

# Unity Backend

“랭킹” 기능을 이용한 유저 랭킹 관리 (랭킹 등록)

Created in 2023-06-07  
Last Updated 2023-06-09  
Unity Version 2022.2.2f1

## *Index*

- ◆ 인게임 점수 구현
- ◆ 게임오버 UI 구현
- ◆ 랭킹으로 사용할 유저 정보 데이터 추가
- ◆ 랭킹 데이터 등록

# 인게임 점수 구현

- 인게임 점수 관리
- 인게임 점수 출력
- 적, 운석 폭발 효과



# 인게임 점수 구현

## ■ 인게임 점수 관리

- 점수를 관리하는 변수와 프로퍼티 정의
  - GameController Script 수정

```
1 using UnityEngine;
2
3 public class GameController : MonoBehaviour
4 {
5     private int score = 0;
6
7     public bool IsGameOver { set; get; } = false;
8     public int Score
9     {
10         set => score = Mathf.Max(0, value);
11         get => score;
12     }
13
14     public void GameOver()...
15
16     public void AfterGameOver()...
17
18 }
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
```



# 인게임 점수 구현

- 적을 처치했을 때 호출하는 OnDie() 메소드 정의
  - Enemy Script 수정

```
1 using UnityEngine;
2
3 public class Enemy : MonoBehaviour
4 {
5     [SerializeField]
6     private int scorePoint = 100; // 적을 처치했을 때 획득하는 점수
7     private GameController gameController;
8
9     public void Setup(GameController gameController) ...
10
11
12
13
14     public void OnDie()
15     {
16         // 플레이어의 점수를 scorePoint만큼 증가
17         gameController.Score += scorePoint;
18         // 적 캐릭터 삭제
19         Destroy(gameObject);
20     }
21
22     private void OnTriggerEnter2D(Collider2D collision) ...
23
24
25
26
27
28
29 }
```



# 인게임 점수 구현

- 플레이어 발사체와 충돌한 적을 바로 삭제하지 않고 OnDie() 메소드 호출
  - PlayerProjectile Script 수정

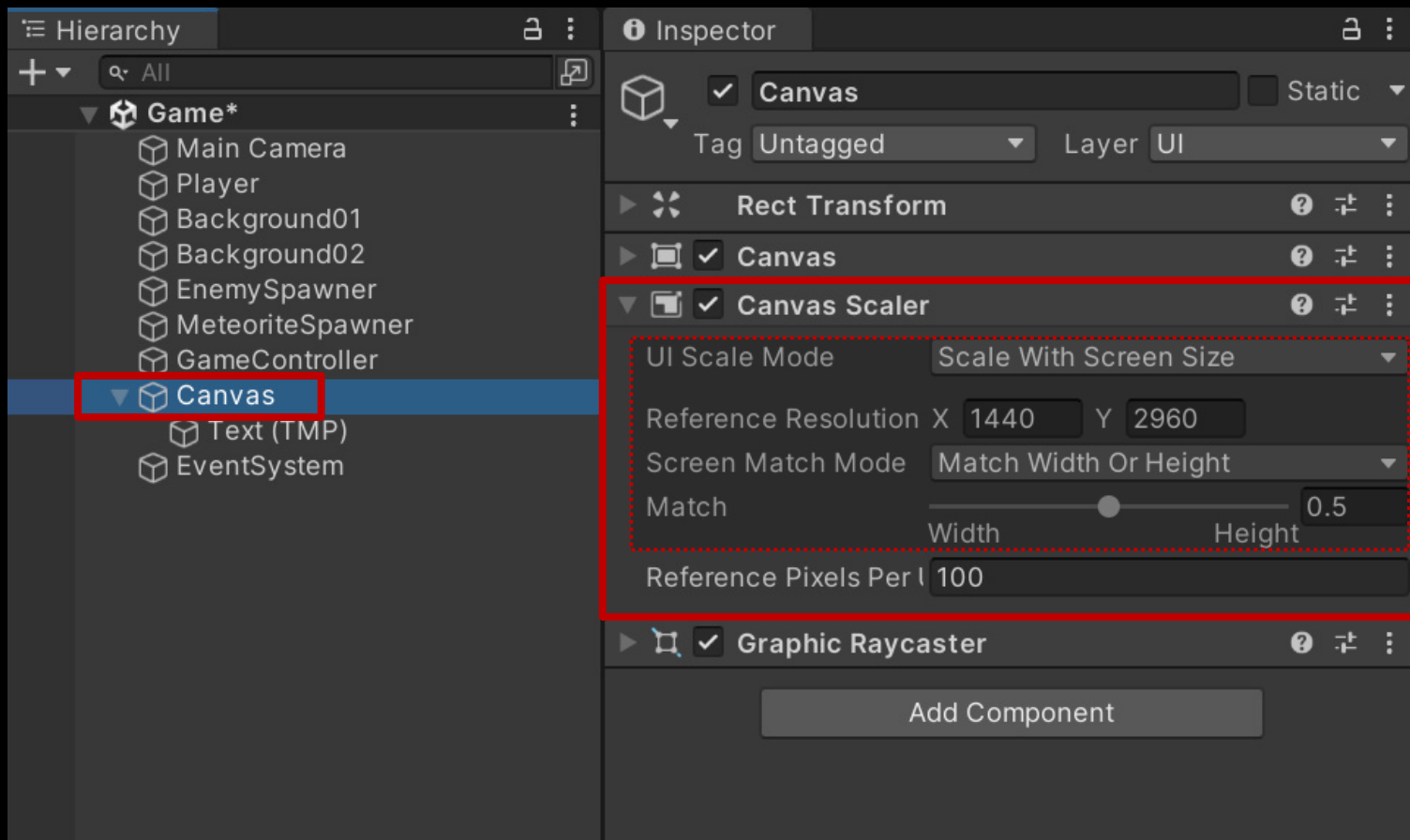
```
1  using UnityEngine;
2
3  public class PlayerProjectile : MonoBehaviour
4  {
5      private void OnTriggerEnter2D(Collider2D collision)
6      {
7          if ( collision.CompareTag("Enemy") )
8          {
9              //Destroy(collision.gameObject);
10             collision.GetComponent<Enemy>().OnDie();
11
12             Destroy(gameObject);
13         }
14     }
15 }
```



# 인게임 점수 구현

## ■ 인게임 점수 출력

- 인게임 획득 점수를 출력하는 "Text - TextMeshPro" UI 생성 및 설정
  - GameObject - UI - "Text - TextMeshPro"





# 인게임 점수 구현

- 인게임 획득 점수를 출력하는 "Text - TextMeshPro" UI 생성 및 설정 (계속)

The image shows the Unity Inspector and Hierarchy panels. The Hierarchy panel on the left shows the 'PlayerScore' object selected under the 'Canvas' component. The Inspector panel on the right shows the 'TextMeshPro - Text (UI)' component selected. The 'Rect Transform' component is highlighted with a red dashed box, showing the following values:

Property	Value
Left	25
Pos Y	-20
Pos Z	0
Right	0
Height	50

The 'Anchors' component is also highlighted with a red dashed box, showing the following values:

Property	X	Y
Min	0	1
Max	1	1
Pivot	0.5	1

The 'TextMeshPro - Text (UI)' component is highlighted with a red dashed box, showing the following values:

Property	Value
Text Input	SCORE 00000
Text Style	Normal
Font Asset	NotoSansKR-Bold SDF (TMP_Fc)
Material Preset	NotoSansKR-Bold SDF Material
Font Style	B I U S ab AB SC
Font Size	50

The 'Alignment' section is also highlighted with a red dashed box, showing the following values:

Property	Value
Alignment	Left





# 인게임 점수 구현

- 인게임, 게임오버 UI들을 제어하는 스크립트 생성 및 작성
  - C# Script 생성 후 스크립트의 이름을 "GameUIController"로 변경

```
1  using UnityEngine;
2  using TMPro;
3
4  public class GameUIController : MonoBehaviour
5  {
6      [SerializeField]
7      private GameController  gameController;
8
9      [Header("InGame")]
10     [SerializeField]
11     private TextMeshProUGUI textScore;
12
13     private void Update()
14     {
15         textScore.text = $"SCORE {gameController.Score}";
16     }
17 }
```



# 인게임 점수 구현

- GameController 오브젝트에 "GameUIController" 컴포넌트 추가 및 설정

The screenshot displays the Unity Hierarchy and Inspector panels. In the Hierarchy panel, the 'GameController' object is selected and highlighted with a blue bar. Red boxes highlight 'GameController' and 'PlayerScore' in the Hierarchy, and 'Game UI Controller (Script)' in the Inspector. Red arrows point from the Hierarchy boxes to the Inspector box. The Inspector panel shows the 'GameController' object with the following components and settings:

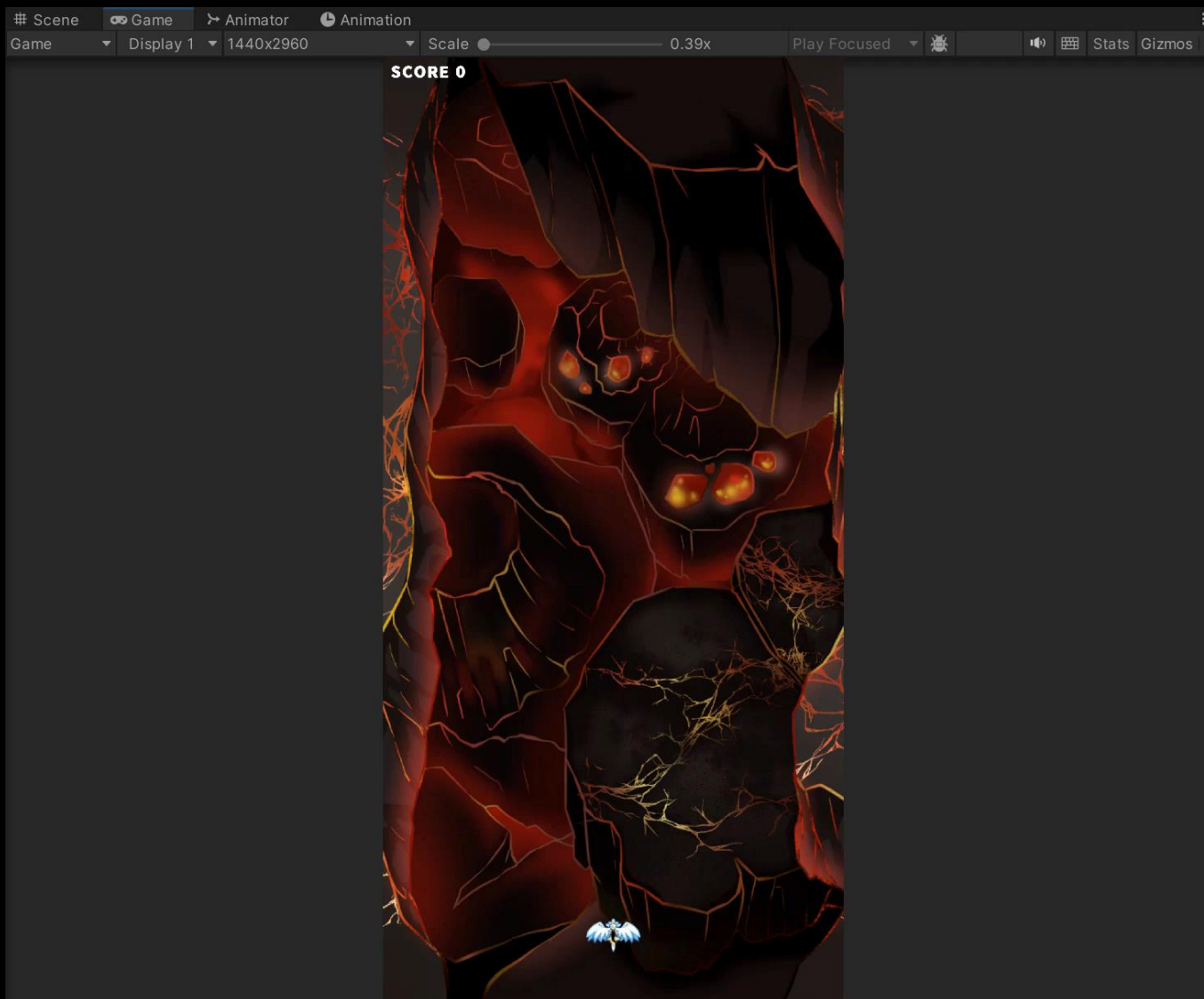
- GameController** (Script): Tag: Untagged, Layer: Default
- Transform** (Component)
- Game Controller (Script)** (Component)
- Game UI Controller (Script)** (Component):
  - Script: GameUIController
  - Game Controller: GameController (Game Controller)
  - inGame:
  - Text Score: PlayerScore (Text Mesh Pro UGI)

An 'Add Component' button is visible at the bottom of the Inspector panel.



# 인게임 점수 구현

## ■ 결과 화면





# 인게임 점수 구현

## ■ 적, 운석 폭발 효과

### ■ 폭발 이펙트 생성

#### □ GameObject - Effects - Particle System

The screenshot displays the Unity Inspector window for a GameObject named 'Explosion'. The Hierarchy panel on the left shows the object's position within the scene. The Inspector panel is divided into two sections: 'Transform' and 'Particle System'.

**Transform Section:** This section is highlighted with a red box. It shows the following properties:

Property	X	Y	Z
Position	0	0	0
Rotation	0	0	0
Scale	1	1	1

**Particle System Section:** This section is also visible and contains the following settings:

- Explosion** (Particle System Name)
- Emission
- Shape
- Velocity over Lifetime
- Limit Velocity over Lifetime
- Inherit Velocity
- Lifetime by Emitter Speed
- Force over Lifetime
- Color over Lifetime
- Color by Speed



# 인게임 점수 구현

## ■ 폭발 이펙트 설정

**Explosion**

Duration: 0.5

Looping:

Prewarm:

Start Delay: 0

Start Lifetime: 0.2

Start Speed: 1

3D Start Size:

Start Size: 0.2 (X) 0.5 (Y)

3D Start Rotation: **Random Between Two Constants 선택**

Start Rotation: 0

Flip Rotation: 0

Start Color:

Gravity Source: 3D Physics

Gravity Modifier: 0

Simulation Space: Local

Simulation Speed: 1

Delta Time: Scaled

Scaling Mode: Local

Play On Awake\*:

Emitter Velocity Mode: Rigidbody

Max Particles: 20

Auto Random Seed:

Stop Action: None

Culling Mode: Automatic

Ring Buffer Mode: Disabled

Emission

Rate over Time: 0

Rate over Distance: 0

Bursts

Time	Count	Cycles	Interval	Probability
0.000	20	Cy 1	0.010	1.00

Shape

Shape: Circle

Radius: 0.1

Radius Thickness: 0.1

Arc: 360

Mode: Random

Spread: 0

Texture: None (Texture 2D)

Position: X: 0, Y: 0, Z: 0

Rotation: X: 0, Y: 0, Z: 0

Scale: X: 1, Y: 1, Z: 1

Align To Direction:

Randomize Direction: 0

Spherize Direction: 0

Randomize Position: 0

Scene Tools



# 인게임 점수 구현

## ■ 폭발 이펙트 설정 (계속)

The image shows the Unity Hierarchy and Inspector panels. The Hierarchy panel on the left shows a 'Particle System' object under 'Explosion'. The Inspector panel on the right shows the 'Particle System' settings. The 'Color over Lifetime' property is selected and highlighted with a red box. The 'Gradient Editor' window is open, showing a gradient from yellow to red. The 'Renderer' panel on the right shows the 'Order in Layer' property set to 2, also highlighted with a red box.

**Particle System**

- Explosion
- Emission
- Shape
- Velocity over Lifetime
- Limit Velocity over Lifetime
- Inherit Velocity
- Lifetime by Emitter Speed
- Force over Lifetime
- Color over Lifetime**
- Color
- Color by Speed
- Size over Lifetime
- Size by Speed
- Rotation over Lifetime
- Rotation by Speed
- External Forces
- Noise
- Collision
- Triggers
- Sub Emitters
- Texture Sheet Animation
- Lights
- Trails
- Custom Data
- Renderer

**Gradient Editor**

Mode: Blend (Classic)

Color (255, 255, 0) Location 0%

Color (255, 0, 0) Location 100%

**Renderer**

- Render Mode: Billboard
- Normal Direction: 1
- Material: Default-ParticleSystem
- Sort Mode: None
- Sorting Fudge: 0
- Min Particle Size: 0
- Max Particle Size: 0.5
- Render Alignment: View
- Flip: X 0 Y 0 Z 0
- Allow Roll:
- Pivot: X 0 Y 0 Z 0
- Visualize Pivot:
- Masking: No Masking
- Apply Active Color Space:
- Custom Vertex Streams:
- Cast Shadows: Off
- Receive Shadows:
- Shadow Bias: 0
- Motion Vectors: Per Object Motion
- Sorting Layer ID: Default
- Order in Layer: 2**
- Light Probes: Off
- Reflection Probes: Off



# 인게임 점수 구현

- 파티클의 재생이 완료되면 오브젝트를 삭제하는 스크립트 생성 및 작성
  - C# Script 생성 후 스크립트의 이름을 "ParticleAutoDestroyer"로 변경

```
1  using UnityEngine;
2
3  public class ParticleAutoDestroyer : MonoBehaviour
4  {
5      private ParticleSystem particle;
6
7      private void Awake()
8      {
9          particle = GetComponent<ParticleSystem>();
10     }
11
12     private void Update()
13     {
14         // 파티클 재생이 완료되면 삭제
15         if ( particle.isPlaying == false )
16         {
17             Destroy(gameObject);
18         }
19     }
20 }
```



# 인게임 점수 구현

- Explosion 오브젝트에 컴포넌트 추가 및 설정

The screenshot displays the Unity Inspector window for an object named 'Explosion'. The 'Audio Source' component is selected, and its 'AudioClip' property is set to 'Explosion1'. The 'Particle Auto Destroyer (Script)' component is also visible, with its 'Script' property set to 'ParticleAutoDestroyer'. The 'Assets' panel on the right shows the 'Explosion1' audio clip file.

**Hierarchy:** Game\* > Explosion

**Inspector:**

- Explosion (Static)
- Tag: Untagged, Layer: Default
- Transform
- Particle System
- Audio Source**
  - AudioClip: Explosion1
  - Output: None (Audio Mixer Group)
  - Mute:
  - Bypass Effects:
  - Bypass Listener Effec:
  - Bypass Reverb Zones:
  - Play On Awake:
  - Loop:
  - Priority: 128
  - Volume: 1
  - Pitch: 1
  - Stereo Pan: 0
  - Spatial Blend: 0
  - Reverb Zone Mix: 1
  - 3D Sound Settings
- Particle Auto Destroyer (Script)**
  - Script: ParticleAutoDestroyer

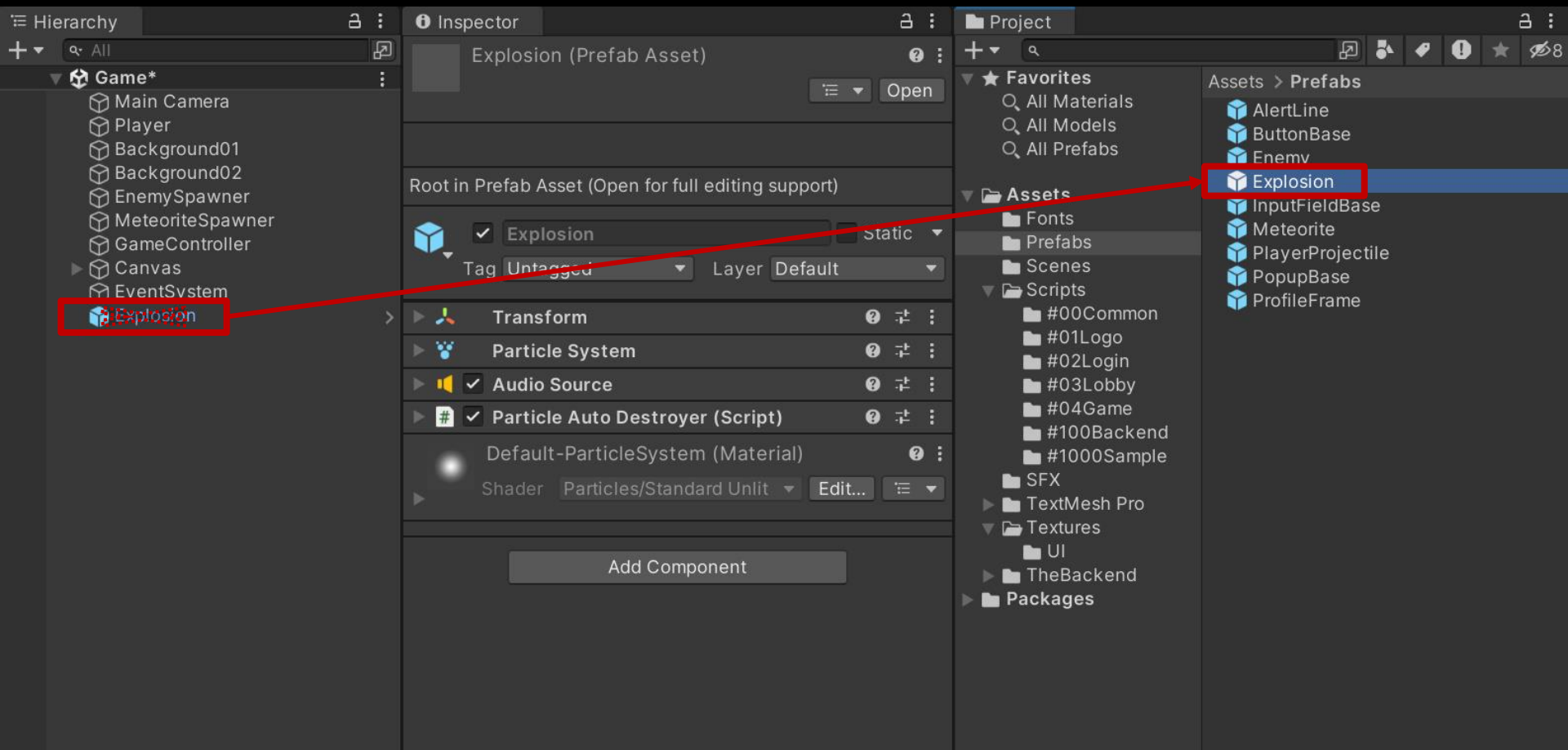
**Assets:** SFX > Explosion1





# 인게임 점수 구현

- Explosion 오브젝트 Prefab 생성
  - Hierarchy View의 "Explosion" 오브젝트를 Project View로 Drag & Drop
  - Hierarchy View에 있는 "Explosion" 오브젝트 삭제





# 인게임 점수 구현

## ■ 적 캐릭터가 사망할 때 폭발 이펙트 생성

### □ Enemy Script 수정

```
1 using UnityEngine;
2
3 public class Enemy : MonoBehaviour
4 {
5     [SerializeField]
6     private int scorePoint = 100; // 적을 처치했을 때 획득하는 점수
7     [SerializeField]
8     private GameObject explosionPrefab; // 폭발 이펙트 프리팹
9     private GameController gameController;
10
11     public void Setup(GameController gameController) {...}
12
13     public void OnDie()
14     {
15         // 폭발 이펙트 생성
16         Instantiate(explosionPrefab, transform.position, Quaternion.identity);
17         // 플레이어의 점수를 scorePoint만큼 증가
18         gameController.Score += scorePoint;
19         // 적 캐릭터 삭제
20         Destroy(gameObject);
21     }
22
23     private void OnTriggerEnter2D(Collider2D collision)
24     {
25         if ( collision.CompareTag("Player") )
26         {
27             OnDie();
28             gameController.GameOver();
29         }
30     }
31 }
32
33
34
```



# 인게임 점수 구현

- Enemy 프리팹의 "Enemy" 컴포넌트 변수 설정

The image shows the Unity Inspector and Project panels. The Inspector panel displays the 'Enemy (Prefab Asset)' with the following components and settings:

- Enemy (Script):
  - Script: Enemy
  - Score Point: 100
  - Explosion Prefab: Explosion
- Destroy By Position (Script)
- Sprites-Default (Material)

The Project panel shows the 'Assets > Prefabs' hierarchy, with the 'Enemy' prefab highlighted. A red box highlights the 'Enemy' and 'Explosion' prefabs in the Project panel, and a red arrow points from the 'Explosion Prefab' field in the Inspector to the 'Explosion' prefab in the Project panel.



# 인게임 점수 구현

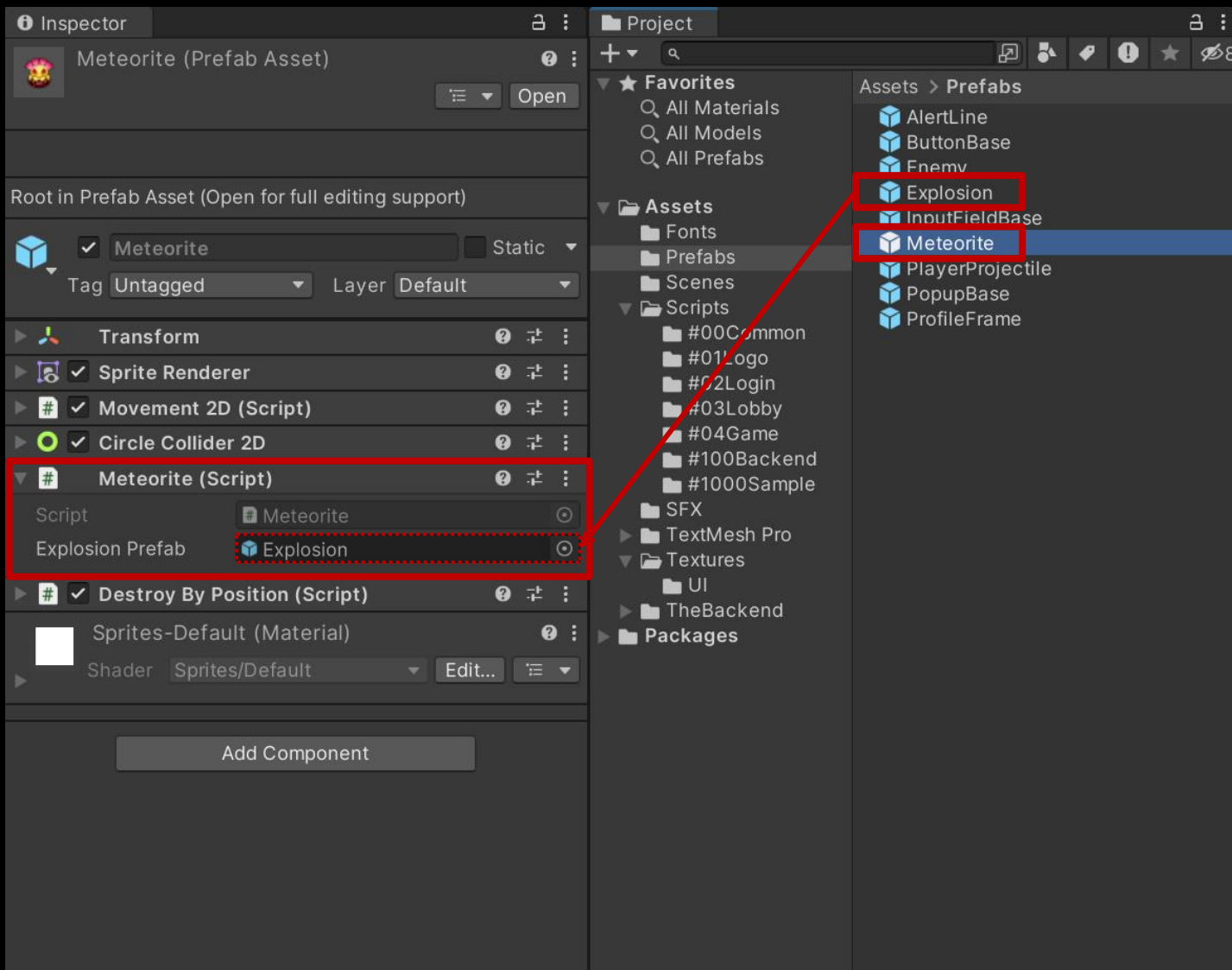
- 운석이 사망할 때 폭발 이펙트 생성
  - Meteorite Script 수정

```
1  using UnityEngine;
2
3  public class Meteorite : MonoBehaviour
4  {
5      [SerializeField]
6      private GameObject explosionPrefab; // 폭발 이펙트 프리팹
7      private GameController gameController;
8
9      public void Setup(GameController gameController) {...}
10
11
12
13
14     private void OnTriggerEnter2D(Collider2D collision)
15     {
16         if ( collision.CompareTag("Player") )
17         {
18             // 폭발 이펙트 생성
19             Instantiate(explosionPrefab, transform.position, Quaternion.identity);
20             // 운석 삭제
21             Destroy(gameObject);
22
23             gameController.GameOver();
24         }
25     }
26 }
```



# 인게임 점수 구현

- Meteorite 프리팹의 "Meteorite" 컴포넌트 변수 설정





# 인게임 점수 구현

## ■ 결과 화면



# 게임오버 UI 구현

- 게임오버 되었을 때 인게임 오브젝트 비활성화
- 게임오버 UI 출력
- Scale Effect
- Counting Effect



# 게임오버 UI 구현

- 게임오버 되었을 때 인게임 오브젝트 비활성화
  - 게임오버 되었을 때 호출하고 싶은 메소드 등록 및 호출
    - GameController Script 수정

```
1 using UnityEngine;
2 using UnityEngine.Events;
3
4 public class GameController : MonoBehaviour
5 {
6     [SerializeField]
7     private UnityEvent onGameOver; // 게임오버 되었을 때 호출할 메소드 등록 및 실행
8
9     private int score = 0;
10
11     public bool IsGameOver { set; get; } = false;
12     public int Score...
13
14
15
16
17
```





# 게임오버 UI 구현

## □ GameController Script 수정 (계속)

```
18 public void GameOver()  
19 {  
20     // 중복 처리 되지 않도록 bool 변수로 제어  
21     if ( IsGameOver == true ) return;  
22  
23     IsGameOver = true;  
24  
25     // 게임오버 되었을 때 호출할 메소드들을 실행  
26     onGameOver.Invoke();  
27  
28     // 경험치 증가 및 레벨업 여부 검사  
29     // (현재 레벨 시스템에 대한 설정이 없기 때문에 경험치의 최대치를 100으로 가정)  
30     // (게임을 한번 플레이할 때마다 경험치는 25씩 증가)  
31     BackendGameData.Instance.UserGameData.experience += 25;  
32     if ( BackendGameData.Instance.UserGameData.experience >= 100 ) ...  
37  
38     // 게임 정보 업데이트  
39     BackendGameData.Instance.GameDataUpdate( /*AfterGameOver*/ );  
40 }  
41  
42 /*public void AfterGameOver()  
43 {  
44     // 로비 씬으로 이동  
45     Utils.LoadScene(SceneNames.Lobby);  
46 }*/  
47 }
```

게임오버 UI 출력 이후에 씬을 변경하기  
때문에 AfterGameOver() 메소드 삭제



# 게임오버 UI 구현

- GameController 오브젝트의 "GameController" 컴포넌트 변수 설정

The screenshot displays the Unity Inspector for the GameController component. The Hierarchy panel on the left shows the following structure:

- Game
  - Main Camera
  - Player
  - Background01
  - Background02
  - EnemySpawner
  - MeteoriteSpawner
  - GameController
  - Canvas
  - EventSystem

The Inspector panel shows the following configuration for the GameController component:

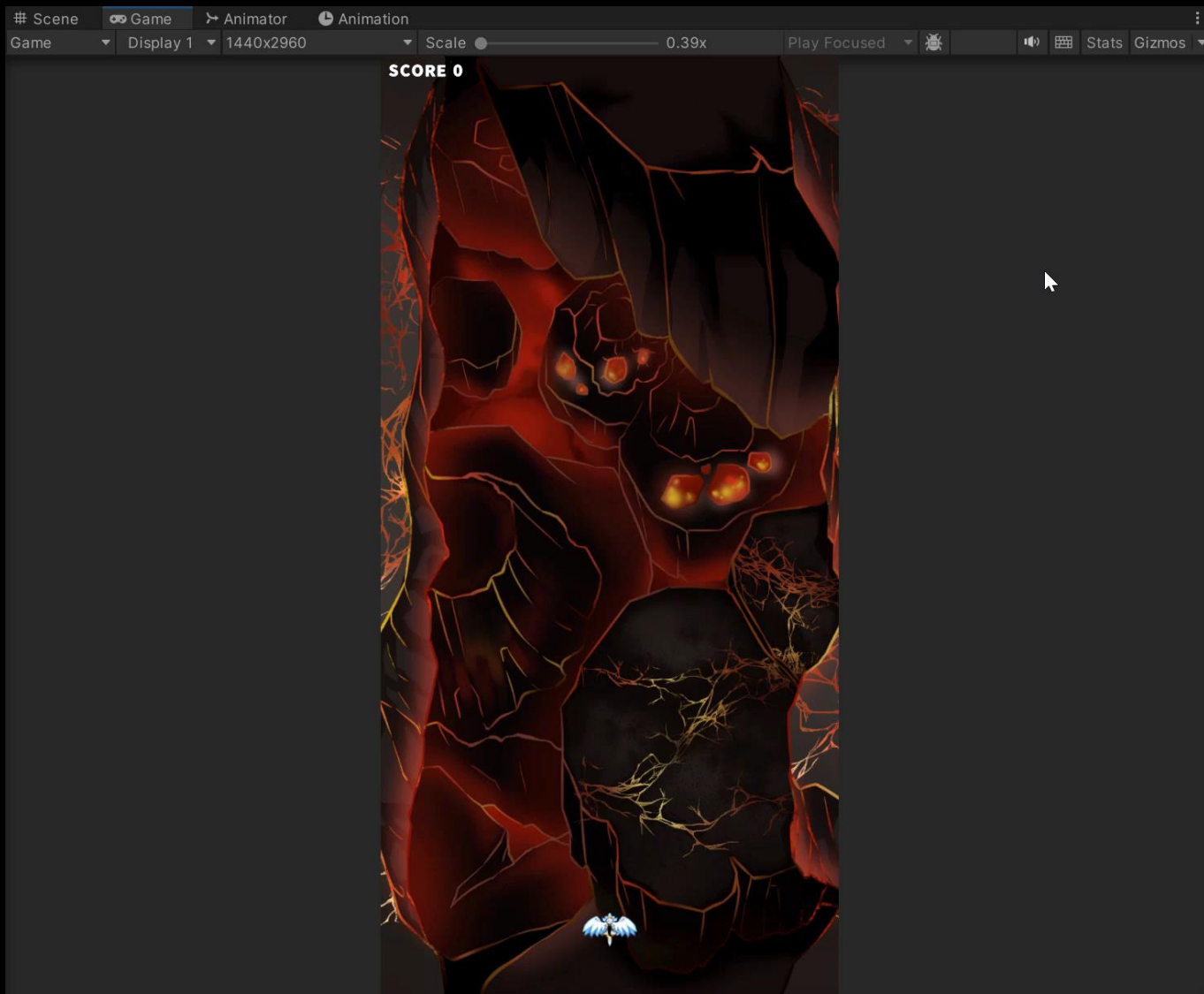
- Component: GameController (Script)
- Script: GameController
- On Game Over ()
  - Event 1: Runtime Only (checked), GameObject.SetActive, Meteorite (target)
  - Event 2: Runtime Only (checked), GameObject.SetActive, EnemySp: (target)
  - Event 3: Runtime Only (checked), GameObject.SetActive, Player (target)

Red boxes and arrows in the image highlight the configuration of the On Game Over events and the corresponding objects in the Hierarchy panel.



# 게임오버 UI 구현

## ■ 결과 화면

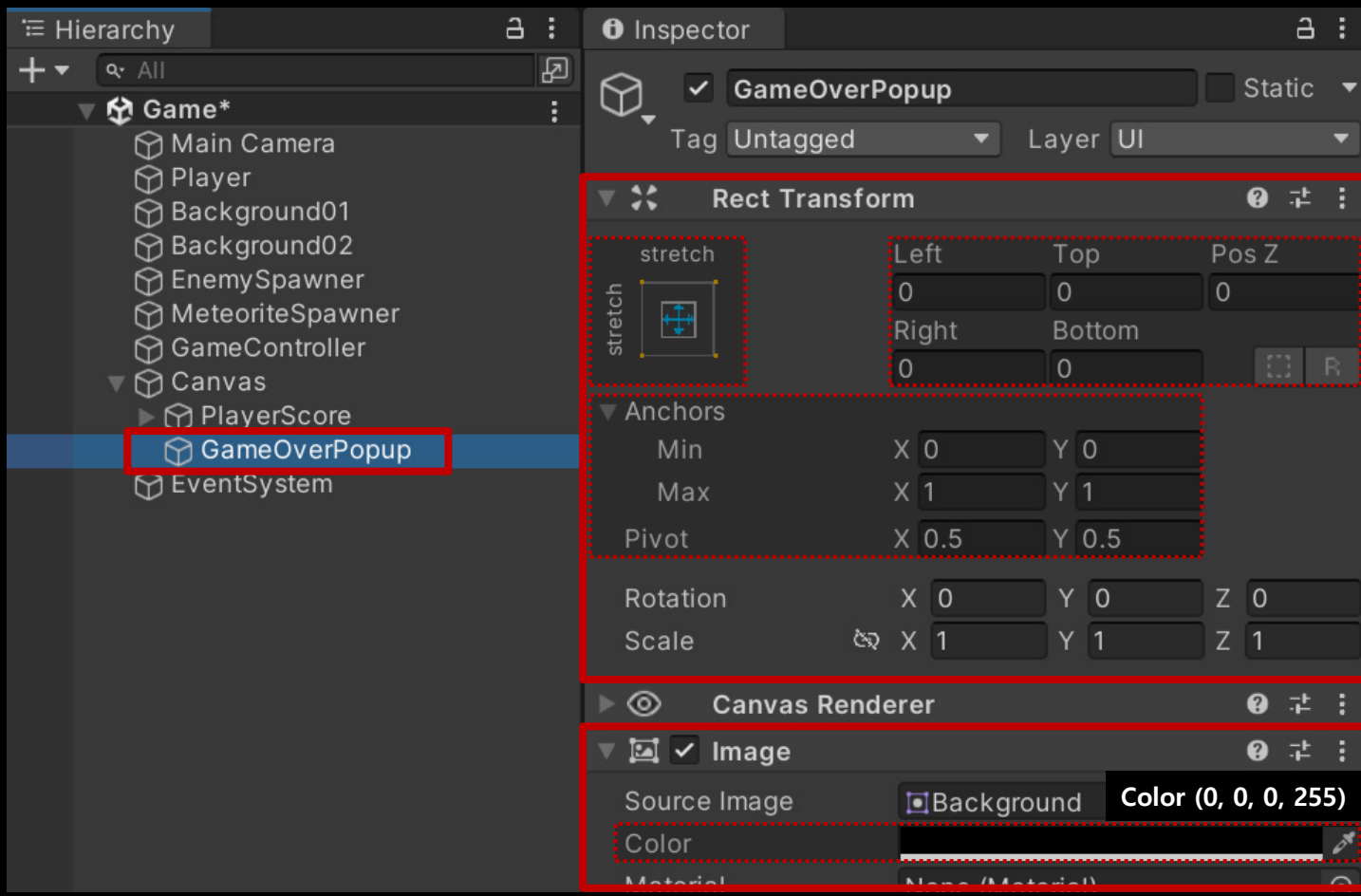




# 게임오버 UI 구현

## ■ 게임오버 UI 출력

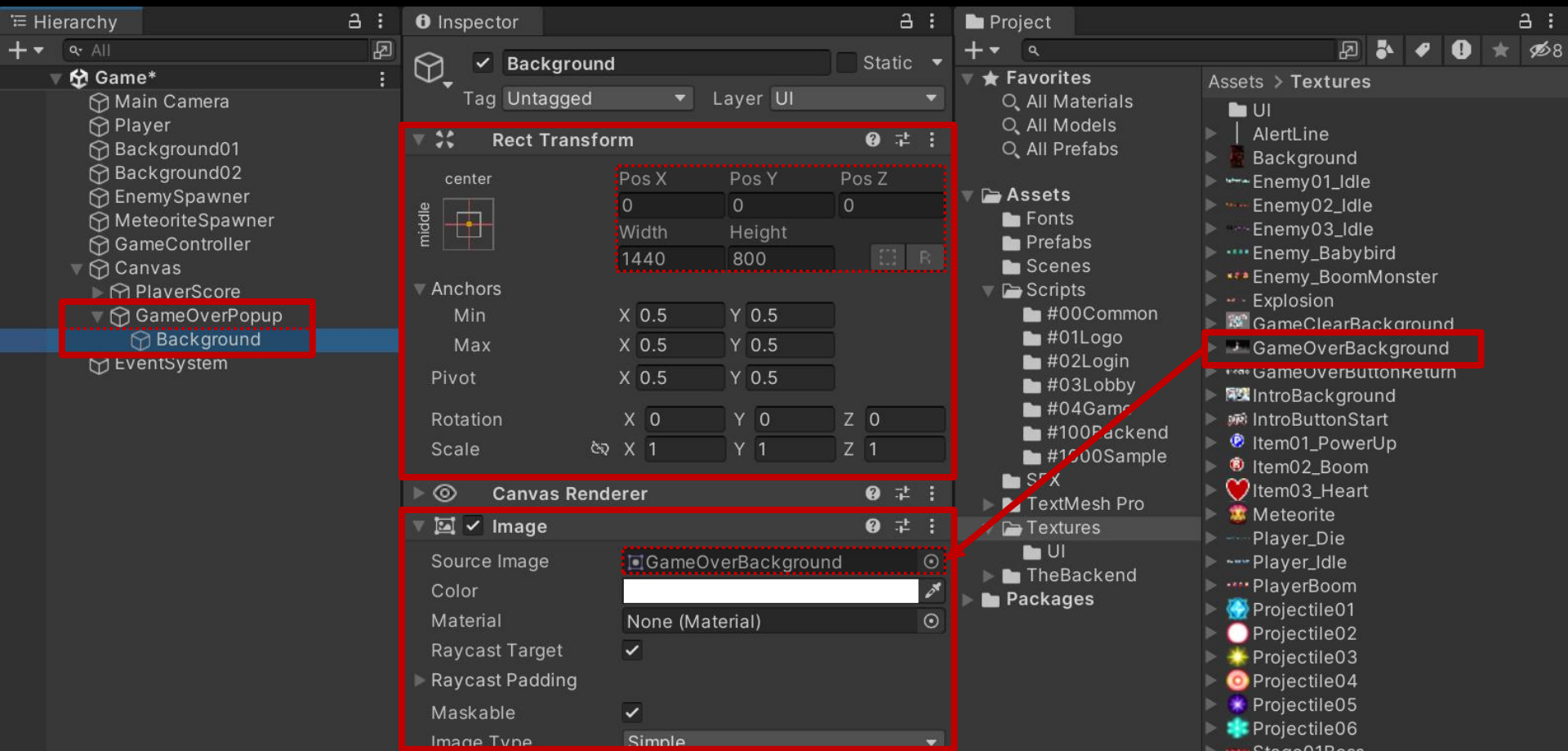
- 게임오버 UI들을 관리하는 Panel UI 생성 및 설정
  - GameObject - UI - Panel





# 게임오버 UI 구현

- 게임오버 배경을 출력하는 Image UI 생성 및 설정
  - GameObject - UI - Image





# 게임오버 UI 구현

- "GAME OVER" 텍스트를 출력하는 "Text - TextMeshPro" UI 생성 및 설정
  - GameObject - UI - "Text - TextMeshPro"

The screenshot shows the Unity Hierarchy and Inspector panels. In the Hierarchy, the 'GameOverText' object is selected under the 'GameOverPopup' folder. The Inspector panel displays the 'Rect Transform' component with the following values:

Property	Value
Left	0
Pos Y	-280
Right	0
Height	50

The 'Anchors' section shows the following values:

Property	X	Y
Min	0	0.5
Max	1	0.5

The 'TextMeshPro - Text (UI)' component is also visible in the Inspector panel.

The screenshot shows the 'TextMeshPro - Text (UI)' Inspector panel. The 'Text Input' field contains the text 'GAME OVER'. The 'Main Settings' section shows the following values:

Property	Value
Font Asset	NotoSansKR-Bold SDF (TMP_Fc)
Material Preset	NotoSansKR-Bold SDF Material
Font Style	B
Font Size	100

The 'Alignment' section shows the text is centered.



# 게임오버 UI 구현

- "SCORE" 텍스트를 출력하는 "Text - TextMeshPro" UI 생성 및 설정
  - GameObject - UI - "Text - TextMeshPro"

The screenshot shows the Unity Hierarchy and Inspector panels. In the Hierarchy, the 'ScoreText' object is selected under 'GameOverPopup'. The Inspector shows the 'Rect Transform' component with a stretch handle and the 'Anchors' component with Min and Max values set to X 0, Y 0.5 and X 1, Y 0.5 respectively. The 'TextMeshPro - Text (UI)' component is also visible.

The screenshot shows the TextMeshPro - Text (UI) component inspector. The 'Text Input' field contains 'SCORE'. The 'Main Settings' section shows 'Font Asset' set to 'NotoSansKR-Bold SDF (TMP\_Fc)', 'Font Style' set to 'B', and 'Font Size' set to '80'. The 'Alignment' section shows the text is centered.



# 게임오버 UI 구현

- 획득한 점수를 출력하는 "Text - TextMeshPro" UI 생성 및 설정
  - GameObject - UI - "Text - TextMeshPro"

Hierarchy

- Game\*
  - Main Camera
  - Player
  - Background01
  - Background02
  - EnemySpawner
  - MeteoriteSpawner
  - GameController
  - Canvas
    - PlayerScore
    - GameOverPopup
      - Background
      - GameOverText
      - ScoreText
      - ResultScore
  - EventSystem

Inspector

ResultScore

Tag Untagged Layer UI

Rect Transform

stretch	Left	Pos Y	Pos Z
	0	500	0
middle	Right	Height	
	0	50	

Anchors

Min	X 0	Y 0.5
Max	X 1	Y 0.5
Pivot	X 0.5	Y 0.5

Rotation

X 0	Y 0	Z 0
-----	-----	-----

Scale

X 1	Y 1	Z 1
-----	-----	-----

Canvas Renderer

TextMeshPro - Text (UI)

NotoSansKR-Bold SDF Material (Material)

Shader TextMeshPro/Distance Fi

Add Component

TextMeshPro - Text (UI)

Text Input

00000

Text Style Normal

Main Settings

Font Asset NotoSansKR-Bold SDF (TMP\_Fc...

Material Preset NotoSansKR-Bold SDF Material

Font Style B I U S ab AB SC

Font Size 80

Auto Size

Vertex Color

Color Gradient

Override Tags

Spacing Options (em) Character 0 Word 0 Line 0 Paragraph 0

Alignment

Wrapping Enabled

Overflow Overflow

Horizontal Mapping Character

Vertical Mapping Character





# 게임오버 UI 구현

- 로비로 이동하는 "Button - TextMeshPro" UI 생성 및 설정
  - GameObject - UI - "Button - TextMeshPro"

The screenshot displays the Unity development environment. On the left, the Hierarchy panel shows a tree structure under the 'Game' root. The 'GoToLobby' object is selected and highlighted with a blue bar. A red box highlights the 'GameOverPopup' folder, which contains 'Background', 'GameOverText', 'ScoreText', 'ResultScore', 'GoToLobby', and 'Text (TMP)'. On the right, the Inspector panel shows the 'GoToLobby' object selected. The 'Rect Transform' component is expanded, showing the following settings:

Property	Value
center	center
Pos X	0
Pos Y	-600
Pos Z	0
Width	800
Height	200
Min X	0.5
Min Y	0.5
Max X	0.5
Max Y	0.5
Pivot X	0.5
Pivot Y	0.5
Rotation X	0
Rotation Y	0
Rotation Z	0
Scale X	1
Scale Y	1
Scale Z	1

Below the 'Rect Transform' component, the 'Canvas Renderer', 'Image', and 'Button' components are visible. The 'Image' and 'Button' components are checked. The 'Default UI Material (Material)' is also visible, with a 'Shader' dropdown set to 'UI/Default'.



# 게임오버 UI 구현

- 로비로 이동하는 "Button - TextMeshPro" UI 생성 및 설정 (계속)

The screenshot displays the Unity development environment. On the left, the Hierarchy panel shows a tree view of the scene objects. The 'Text (TMP)' object is highlighted with a red box. On the right, the Inspector panel shows the properties of the selected 'TextMeshPro - Text (UI)' component. The text content is '로비로 이동'. The 'Font Asset' is set to 'NotoSansKR-Bold SDF (TMP\_Fc)'. The 'Font Style' is set to 'B' (Bold). The 'Font Size' is set to 80. The 'Text Style' is set to 'Normal'. The 'Main Settings' section is also visible, showing the material preset and other options.

**Hierarchy Panel:**

- Game
  - Main Camera
  - Player
  - Background01
  - Background02
  - EnemySpawner
  - MeteoriteSpawner
  - GameController
  - Canvas
    - PlayerScore
    - GameOverPopup
      - Background
      - GameOverText
      - ScoreText
      - ResultScore
      - GoToLobby
    - Text (TMP)**
  - EventSystem

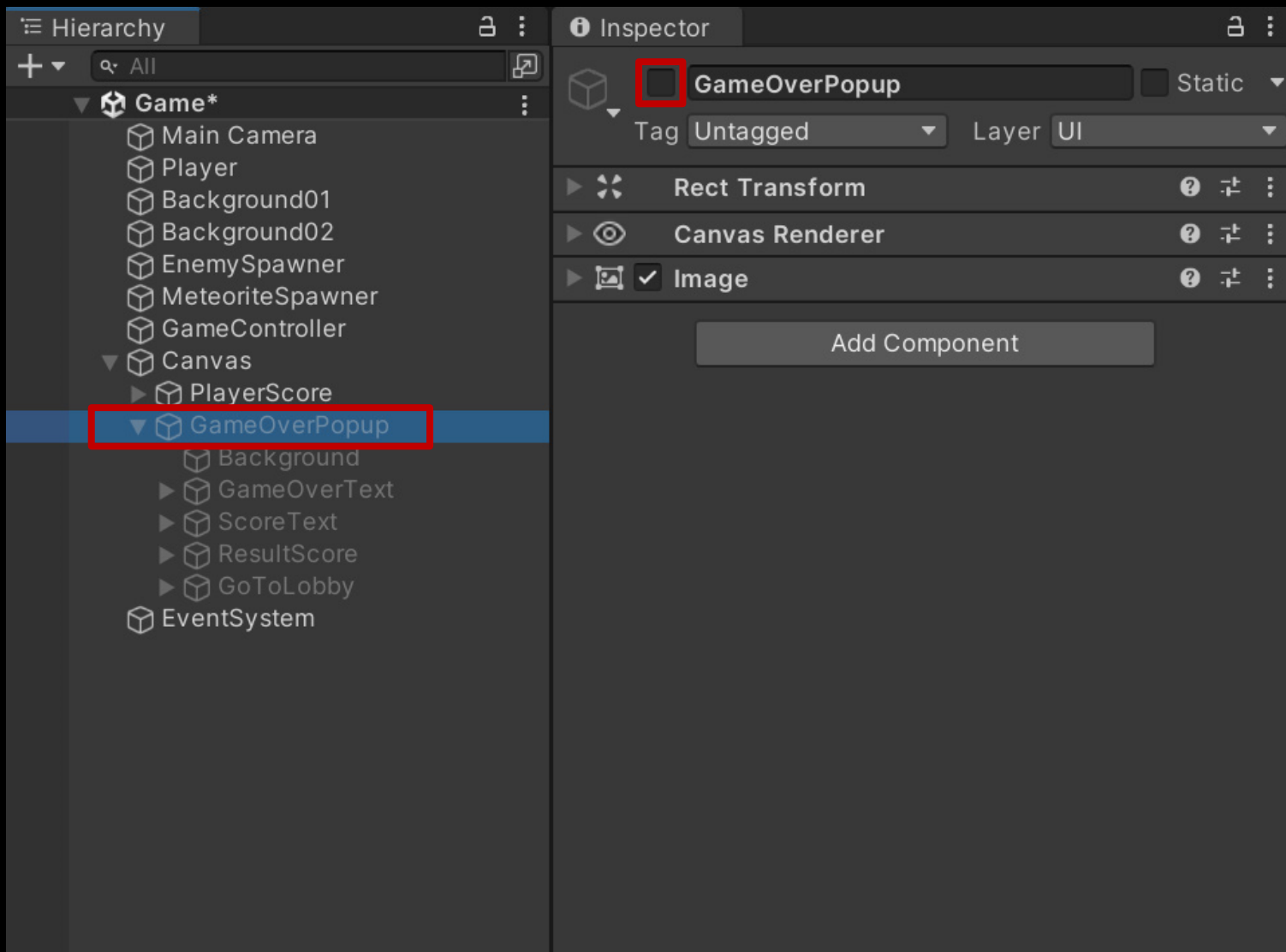
**Inspector Panel:**

- Text (TMP) [Static]
- Tag: Untagged, Layer: UI
- Rect Transform
- Canvas Renderer
- TextMeshPro - Text (UI)**
  - Text Input: Enable RTL Editor [ ]
  - Text: 로비로 이동
  - Text Style: Normal
  - Main Settings
    - Font Asset: **F** NotoSansKR-Bold SDF (TMP\_Fc)
    - Material Preset: NotoSansKR-Bold SDF Material
    - Font Style: **B** | I | U | S | ab | AB | SC
    - Font Size: 80
    - Auto Size: [ ]
    - Vertex Color: [ ]
    - Color Gradient: [ ]



# 게임오버 UI 구현

- GameOverPopup 오브젝트 비활성화





# 게임오버 UI 구현

- 게임오버 되었을 때 호출하는 OnGameOver() 메소드 정의
  - GameController Script 수정

```
1  using UnityEngine;
2  using TMPro;
3
4  public class GameController : MonoBehaviour
5  {
6      [SerializeField]
7      private GameController gameController;
8
9      [Header("InGame")]
10     [SerializeField]
11     private TextMeshProUGUI textScore;
12
13     [Header("Game Over")]
14     [SerializeField]
15     private GameObject panelGameOver;
16     [SerializeField]
17     private TextMeshProUGUI textResultScore;
18 }
```



# 게임오버 UI 구현

- GameUIController Script 수정

```
19 private void Update()...
23
24 public void OnGameOver()
25 {
26     // GameOver Panel UI 활성화
27     panelGameOver.SetActive(true);
28     // 획득 점수 출력
29     textResultScore.text = gameController.Score.ToString();
30 }
31
32 public void BtnClickGoToLobby()
33 {
34     Utils.LoadScene(SceneNames.Lobby);
35 }
36 }
```



# 게임오버 UI 구현

## ■ GameController 오브젝트의 컴포넌트 변수 설정

The image shows the Unity Inspector window with the following configurations:

- Hierarchy Panel:** Shows the scene hierarchy. The **GameController** object is selected and highlighted with a red box. Other objects like **GameOverPopup**, **ResultScore**, and **GoToLobby** are also highlighted with red boxes.
- Inspector Panel:** Shows the **GameController** component. The **Game Controller (Script)** component is expanded, showing the **On Game Over ()** event. The event is configured with the following settings:
  - Runtime Only
  - GameObject.SetActive
  - Meteorite (checked)
  - EnemySp (checked)
  - Player (checked)
  - GameUIController.OnGameOver (checked)
  - GameCon (checked)
- Game UI Controller (Script) Component:** Shows the **Game UI Controller** component. The **Game Controller** is set to **GameController (Game Controlle**. The **InGame** section has **Text Score** set to **PlayerScore (Text Mesh Pro UGL**. The **Game Over** section has **Panel Game Over** set to **GameOverPopup** and **Text Result Score** set to **ResultScore (Text Mesh Pro UGL**.



# 게임오버 UI 구현

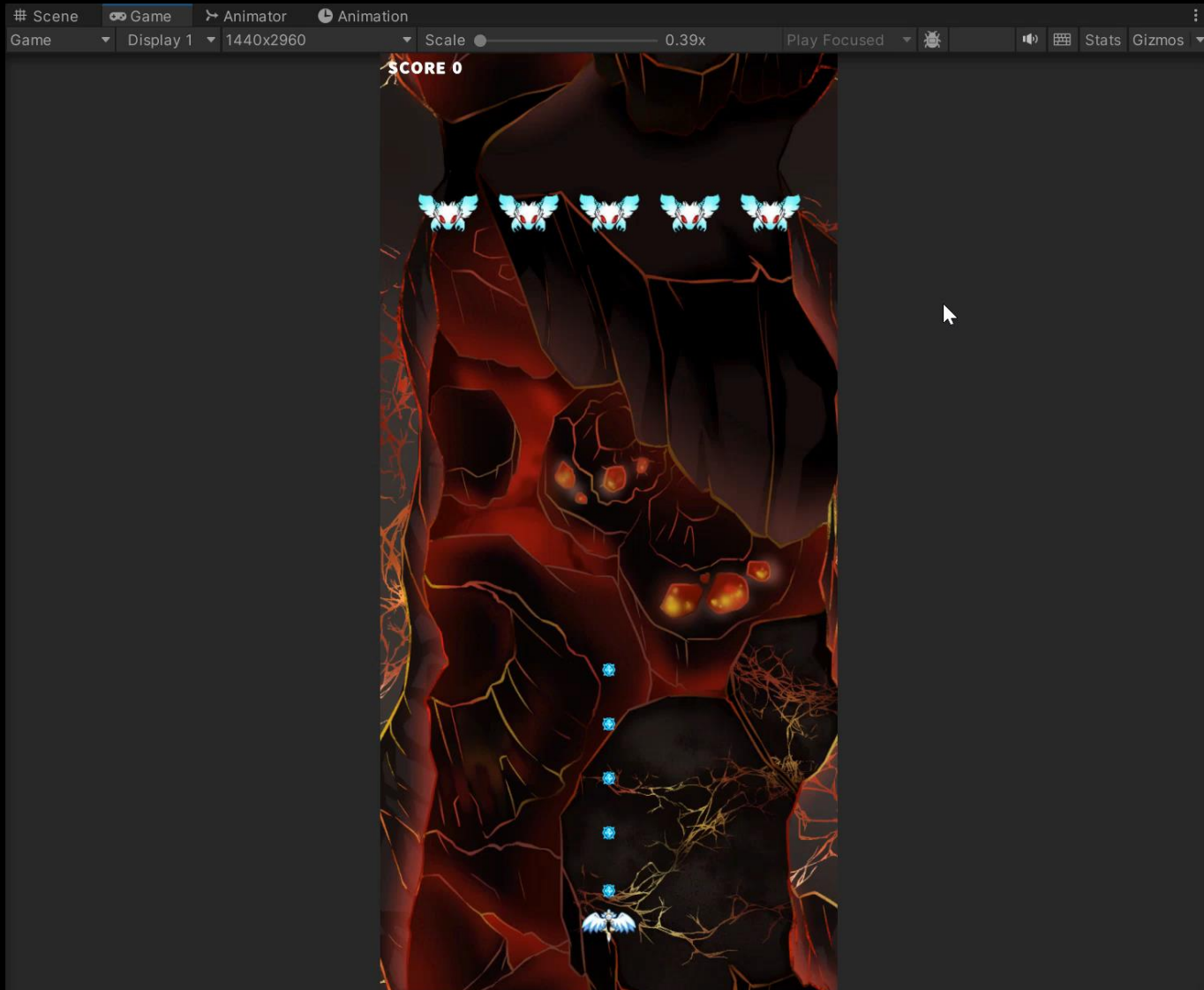
- GotoLobby 오브젝트의 "Button" 컴포넌트 OnClick 이벤트 등록

The screenshot displays the Unity Hierarchy and Inspector panels. In the Hierarchy panel, the **GoToLobby** object is selected under the **Canvas** hierarchy. The Inspector panel shows the **Button** component configuration for this object. The **OnClick ()** event list contains one entry: **Runtime Only** (indicated by a red dashed box) with the target **GameUIController.BtnClickGoToLobby** (also indicated by a red dashed box) and the method **GameCont** (indicated by a red dashed box). A red arrow points from the **GoToLobby** object in the Hierarchy to the **Button** component in the Inspector.



# 게임오버 UI 구현

## ■ 결과 화면







# 게임오버 UI 구현

## ■ Scale Effect

- Text UI의 크기 애니메이션을 제어하는 스크립트 생성 및 작성
  - C# Script 생성 후 스크립트의 이름을 "ScaleEffect"로 변경

```
1  using System.Collections;
2  using UnityEngine;
3  using TMPro;
4
5  public class ScaleEffect : MonoBehaviour
6  {
7      [SerializeField]
8      [Range(0.01f, 10f)]
9      private float      effectTime;          // 크기 확대/축소 되는 시간
10     private TextMeshProUGUI effectText;     // 크기 확대/축소 효과에 사용되는 텍스트
11
12     private void Awake()
13     {
14         effectText = GetComponent<TextMeshProUGUI>();
15     }
16
```



# 게임오버 UI 구현

- Text UI의 크기 애니메이션을 제어하는 스크립트 생성 및 작성 (계속)

```
17 public void Play(float start, float end)
18 {
19     StartCoroutine(Process(start, end));
20 }
21
22 private IEnumerator Process(float start, float end)
23 {
24     float current = 0;
25     float percent = 0;
26
27     while ( percent < 1 )
28     {
29         current += Time.deltaTime;
30         percent = current / effectTime;
31
32         effectText.fontSize = Mathf.Lerp(start, end, percent);
33
34         yield return null;
35     }
36 }
37 }
```



# 게임오버 UI 구현

- GameOverText 오브젝트에 "ScaleEffect" 컴포넌트 추가 및 설정

The screenshot displays the Unity Hierarchy and Inspector panels. In the Hierarchy panel, the 'GameOverText' object is selected and highlighted with a red box. The Inspector panel shows the 'Scale Effect (Script)' component added to the selected object. The 'Effect Time' property is set to 0.5, also highlighted with a red dashed box. The 'Add Component' button is visible at the bottom of the Inspector panel.

**Hierarchy Panel:**

- Game\*
- Main Camera
- Player
- Background01
- Background02
- EnemySpawner
- MeteoriteSpawner
- GameController
- Canvas
  - PlayerScore
  - GameOverPopup
    - Background
    - GameOverText**
    - Score Text
    - ResultScore
  - GoToLobby
- EventSystem

**Inspector Panel:**

- GameObject: GameOverText
- Tag: Untagged
- Layer: UI
- Rect Transform
- Canvas Renderer
- TextMeshPro - Text (UI)
- Scale Effect (Script)**
  - Script: ScaleEffect
  - Effect Time: 0.5
- Add Component



# 게임오버 UI 구현

- "GAME OVER" 텍스트 크기 축소 애니메이션 재생
  - GameController Script 수정

```
1  using UnityEngine;
2  using TMPro;
3
4  public class GameController : MonoBehaviour
5  {
6      [SerializeField]
7      private GameController  gameController;
8
9      [Header("InGame")]
10     [SerializeField]
11     private TextMeshProUGUI textScore;
12
13     [Header("Game Over")]
14     [SerializeField]
15     private GameObject      panelGameOver;
16     [SerializeField]
17     private TextMeshProUGUI textResultScore;
18
19     [Header("Game Over UI Animation")]
20     [SerializeField]
21     private ScaleEffect      effectGameOver;
22
```



# 게임오버 UI 구현

- GameUIController Script 수정 (계속)

```
23  + private void Update()...
27
28  - public void OnGameOver()
29  {
30      // GameOver Panel UI 활성화
31      panelGameOver.SetActive(true);
32      // 획득 점수 출력
33      textResultScore.text = gameController.Score.ToString();
34      // "GAME OVER" 텍스트 크기 축소 애니메이션
35      effectGameOver.Play(200, 100);
36  }
37
38  + public void BtnClickGoToLobby()...
42 }
```



# 게임오버 UI 구현

- GameController 오브젝트의 "GameUIController" 컴포넌트 변수 설정

The screenshot displays the Unity Inspector window for the 'GameController' object. The Hierarchy panel on the left shows the object structure, with 'GameController' and 'GameOverText' highlighted by red boxes. The Inspector panel on the right shows the 'Game UI Controller (Script)' component, which is also highlighted by a red box. The component's variables are configured as follows:

- Script:** GameUIController
- Game Controller:** GameController (Game Controller)
- InGame Text Score:** PlayerScore (Text Mesh Pro UGI)
- Game Over Panel Game Over:** GameOverPopup
- Game Over Text Result Score:** ResultScore (Text Mesh Pro UGI)
- Game Over UI Animation Effect Game Over:** GameOverText (Scale Effect)

An arrow points from the 'GameOverText' box in the Hierarchy panel to the 'GameOverText (Scale Effect)' variable in the Inspector panel. The 'Add Component' button is visible at the bottom of the Inspector panel.



# 게임오버 UI 구현

## ■ Counting Effect

- 점수 카운팅 애니메이션을 제어하는 스크립트 생성 및 작성
  - C# Script 생성 후 스크립트의 이름을 "CountingEffect"로 변경

```
1  using System.Collections;
2  using UnityEngine;
3  using TMPro;
4
5  public class CountingEffect : MonoBehaviour
6  {
7      [SerializeField]
8      [Range(0.01f, 10f)]
9      private float      effectTime;           // 카운팅 되는 시간
10     private TextMeshProUGUI effectText;      // 카운팅 효과에 사용되는 텍스트
11
12     private void Awake()
13     {
14         effectText = GetComponent<TextMeshProUGUI>();
15     }
16
```



# 게임오버 UI 구현

- 점수 카운팅 애니메이션을 제어하는 스크립트 생성 및 작성 (계속)

```
17 public void Play(int start, int end)
18 {
19     StartCoroutine(Process(start, end));
20 }
21
22 private IEnumerator Process(int start, int end)
23 {
24     float current = 0;
25     float percent = 0;
26
27     while ( percent < 1 )
28     {
29         current += Time.deltaTime;
30         percent = current / effectTime;
31
32         effectText.text = Mathf.Lerp(start, end, percent).ToString("F0");
33
34         yield return null;
35     }
36 }
37 }
```





# 게임오버 UI 구현

- ResultScore 오브젝트에 "CountingEffect" 컴포넌트 추가 및 설정

The screenshot displays the Unity development environment. On the left, the Hierarchy panel shows a tree view of the scene objects. The 'ResultScore' object is selected and highlighted with a red box. On the right, the Inspector panel shows the properties of the selected object. The 'Counting Effect (Script)' component is highlighted with a red box. The 'Effect Time' property is set to 3, also highlighted with a red box. The 'Add Component' button is visible at the bottom of the Inspector panel.

**Hierarchy Panel:**

- Game\*
  - Main Camera
  - Player
  - Background01
  - Background02
  - EnemySpawner
  - MeteoriteSpawner
  - GameController
  - Canvas
    - PlayerScore
    - GameOverPopup
      - Background
      - GameOverText
      - ScoreText
    - ResultScore**
    - GoToLobby
  - EventSystem

**Inspector Panel:**

- ResultScore (Static)
  - Tag: Untagged
  - Layer: UI
  - Rect Transform
  - Canvas Renderer
  - TextMeshPro - Text (UI)
  - Counting Effect (Script)**
    - Script: CountingEffect
    - Effect Time: 3
  - Add Component



# 게임오버 UI 구현

- 획득 점수 카운팅 애니메이션 재생
  - GameController Script 수정

```
1  +using ...
2
3
4  public class GameController : MonoBehaviour
5  {
6      [SerializeField]
7      private GameController  gameController;
8
9      [Header("InGame")]
10     [SerializeField]
11     private TextMeshProUGUI textScore;
12
13     [Header("Game Over")]
14     [SerializeField]
15     private GameObject      panelGameOver;
16     [SerializeField]
17     private TextMeshProUGUI textResultScore;
18
19     [Header("Game Over UI Animation")]
20     [SerializeField]
21     private ScaleEffect      effectGameOver;
22     [SerializeField]
23     private CountingEffect   effectResultScore;
24 }
```



# 게임오버 UI 구현

## □ GameController Script 수정

```
25 private void Update()...
29
30 public void OnGameOver()
31 {
32     // GameOver Panel UI 활성화
33     panelGameOver.SetActive(true);
34     // 획득 점수 출력
35     textResultScore.text = gameController.Score.ToString();
36     // "GAME OVER" 텍스트 크기 축소 애니메이션
37     effectGameOver.Play(200, 100);
38     // 0 -> gameController.Score까지 점수를 카운팅하는 애니메이션
39     effectResultScore.Play(0, gameController.Score);
40 }
41
42 public void BtnClickGoToLobby()...
46 }
```



# 게임오버 UI 구현

- GameController 오브젝트의 "GameUIController" 컴포넌트 변수 설정

The screenshot displays the Unity Hierarchy and Inspector panels. In the Hierarchy panel, the **GameController** object is selected and highlighted with a red box. In the Inspector panel, the **Game UI Controller (Script)** component is selected and highlighted with a red box. The Inspector panel shows the following configuration:

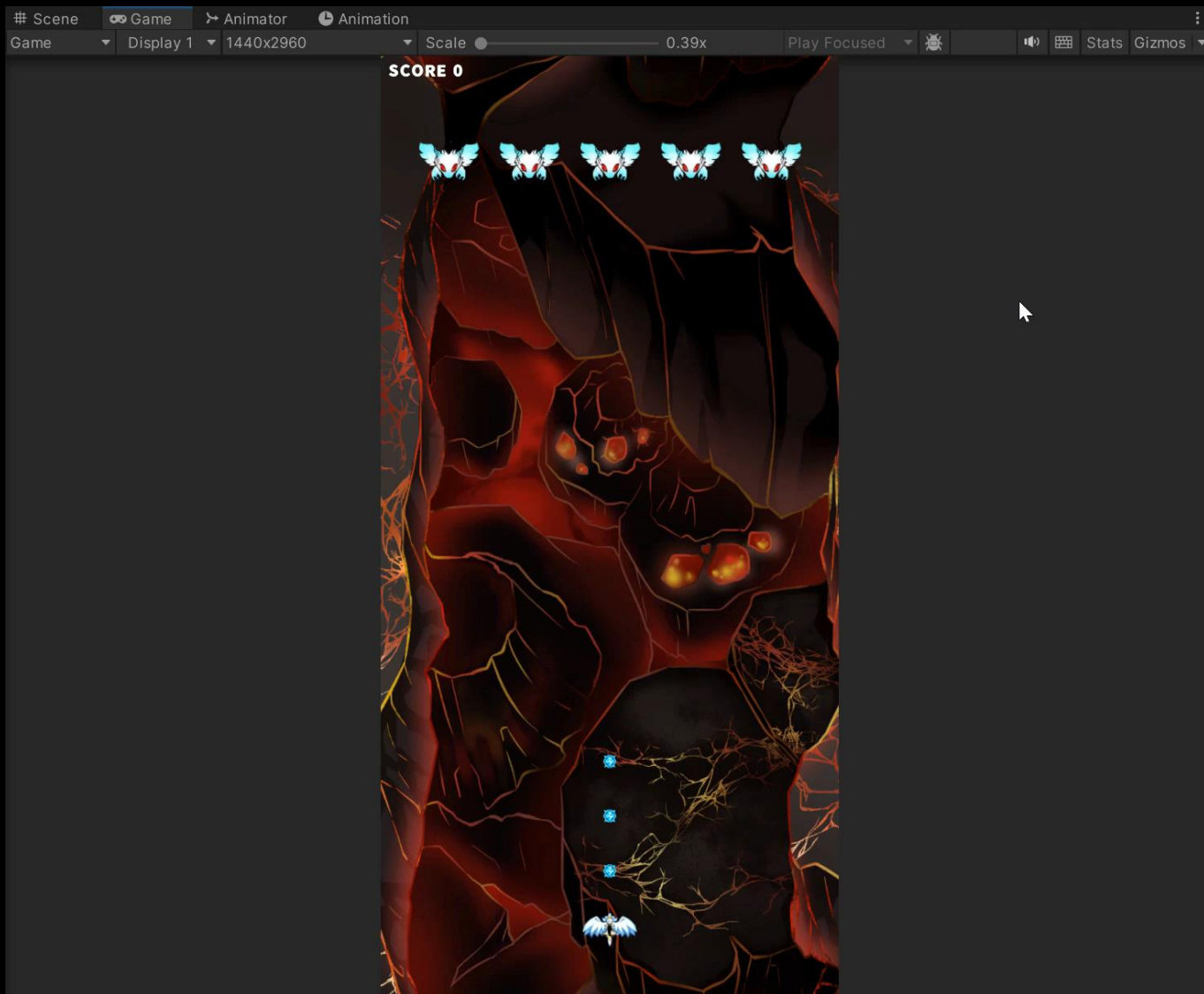
- Script:** GameUIController
- Game Controller:** GameController (Game Controller)
- InGame:**
  - Text Score:** PlayerScore (Text Mesh Pro UGI)
- Game Over:**
  - Panel Game Over:** GameOverPopup
  - Text Result Score:** ResultScore (Text Mesh Pro UGI)
- Game Over UI Animation:**
  - Effect Game Over:** GameOverText (Scale Effect)
  - Effect Result Score:** ResultScore (Counting Effect)

A red arrow points from the **ResultScore** object in the Hierarchy panel to the **Effect Result Score** field in the Inspector panel, indicating the assignment of the variable.



# 게임오버 UI 구현

## ■ 결과 화면



# 랭킹으로 사용할 유저 정보 데이터 추가

- 랭킹 등록을 위해 필요한 기능들
- 일일 최고 점수 데이터 추가
- 랭킹 생성



# 랭킹으로 사용할 유저 정보 데이터 추가

## ■ 랭킹 등록을 위해 필요한 기능들

- 계정 생성 / 로그인 시스템을 이용해 등록한 유저

Backnd Console << ProjectA [Settings] [Logout] [Admin] [Profile]

ProjectA [Alert] [Refresh] [Help] [SDK 문서] [문서 가이드]

★ 유저 관리 [계정 유저 생성] [삭제]

대상: 회원번호 [Input Field]

상세 검색 ▶

[검색] [검색 조건 초기화]

<input type="checkbox"/>	번호	회원번호	회원 아이디	닉네임	가입일	최종 접속일	국가	접속 OS	<input type="checkbox"/>
<input type="checkbox"/>	1	<a href="#">2d622750-04f5-11ee-a3c4-73805164c040</a>	user02	-	2023.06.07 14:35	2023.06.07 14:35	-	Windows 10 &#4010...	<input type="checkbox"/>
<input type="checkbox"/>	2	<a href="#">7814bb60-04f4-11ee-bb76-8582e94b5d40</a>	user01	고박사	2023.06.07 14:30	2023.06.07 16:15	-	Windows 10 &#4010...	<input type="checkbox"/>

< 1 > [10개씩 보기]

ProjectA [Alert] [Refresh] [Help] [SDK 문서] [문서 가이드]

- ★ 즐겨찾기
- 대시보드
- 서버 설정
- 프로젝트
- 인증 정보
- 푸시
- 스토어 정보
- 소셜
- 예산 알림
- 요금제
- 뒤끝베이스
- 유저 관리**
- 유저 접근 관리
- 게임 정보 관리
- 랭킹 관리
- 우편 관리
- 푸시 관리
- 쿠폰 관리
- 차트 관리
- 확률 관리
- 로그 관리
- 영수증 검증



# 랭킹으로 사용할 유저 정보 데이터 추가

- 랭킹 데이터를 저장하기 위한 private 테이블

Backnd Console ProjectA

★ 게임 정보 관리 테이블 생성 삭제

전체 테이블명으로 검색 SDK 문서 콘솔 가이드

테이블 데이터

\*테이블은 최대 100개까지 생성될 수 있습니다.

테이블명	설명	분류	스키마 정의	상태
<input type="checkbox"/> USER_DATA		private	스키마 미정의	활성

1 10개씩 보기





# 랭킹으로 사용할 유저 정보 데이터 추가

- 테이블 내부에 레벨, 점수 등과 같이 순위의 기준이 되는 숫자형 데이터 row

Backnd Console ProjectA

ProjectA **게임 정보 관리** **랭킹 생성** **삭제** **테이블 초기화** SDK 문서 **콘솔 가이드**

테이블 데이터

테이블 선택: USER\_DATA

상세 검색

검색 검색 조건 초기화

r_id	inDate	owner_inDate	updatedAt	client_date	exp...	gold	heart	jewel	level
<a href="#">1c20-04f2-11ee-a716-9db41e8f5999</a>	2023-06-07T05:19:14.305Z	2023-06-07T05:19:13.634Z	2023-06-07T05:19:14.305Z	2023-06-07T05:19:13.192Z	0	0	30	0	1
<a href="#">3940-04f2-11ee-b4fa-a998ca032138</a>	2023-06-07T05:18:59.437Z	2023-06-07T05:18:58.772Z	2023-06-07T05:18:59.437Z	2023-06-07T05:18:58.352Z	0	0	30	0	1
<a href="#">8290-04f2-11ee-87d2-03970512c2d3</a>	2023-06-07T05:18:42.149Z	2023-06-07T05:18:41.465Z	2023-06-07T05:18:42.149Z	2023-06-07T05:18:41.070Z	0	0	30	0	1

< 1 > 10개씩 보기



# 랭킹으로 사용할 유저 정보 데이터 추가

- 랭킹을 생성할 때 private 테이블의 "숫자형" 데이터를 컬럼으로 사용 가능

Backnd Console

ProjectA

랭킹 관리

랭킹 생성

삭제

신규 랭킹(SDK 5.4.0 이상)

전체

랭킹명으로 검색

SDK 문서

콘솔 가이드

관리자

ProjectA

랭킹 보상

유형

랭킹명

생성한 랭킹이 없습니다.

랭킹 생성

\*새로운 랭킹을 사용하기 위해서는 SDK 5.4.0 이상이 필요합니다.

유형

유저 랭킹

길드 랭킹

랭킹명\*

초기화 기간

일간

주간

월간

누적랭킹

일회성 랭킹

랭킹 종료 시 컬럼 초기화

적용

미적용

랭킹 항목

테이블

USER\_DATA

컬럼

experience

experience

level

jewel

heart

gold

추가 항목

정렬 기준

랭킹 보상

보상 우편 제목

확인

취소

Tip. 랭킹 생성/등록을 하려면 "게임 정보 기능"으로 등록된 private 테이블의 "숫자형" 데이터가 있어야 함

회사소개

이용약관

서비스수준협약

개인정보처리방침



# 랭킹으로 사용할 유저 정보 데이터 추가

## ■ 일일 최고 점수 데이터 추가

### ■ 일일 최고 점수를 저장하는 정수형 변수 선언

#### □ UserGameData Script 수정

```
1 [System.Serializable]
2 public class UserGameData
3 {
4     public int level; // Lobby Scene에 보이는 플레이어 레벨
5     public float experience; // Lobby Scene에 보이는 플레이어 경험치
6     public int gold; // 무료 재화
7     public int jewel; // 유료 재화
8     public int heart; // 게임 플레이에 소모되는 재화
9     public int dailyBestScore; // 일일 최고 점수
10
11     public void Reset()
12     {
13         level = 1;
14         experience = 0;
15         gold = 0;
16         jewel = 0;
17         heart = 30;
18         dailyBestScore = 0;
19     }
20 }
```



# 랭킹으로 사용할 유저 정보 데이터 추가

- 유저 정보 추가/불러오기/갱신에 일일 최고 점수 데이터 추가
  - BackendGameData Script 수정

```
30  // <summary> 뒤끝 콘솔 테이블에 새로운 유저 정보 추가
33  public void GameDataInsert()
34  {
35      // 유저 정보를 초기값으로 설정
36      userGameData.Reset();
37
38      // 테이블에 추가할 데이터로 가공
39      Param param = new Param()
40      {
41          { "level",      userGameData.level },
42          { "experience", userGameData.experience },
43          { "gold",      userGameData.gold },
44          { "jewel",     userGameData.jewel },
45          { "heart",     userGameData.heart },
46          { "dailyBestScore", userGameData.dailyBestScore }
47      };
48
49      // 첫 번째 매개변수는 뒤끝 콘솔의 "게임 정보 관리" 탭에 생성한 테이블 이름
50      Backend.GameData.Insert("USER_DATA", param, callback =>...);
66  }
67
```



# 랭킹으로 사용할 유저 정보 데이터 추가

## BackendGameData Script 수정 (계속)

```
68  // <summary> 뒤끝 콘솔 테이블에서 유저 정보를 불러올 때 호출
71  public void GameDataLoad()
72  {
73      Backend.GameData.GetMyData("USER_DATA", new Where(), callback =>
74      {
75          // 게임 정보 불러오기에 성공했을 때
76          if ( callback.IsSuccess() )
77          {
78              Debug.Log($"게임 정보 데이터 불러오기에 성공했습니다. : {callback}");
79
80              // JSON 데이터 파싱 성공
81              try
82              {
83                  LitJson.JsonData gameDataJson = callback.FlattenRows();
84
85                  // 받은 데이터의 개수가 0이면 데이터가 없는 것
86                  if ( gameDataJson.Count <= 0 )...
87                  else
88                  {
89                      // 불러온 게임 정보의 고유값
90                      gameDataRowInDate = gameDataJson[0]["inDate"].ToString();
91                      // 불러온 게임 정보를 userData 변수에 저장
92                      userGameData.level = int.Parse(gameDataJson[0]["level"].ToString());
93                      userGameData.experience = float.Parse(gameDataJson[0]["experience"].ToString());
94                      userGameData.gold = int.Parse(gameDataJson[0]["gold"].ToString());
95                      userGameData.jewel = int.Parse(gameDataJson[0]["jewel"].ToString());
96                      userGameData.heart = int.Parse(gameDataJson[0]["heart"].ToString());
97                      userGameData.dailyBestScore = int.Parse(gameDataJson[0]["dailyBestScore"].ToString());
98
99                      onGameDataLoadEvent?.Invoke();
100
101                  }
102              }
103          }
104      }
```



# 랭킹으로 사용할 유저 정보 데이터 추가

## BackendGameData Script 수정 (계속)

```
122 123 124 125 126 127 128 129 130 131 132 133 134 135 136 137 138 139 140 141 142 143 144 145 146 147 148 149 150 151 152 153 154 155 156 157 158 159 160 161 162 163 164 165 166 167 168 169 170
```

```
/// <summary> 뒤끝 콘솔 테이블에 있는 유저 데이터 갱신
public void GameDataUpdate(UnityAction action=null)
{
    if ( userGameData == null )...

    Param param = new Param()
    {
        { "level",          userGameData.level },
        { "experience",     userGameData.experience },
        { "gold",           userGameData.gold },
        { "jewel",          userGameData.jewel },
        { "heart",          userGameData.heart },
        { "dailyBestScore", userGameData.dailyBestScore }
    };

    // 게임 정보의 고유값(gameDataRowInDate)이 없으면 에러 메시지 출력
    if ( string.IsNullOrEmpty(gameDataRowInDate) )...
    // 게임 정보의 고유값이 있으면 테이블에 저장되어 있는 값 중 inDate 컬럼의 값과
    // 소유하는 유저의 owner_inDate가 일치하는 row를 검색하여 수정하는 UpdateV2() 호출
    else...
}
}
```



# 랭킹으로 사용할 유저 정보 데이터 추가

## ■ 기존 코드로 등록했던 유저 삭제

Backnd Console << ProjectA

ProjectA **기밀 모드**

★ 유저 관리 ◀ 게임 유저 생성 **삭제** SDK 문서 콘솔 가이드

대상 회원번호

상세 검색

검색 검색 조건 초기화

<input checked="" type="checkbox"/>	번호	회원번호	회원 아이디	닉네임	가입일	최종 접속일	국가	접속 OS
<input checked="" type="checkbox"/>	1	d7551c20-04f2-11ee-a716-9db41e8f5999	user03	-	2023.06.07 14:19	2023.06.07 14:19	-	Windows 10 &#4010...
<input checked="" type="checkbox"/>	2	ca795940-04f2-11ee-b4fa-a998ca03				2023.06.07 14:18	-	Windows 10 &#4010...
<input checked="" type="checkbox"/>	3	c4288290-04f2-11ee-87d2-03970512				2023.06.07 14:29	-	Windows 10 &#4010...

10개씩 보기

**게임 유저 삭제**

유저 데이터 삭제 여부\*  삭제함  삭제하지 않음

랭킹 삭제 여부\*  삭제함  삭제하지 않음

**dailyBestScore row가 없는 기존 유저를 모두 삭제하고, 게임을 실행해 "계정 생성"으로 새로운 유저 생성**



# 랭킹으로 사용할 유저 정보 데이터 추가

## ■ 새로 생성한 계정의 row 데이터

Backnd Console ProjectA

☆ 게임 정보 관리 | 행 생성 | 삭제 | 테이블 초기화

SDK 문서 | 콘솔 가이드

테이블 데이터

테이블 선택: USER\_DATA

상세 검색

검색 | 검색 조건 초기화

	inDate	owner_inDate	updatedAt	client_date	dail...	exp...	gold	heart	jewel	level
11ee-bb76-8582e94b5d40	2023-06-07T05:30:53.541Z	2023-06-07T05:30:52.822Z	2023-06-07T05:30:53.541Z	2023-06-07T05:30:52.426Z	0	0	0	30	0	1

< 1 > 10개씩 보기

게임 정보 관리

회사소개 | 이용약관 | 서비스수준협약 | 개인정보처리방침

© AFI, Inc. All rights reserved.





# 랭킹으로 사용할 유저 정보 데이터 추가

## 랭킹 생성

### 일일 최고 점수 랭킹 생성

The screenshot shows the Backnd Console interface with a modal dialog for creating a ranking. The dialog is titled '랭킹 생성' (Ranking Creation) and contains the following fields and options:

- 유형 (Type):** 유저 랭킹 (User Ranking) is selected.
- 랭킹명\* (Ranking Name):** DAILY\_RANK is entered.
- 초기화 기간 (Reset Period):** 일간 (Daily) is selected.
- 랭킹 종료 시 컬럼 초기화\* (Reset Column at Ranking End):** 적용 (Apply) is selected.
- 랭킹 항목\* (Ranking Item):** 테이블 (Table) is set to USER\_DATA and 컬럼 (Column) is set to dailyBestScore.
- 추가 항목 (Additional Item):** 없음 (None) is selected.
- 정렬 기준 (Sort Criteria):** 내림차순 (Descending) is selected.
- 랭킹 보상 (Ranking Reward):** 없음 (None) is selected.
- 보상 우편 제목 (Reward Email Subject):** (Empty field)

At the bottom of the dialog, there are two buttons: '확인' (Confirm) and '취소' (Cancel).



# 랭킹으로 사용할 유저 정보 데이터 추가

## ■ 일일 최고 점수 랭킹 생성 (계속)

Backnd Console ProjectA

랭킹 관리 | 랭킹 생성 | 삭제

신규 랭킹(SDK 5.4.0 이상) | 전체 | 랭킹명으로 검색 | SDK 문서 | 콘솔 가이드

랭킹	보상
<input type="checkbox"/> 유형	랭킹명 초기화 기간 랭킹 보상 UUID
<input type="checkbox"/> 유저	<b>DAILY_RANK</b> 1일 (UUID)

**DAILY\_RANK** 수정

항목 USER\_DATA > dailyBestScore 추가 항목 --

기간 2023.06.07 - 2023.06.07

UUID e795f580-04f4-11ee-958d-ad4ae0c79832

지난 랭킹 CSV 다운로드 삭제 회원번호 또는 닉네임을 입력해

순위	회원번호	닉네임	스코어	추가 항목
등록된 내용이 없습니다.				

확인

회사소개 이용약관 서비스수준협약 개인정보처리방침 © AFI, Inc. All rights reserved.

# 랭킹 데이터 등록

- 랭킹 데이터 등록
- 랭킹 데이터 확인 및 갱신



# 랭킹 데이터 등록

- 랭킹 데이터 등록
  - 랭킹 테이블의 UUID

Backnd Console << ProjectA

신규 랭킹(SDK 5.4.0 이상) 전체 랭킹명으로 검색 SDK 문서 콘솔 가이드

랭킹 보상

유형	랭킹명	초기화 기간	랭킹 보상	UUID
<input type="checkbox"/>	유저	DAILY_RANK	1일	<input type="text" value="e795f580-04f4-11ee-958d-ad4ae0c79832"/>

**DAILY\_RANK** 수정

항목 USER\_DATA > dailyBestScore 추가 항목

기간 2023.06.07 - 2023.06.07

**UUID e795f580-04f4-11ee-958d-ad4ae0c79832**

회원번호 또는 닉네임을 입력해

순위	회원번호	닉네임	스코어	추가 항목
등록된 내용이 없습니다.				

확인

**Tip. 랭킹 테이블이 여러 개 일 수 있기 때문에 각 랭킹 테이블의 UUID를 기반으로 랭킹 테이블에 접근**



# 랭킹 데이터 등록

- 공용으로 사용하는 상수를 선언하는 스크립트 생성 및 작성
  - C# Script 생성 후 스크립트의 이름을 "Constants"로 변경

```
1 public static class Constants
2 {
3     public static readonly string USER_DATA_TABLE = "USER_DATA";
4     public static readonly string DAILY_RANK_UUID = "e795f580-04f4-11ee-958d-ad4ae0c79832";
5 }
```



# 랭킹 데이터 등록

- 일일 최고 점수를 랭킹에 등록하는 스크립트 생성 및 작성
  - C# Script 생성 후 스크립트의 이름을 "DailyRankRegister"로 변경

```
1  using BackEnd;
2  using UnityEngine;
3
4  public class DailyRankRegister : MonoBehaviour
5  {
6      public void Process(int newScore)
7      {
8          UpdateMyRankData(newScore);
9      }
10
11     private void UpdateMyRankData(int newScore)
12     {
13         string rowInDate = string.Empty;
14
15         // 랭킹 데이터를 업데이트하려면 게임 데이터에서 사용하는 데이터의 inDate 값이 필요
16         Backend.GameData.GetMyData(Constants.USER_DATA_TABLE, new Where(), callback =>
17         {
18             if ( !callback.IsSuccess() )
19             {
20                 Debug.LogError($"데이터 조회 중 문제가 발생했습니다 : {callback}");
21                 return;
22             }
23         }
```



# 랭킹 데이터 등록

- 일일 최고 점수를 랭킹에 등록하는 스크립트 생성 및 작성 (계속)

```
24 Debug.Log($"데이터 조회에 성공했습니다 : {callback}");
25
26 if ( callback.FlattenRows().Count > 0 )
27 {
28     rowInDate = callback.FlattenRows()[0]["inDate"].ToString();
29 }
30 else
31 {
32     Debug.LogError("데이터가 존재하지 않습니다.");
33     return;
34 }
35
36 Param param = new Param()
37 {
38     { "dailyBestScore", newScore }
39 };
40
41 Backend.URank.User.UpdateUserScore(Constants.DAILY_RANK_UUID, Constants.USER_DATA_TABLE, rowInDate, param, callback =>
42 {
43     if ( callback.IsSuccess() )
44     {
45         Debug.Log($"랭킹 등록에 성공했습니다 : {callback}");
46     }
47     else
48     {
49         Debug.LogError($"랭킹 등록 중 오류가 발생했습니다 : {callback}");
50     }
51 });
52 });
53 }
54 }
```

== backend method ==

Backend.URank.User.UpdateUserScore(string rankUuid, string tableName, string rowInDate, Param param);  
tableName 테이블의 rowInDate row 데이터를 param 값으로 갱신하고, rankUuid 랭킹 테이블의 랭킹 정보 갱신



# 랭킹 데이터 등록

- 게임오버 되었을 때 score를 랭킹에 등록

- GameController Script 수정

```
4 public class GameController : MonoBehaviour
5 {
6     [SerializeField]
7     private UnityEvent onGameOver; // 게임오버 되었을 때 호출할 메소드 등록 및 실행
8     [SerializeField]
9     private DailyRankRegister dailyRank;
10
11     private int score = 0;
12
13     public bool IsGameOver { set; get; } = false;
14     public int Score...
15
16
17
18
19
20     public void GameOver()
21     {
22         // 중복 처리 되지 않도록 bool 변수로 제어
23         if ( IsGameOver == true ) return;
24
25         IsGameOver = true;
26
27         // 게임오버 되었을 때 호출할 메소드들을 실행
28         onGameOver.Invoke();
29
30         // 현재 점수 정보를 바탕으로 랭킹 데이터 갱신
31         dailyRank.Process(score);
32
33         // 경험치 증가 및 레벨업 여부 검사
34         // (현재 레벨 시스템에 대한 설정이 없기 때문에 경험치의 최대치를 100으로 가정)
35         // (게임을 한번 플레이할 때마다 경험치는 25씩 증가)
36         BackendGameData.Instance.UserGameData.experience += 25;
37         if ( BackendGameData.Instance.UserGameData.experience >= 100 )...
38
39
40
41
42
43         // 게임 정보 업데이트
44         BackendGameData.Instance.GameDataUpdate();
45     }
46 }
```





# 랭킹 데이터 등록

- GameController 오브젝트에 "DailyRankRegister" 컴포넌트 추가 및 설정

The screenshot displays the Unity Hierarchy and Inspector panels. In the Hierarchy panel, the **GameController** object is selected and highlighted with a red box. A red arrow points from this box to the Inspector panel. In the Inspector panel, the **Game Controller (Script)** component is expanded, showing its configuration. The **On Game Over ()** event is configured with the following actions:

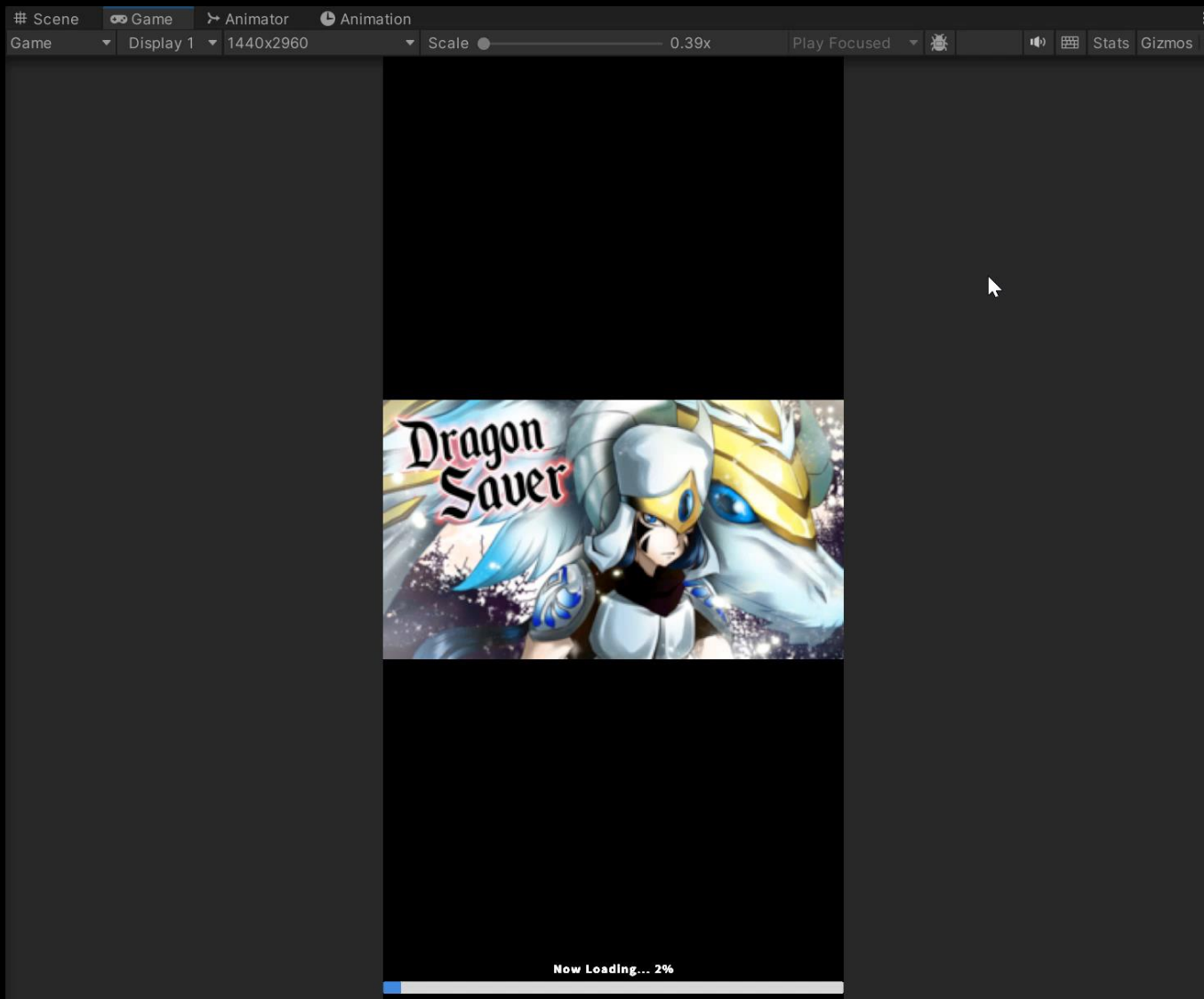
- Runtime Only, GameObject.SetActive, Meteorite:
- Runtime Only, GameObject.SetActive, EnemySp:
- Runtime Only, GameObject.SetActive, Player:
- Runtime Only, GameController.OnGameOver, GameCon:

At the bottom of the Inspector panel, the **Daily Rank** property is set to **GameController (Daily Rank Regi)**, which is also highlighted with a red dashed box. Below this, the **Game UI Controller (Script)** and **Daily Rank Register (Script)** components are visible.



# 랭킹 데이터 등록

## ■ 결과 화면





# 랭킹 데이터 등록

## ■ 결과 화면 (Backend Console)

Backnd Console

ProjectA

랭킹 관리

랭킹명: DAILY\_RANK

초기화 기간: 1일

랭킹 보상: UUID

UUID: e795f580-04f4-11ee-958d-ad4ae0c79832

항목: USER\_DATA > dailyBestScore 추가 항목: --

기간: 2023.06.07 - 2023.06.07

UUID: e795f580-04f4-11ee-958d-ad4ae0c79832

회원번호 또는 닉네임을 입력해

<input type="checkbox"/>	순위	회원번호	닉네임	스코어	추가 항목
<input type="checkbox"/>	1	7814bb60-04f4-11ee-bb76-8582e94b5d40	--	900	--

< 1 >

10개씩 보기

확인



# 랭킹 데이터 등록

## ■ 결과 화면 (Backend Console)

The image shows a Backend Console interface. On the left, a 'GAME OVER' screen displays 'SCORE 200' and a '로비로 이동' (Move to Lobby) button. A red box highlights the 'SCORE 200' text, with a red arrow pointing to a table in a modal window. The modal window, titled 'DAILY\_RANK', shows details for a specific ranking entry. The table below is highlighted with a red box:

<input type="checkbox"/>	순위	회원번호	닉네임	스코어	추가 항목
<input type="checkbox"/>	1	7814bb60-04f4-11ee-bb76-8582e94b5d40		200	

Below the table, there are navigation arrows, a page number '1', and a '10개씩 보기' dropdown. A '확인' (Confirm) button is at the bottom right of the modal.

현재는 게임오버 될 때마다 아무 조건 없이 랭킹 데이터를 갱신하기 때문에 900 -> 200으로 더 낮은 점수를 획득해도 랭킹 데이터가 갱신된다.



# 랭킹 데이터 등록

## ■ 랭킹 데이터 확인 및 갱신

- 내 랭킹 데이터를 불러와 점수를 비교하고, 최고 점수 일 때만 랭킹 갱신
  - DailyRankRegister Script 수정

```
1  using BackEnd;
2  using UnityEngine;
3
4  public class DailyRankRegister : MonoBehaviour
5  {
6      public void Process(int newScore)
7      {
8          //UpdateMyRankData(newScore);
9          UpdateMyBestRankData(newScore);
10     }
11
12     private void UpdateMyRankData(int newScore) ...
13
55
```



# 랭킹 데이터 등록

## □ DailyRankRegister Script 수정 (계속)

```
56 private void UpdateMyBestRankData(int newScore)
57 {
58     Backend.URank.User.GetMyRank(Constants.DAILY_RANK_UUID, callback =>
59     {
60         if ( callback.IsSuccess() )
61         {
62             // JSON 데이터 파싱 성공
63             try{...}
64         }
65         else
66         {
67             // 자신의 랭킹 정보가 존재하지 않을 때는 현재 점수를 새로운 랭킹으로 등록
68             if ( callback.GetMessage().Contains("userRank") )
69             {
70                 UpdateMyRankData(newScore);
71                 Debug.Log($"새로운 랭킹 데이터 생성 및 등록 : {callback}");
72             }
73         }
74     });
75 }
```

뒷장

== backend method ==

Backend.URank.User.GetMyRank(string rankUuid, int gap=0);

rankUuid 랭킹 테이블에 등록되어 있는 내 랭킹 조회, gap 값을 설정하면 내 랭킹 위/아래 인접 유저 검색 가능  
gap이 3일 때 내 랭킹이 10위이면 10-3 ~ 10+3까지 7~13위의 유저 랭킹이 조회된다.



# 랭킹 데이터 등록

## □ DailyRankRegister Script 수정 (계속)

```
62 // JSON 데이터 파싱 성공
63 try
64 {
65     LitJson.JsonData rankDataJson = callback.FlattenRows();
66
67     // 받은 데이터의 개수가 0이면 데이터가 없는 것
68     if ( rankDataJson.Count <= 0 )
69     {
70         Debug.LogWarning("데이터가 존재하지 않습니다.");
71     }
72     else
73     {
74         // 랭킹을 등록할 때는 컬럼명을 "dailyBestScore"로 저장했지만
75         // 랭킹을 불러올 때는 컬럼명이 "score"로 통일되어 있다.
76
77         // 추가로 등록한 항목은 컬럼명을 그대로 사용
78         int bestScore = int.Parse(rankDataJson[0]["score"].ToString());
79
80         // 현재 점수가 최고 점수보다 높으면
81         if ( newScore > bestScore )
82         {
83             // 현재 점수를 새로운 최고 점수로 설정하고, 랭킹에 등록
84             UpdateMyRankData(newScore);
85
86             Debug.Log($"최고 점수 갱신 {bestScore} -> {newScore}");
87         }
88     }
89 }
90 // JSON 데이터 파싱 실패
91 catch ( System.Exception e )
92 {
93     // try-catch 에러 출력
94     Debug.LogError(e);
95 }
```



# 랭킹 데이터 등록

## ■ 결과 화면

Console

Clear Collapse Error Pause Editor

message : Success

[15:14:28] 2023-06-07T05:30:53.541Z의 게임 정보 데이터 수정을 요청합니다.  
UnityEngine.Debug:Log (object)

[15:14:28] 게임 정보 데이터 수정에 성공했습니다. : statusCode : 204  
message : Success

[15:14:28] 최고 점수 갱신 200 -> 400  
UnityEngine.Debug:Log (object)

[15:14:28] 데이터 조회에 성공했습니다 : statusCode : 200  
message : Success

[15:14:28] 랭킹 등록에 성공했습니다 : statusCode : 204  
message : Success

랭킹에 등록된 최고 점수보다 더 높은 점수를 획득했을 때만 랭킹 데이터를 갱신한다.