI highScoreText;        // drag High Score UI here



    // gameplay state
    private Rigidbody rb;
    private bool isGameOver = false;
    private int coinCount = 0;
    private float elapsedTime = 0f;

    public TextMeshProUGUI coinText;
    public TextMeshProUGUI timerText;
    private float survivalTime = 0f;


    [Header("Difficulty Settings")]
    public float difficultyIncreaseInterval = 30f; // every 10 seconds
    public float speedIncreaseAmount = 10f;         // increase by this much
    private float nextDifficultyTime = 0f;

    void Awake()
```

```
43      {
44          rb = GetComponent<Rigidbody>();
45          if (rb == null) rb = gameObject.AddComponent<Rigidbody>();
46          rb.freezeRotation = true;
47      }
48
49      void Start()
50      {
51          // Make sure GameOver panel is hidden at start
52          if (gameOverPanel != null) gameOverPanel.SetActive(false);
53
54          // Optional: hook up restart button if you want to wire via script
55          if (restartButton != null)
56          {
57              restartButton.onClick.RemoveAllListeners();
58              restartButton.onClick.AddListener(RestartGame);
59          }
60          if (mainMenuButton != null)
61          {
62              mainMenuButton.onClick.RemoveAllListeners();
63              mainMenuButton.onClick.AddListener(GoToMainMenu);
64          }
65          if (coinText != null)
66              coinText.text = "Coins: 0";
67          highScore = PlayerPrefs.GetInt("HighScore", 0); // Load saved
              high score
68          UpdateHighScoreUI();
69
70          PlayerPrefs.DeleteKey("HighScore");
71
72
73      }
74
75      void Update()
76      {
77          if (!isGameOver)
78          {
79              // accumulate survival time
80              elapsedTime += Time.deltaTime;
81              // (optional) update a live timer UI here if you have one
82          }
83
84          // debug restart hotkey while testing
85          if (isGameOver && Input.GetKeyDown(KeyCode.R))
86          {
87              RestartGame();
88          }
89          if (!isGameOver)
90          {
```

```
91              survivalTime += Time.deltaTime;
92              if (timerText != null)
93                  timerText.text = "Time: " + Mathf.FloorToInt
                      (survivalTime).ToString();
94          }
95
96          // Optional restart key for testing
97          if (isGameOver && Input.GetKeyDown(KeyCode.R))
98          {
99              UnityEngine.SceneManagement.SceneManager.LoadScene(
100                 UnityEngine.SceneManagement.SceneManager.GetActiveScene
                      ().name
101             );
102         }
103
104         if (!isGameOver && survivalTime >= nextDifficultyTime)
105         {
106             nextDifficultyTime += difficultyIncreaseInterval;
107             IncreaseDifficulty();
108         }
109
110     }
111
112     void UpdateHighScoreUI()
113 {
114     if (highScoreText != null)
115         highScoreText.text = "High Score: " + highScore.ToString();
116 }
117
118
119     void FixedUpdate()
120     {
121         if (isGameOver) return;
122
123         // Physics-based movement while keeping gravity intact
124         Vector3 newVelocity = rb.linearVelocity;
125         newVelocity.z = forwardSpeed;
126         float horizontal = Input.GetAxis("Horizontal");
127         newVelocity.x = horizontal * sidewaysSpeed;
128         rb.linearVelocity = new Vector3(newVelocity.x,
              rb.linearVelocity.y, newVelocity.z);
129
130         // clamp X position
131         Vector3 pos = rb.position;
132         pos.x = Mathf.Clamp(pos.x, leftLimit, rightLimit);
133         rb.MovePosition(pos);
134     }
135
136     private void OnTriggerEnter(Collider other)
```

```
137     {
138         // Coin pickup
139         if (other.CompareTag("Coin"))
140         {
141             coinCount++;
142             Destroy(other.gameObject);
143 
144             if (coinText != null)
145                 coinText.text = "Coins: " + coinCount;
146 
147             // ☑ High score check and save
148             if (coinCount > highScore)
149             {
150                 highScore = coinCount;
151                 PlayerPrefs.SetInt("HighScore", highScore);
152                 UpdateHighScoreUI();
153             }
154 
155         }
156         // Deadly collisions (snakes, obstacles)
157         else if (other.CompareTag("Snake") || other.CompareTag ↵
              ("Obstacle"))
158         {
159             Debug.Log($"[DEBUG] Hit deadly object: {other.tag}. ↵
                Triggering GameOver.");
160             GameOver();
161         }
162         if (other.CompareTag("Coin"))
163         {
164             coinCount++;
165             Destroy(other.gameObject);
166 
167             if (coinText != null)
168                 coinText.text = "Coins: " + coinCount;
169         }
170         if (other.CompareTag("Coin"))
171         {
172             SoundManager.instance.PlayCoin();
173             // existing coin logic
174         }
175         if (other.CompareTag("Snake") || other.CompareTag("Obstacle"))
176         {
177             SoundManager.instance.PlayHit();
178             // existing game-over logic
179         }
180 
181 
182     }
183 
```

```
184     public void GameOver()
185     {
186         if (isGameOver) return;
187         isGameOver = true;
188 
189         // Stop gameplay
190         rb.linearVelocity = Vector3.zero;
191         // Pause time-based systems if desired (but UI will still work)
192         Time.timeScale = 0f;
193 
194         // Debug print the values we will show
195         Debug.Log($"[DEBUG] GameOver called - coins: {coinCount},
              elapsedTime: {elapsedTime}");
196 
197         // Activate panel and set UI texts (null-checks for safety)
198         if (gameOverPanel != null) gameOverPanel.SetActive(true);
199         else Debug.LogWarning("GameOverPanel reference is null on
              PlayerController.");
200 
201         if (finalScoreText != null)
202             finalScoreText.text = "Final Score: " + coinCount.ToString();
203         else Debug.LogWarning("finalScoreText not assigned in
              Inspector.");
204 
205         if (finalTimeText != null)
206             finalTimeText.text = "Time: " + Mathf.FloorToInt
                  (elapsedTime).ToString() + "s";
207         else Debug.LogWarning("finalTimeText not assigned in Inspector.");
208 
209         if (finalScoreText != null)
210             finalScoreText.text = "Final Score: " + coinCount.ToString();
211 
212         if (finalTimeText != null)
213             finalTimeText.text = "Time: " + Mathf.FloorToInt
                  (elapsedTime).ToString() + "s";
214 
215         // ☑ Show High Score too (if you have a text in the
              GameOverPanel)
216         if (highScoreText != null)
217             highScoreText.text = "High Score: " + highScore.ToString();
218 
219 
220 
221     }
222 
223     public void RestartGame()
224     {
225         Time.timeScale = 1f; // restore time scale before reload
226         SceneManager.LoadScene(SceneManager.GetActiveScene().name);
```

```
227     }
228
229     public void GoToMainMenu()
230     {
231         Time.timeScale = 1f;
232         SceneManager.LoadScene("MainMenuScenne");
233     }
234
235     void IncreaseDifficulty()
236     {
237         forwardSpeed += speedIncreaseAmount;
238         sidewaysSpeed += 0.2f;
239
240         // Update all obstacle spawners using the new Unity 6 API
241         ObstacleSpawner[] spawners = FindObjectsByType<ObstacleSpawner>
              (FindObjectsSortMode.None);
242         foreach (var spawner in spawners)
243         {
244             spawner.spawnInterval = Mathf.Max(0.5f, spawner.spawnInterval
                  - 0.3f);
245         }
246
247         Debug.Log("Difficulty increased! New speed: " + forwardSpeed);
248     }
249
250
251 }
252
```