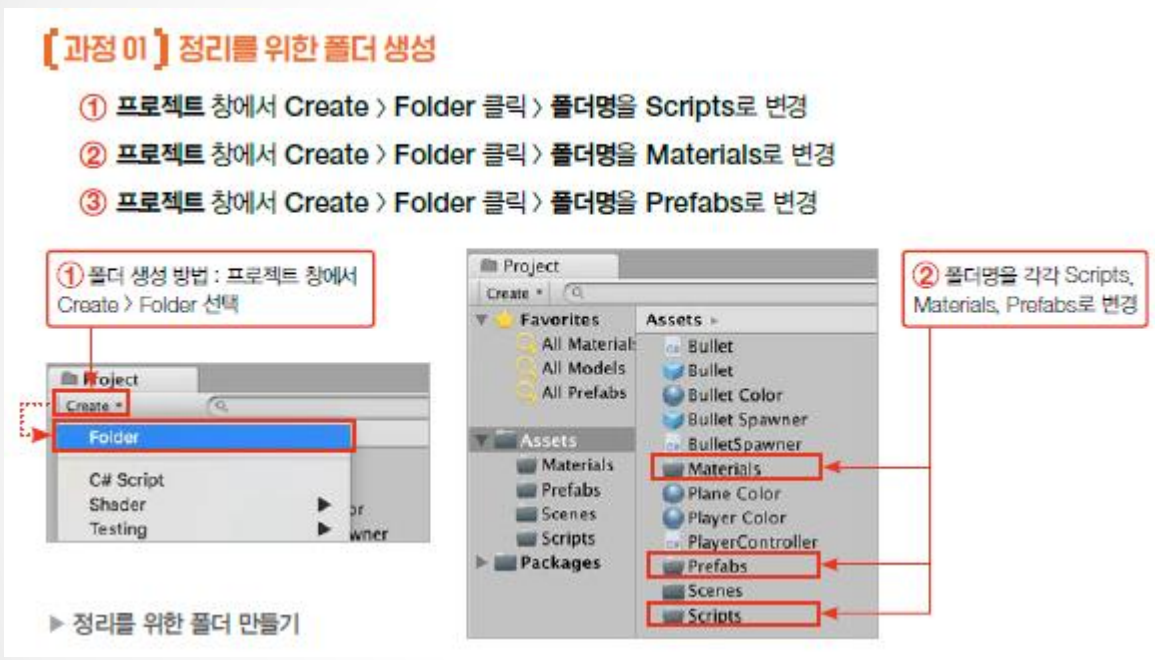
유니티 프로그래밍 강의

이준

닷지 게임 매니저와 UI, 최종 완성

8.1 프로젝트 정리

- 게임을 완성하기 전에 프로젝트 창에 폴더를 만들어 에셋과 스크립트를 정돈하겠음.



8.2 바닥 회전

◦ 8.2.1 Rotator 스크립트 준비

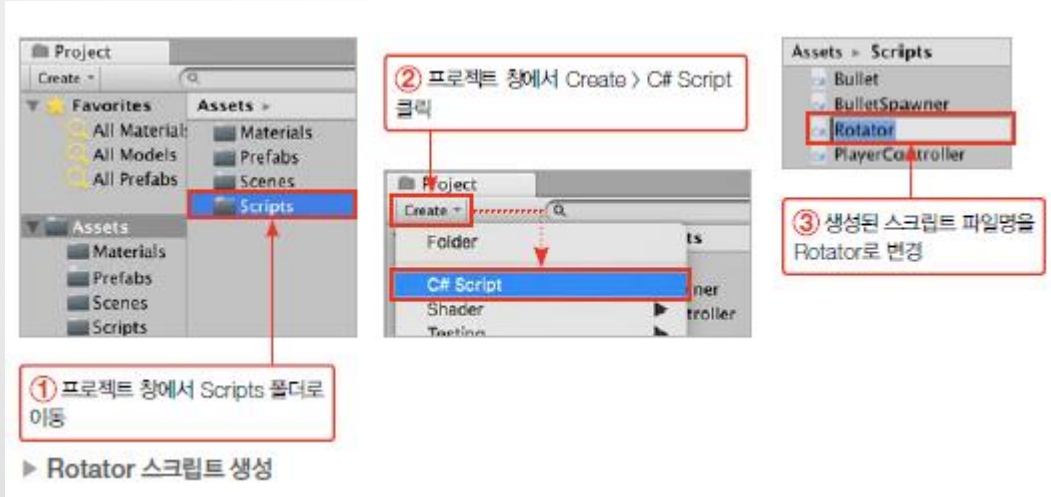
- 게임 오브젝트를 일정 속도로 회전시키는 Rotator 스크립트를 작성함.

[과정 01] Rotator 스크립트 생성

① **프로젝트** 창에서 **Scripts** 폴더로 이동

② **프로젝트** 창에서 **Create** > **C# Script** 클릭

③ 생성된 스크립트 파일명을 **Rotator**로 변경



▶ Rotator 스크립트 생성

[과정 02] Rotator 스크립트에 변수 선언

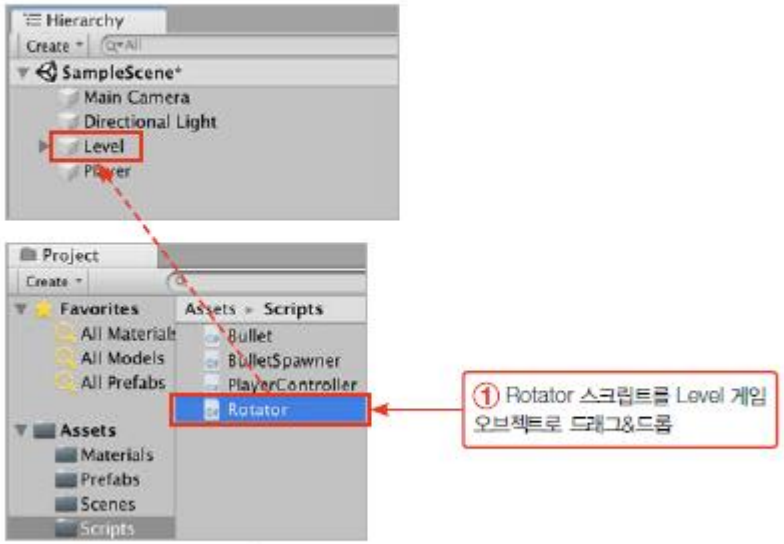
① Rotator 스크립트를 열고 다음과 같이 수정

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Rotator : MonoBehaviour {
    public float rotationSpeed = 60f;

    void Update() {
        transform.Rotate(0f, rotationSpeed, 0f);
    }
}
```

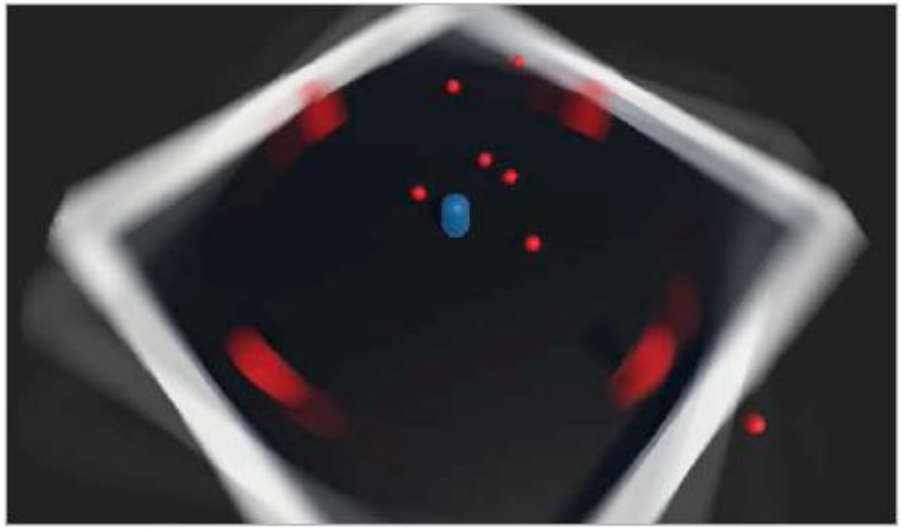
8.2 바닥 회전

[과정 03] Rotator 스크립트 적용

① Rotator 스크립트를 **하이어라키** 창의 **Level** 게임 오브젝트로 **드래그&드롭**



▶ Rotator 스크립트 적용



▶ 비정상적으로 빠른 회전

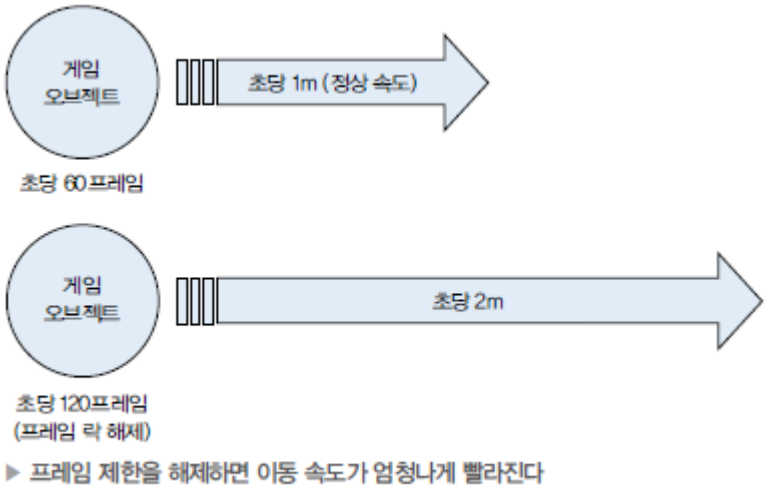
8.2 바닥 회전

- 8.2.2 속도와 시간 간격

  - Update( ) 메서드에서 초당 이동 속도나 회전 속도 등 시간과 관련된 수치를 다룰 때는 시간 간격을 고려해야 함.

```
void Update() {
    // 한 번에 1/60미터 이동
}
```

프레임 제한 해제



▶ 프레임 제한을 해제하면 이동 속도가 엄청나게 빨라진다

초당 프레임이 서로 다른 컴퓨터

- **A 컴퓨터** : 성능이 낮아 **1초에 60번** 게임 화면 갱신
- **B 컴퓨터** : 성능이 좋아 **1초에 120번** 게임 화면 갱신

그리고 각 컴퓨터에서 **초당 60도 회전을 의도하고** 아래 코드를 동작시켰다고 가정합니다.

```
void Update() {
    transform.Rotate(0f, 60f, 0f);
}
```

Update( )가 1초에 60번 실행되는 환경을 기준으로 설명하면

1. 60도를 실행횟수인 60조각으로 나누기(1/60 곱하기) → 1조각은 1도
2. Update( )가 1초 동안 60번 실행 → 한 번에 1조각(1도)씩 60번 누적
3. 1초 동안 누적된 회전은 60도

8.2 바닥 회전

- 8.2.3 Rotator 스크립트 수정
  - Rotator 스크립트를 다시 열고 Update( ) 메서드를 수정함.

[과정 01] Rotator 스크립트 수정

① Rotator 스크립트의 Update( ) 메서드를 다음과 같이 수정

```
void Update() {
    transform.Rotate(0f, rotationSpeed * Time.deltaTime, 0f);
}
```

8.3 게임 UI 제작

- 생존 시간, 게임오버, 최고 기록을 표현하는 UI를 만듬.
- 유니티의 UI 시스템1은 UI 요소를 게임 월드 속의 게임 오브젝트로 취급함.

▶ 유니티에서는 UI도 게임 오브젝트

8.3 게임 UI 제작

◦ 8.3.1 생존 시간 텍스트 제작

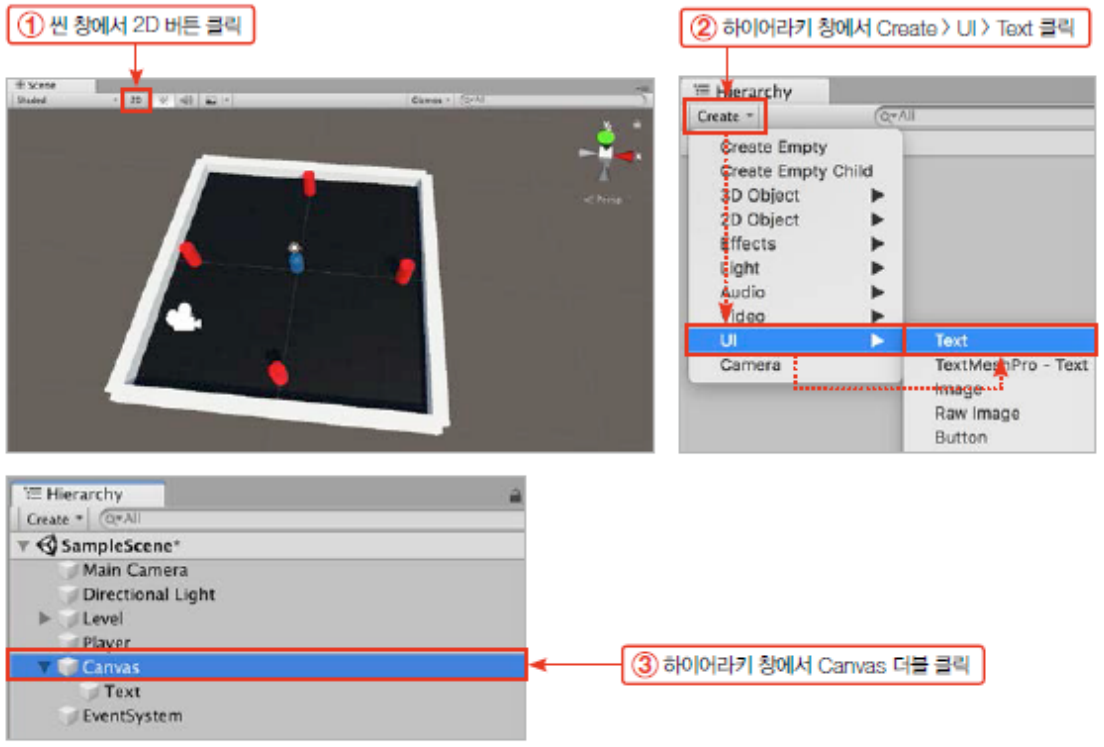
- 씬 편집 모드를 2D 모드로 바꿔 UI를 편집하기 쉽게 하고, 시간을 표시하는 UI 텍스트게임 오브젝트를 만들어 봄.

[과정 01] UI 텍스트 만들기

① **씬** 창에서 **2D** 버튼 클릭 → 2D 편집 모드로 전환됨

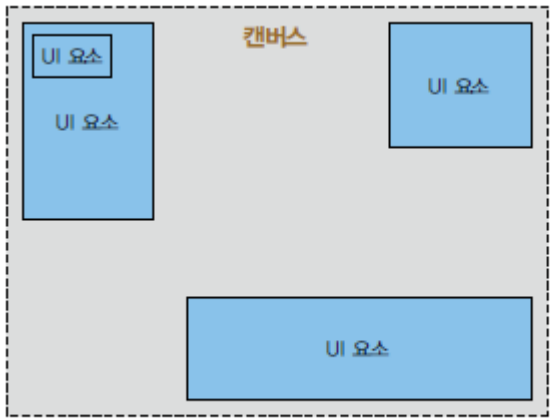
② **하이어라키** 창에서 **Create** > **UI** > **Text** 클릭

③ **하이어라키** 창에서 **Canvas** 더블 클릭 → 씬 창에서 Canvas가 포커스됨



▶ UI 텍스트 만들기

캔버스와 이벤트 시스템

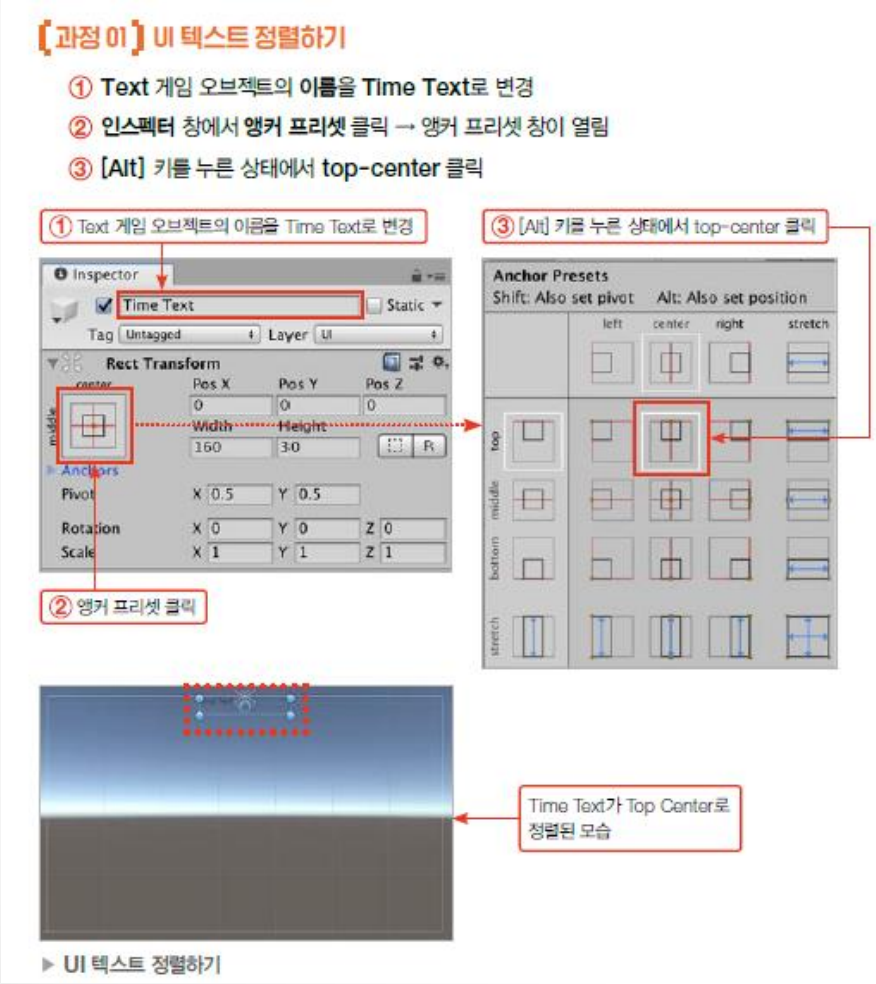


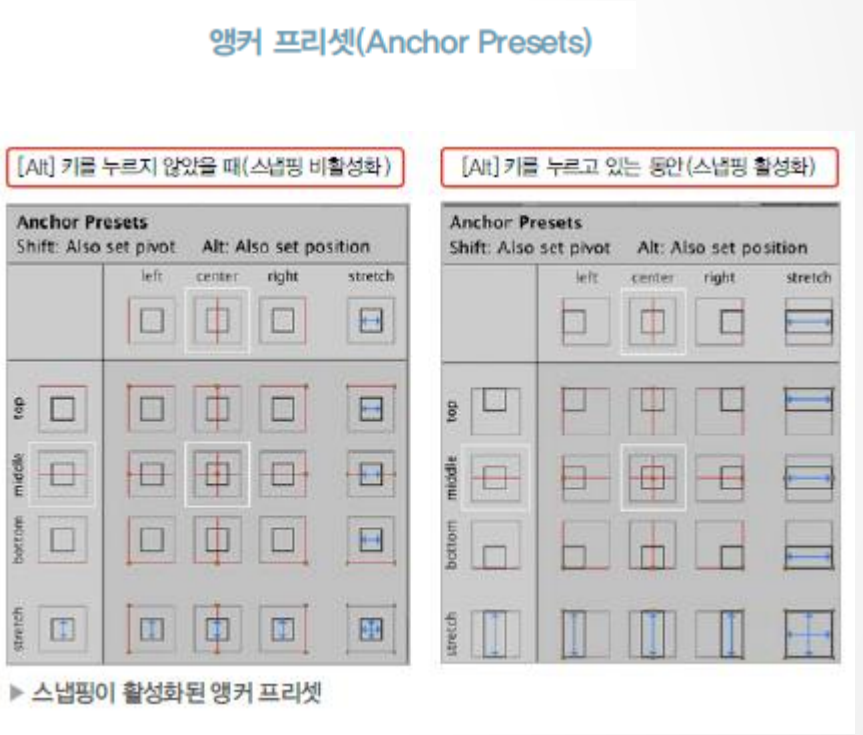
▶ 캔버스는 UI들의 액자

8.3 게임 UI 제작

- 8.3.2 텍스트 배치

  - Text 게임 오브젝트의 이름을 변경하고 캔버스 중앙 상단으로 정렬하여 배치하겠음.





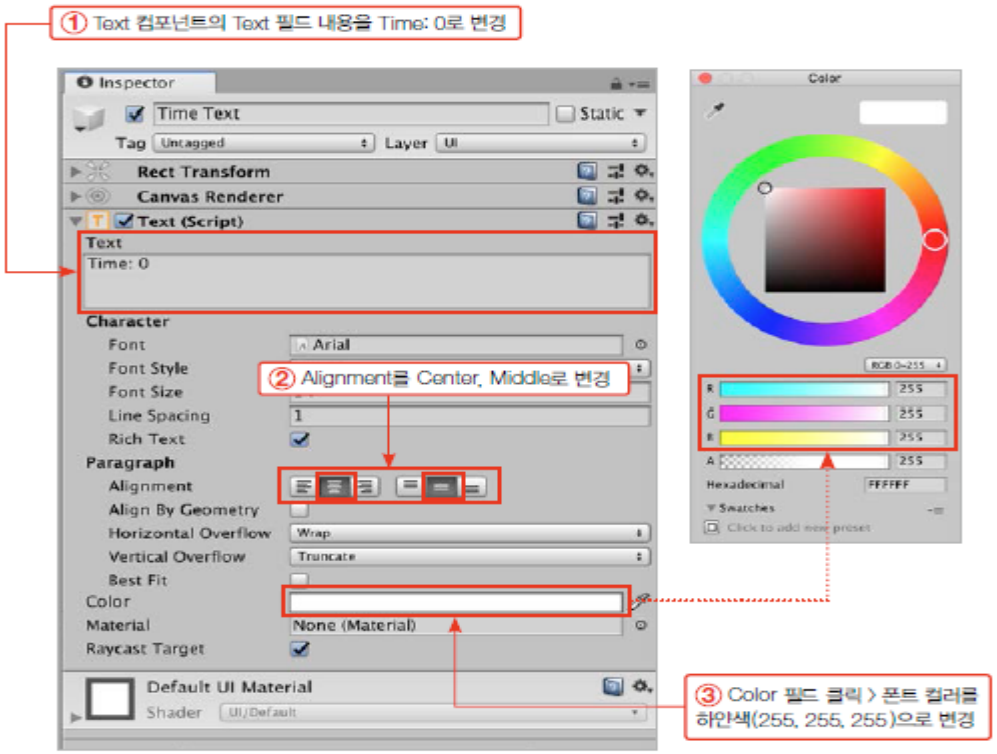
8.3 게임 UI 제작

◦ 8.3.3 텍스트 꾸미기

- UI 텍스트를 배치했으니 텍스트 내용과 폰트 색깔을 변경함.

[과정 01] Time Text의 텍스트 변경하기

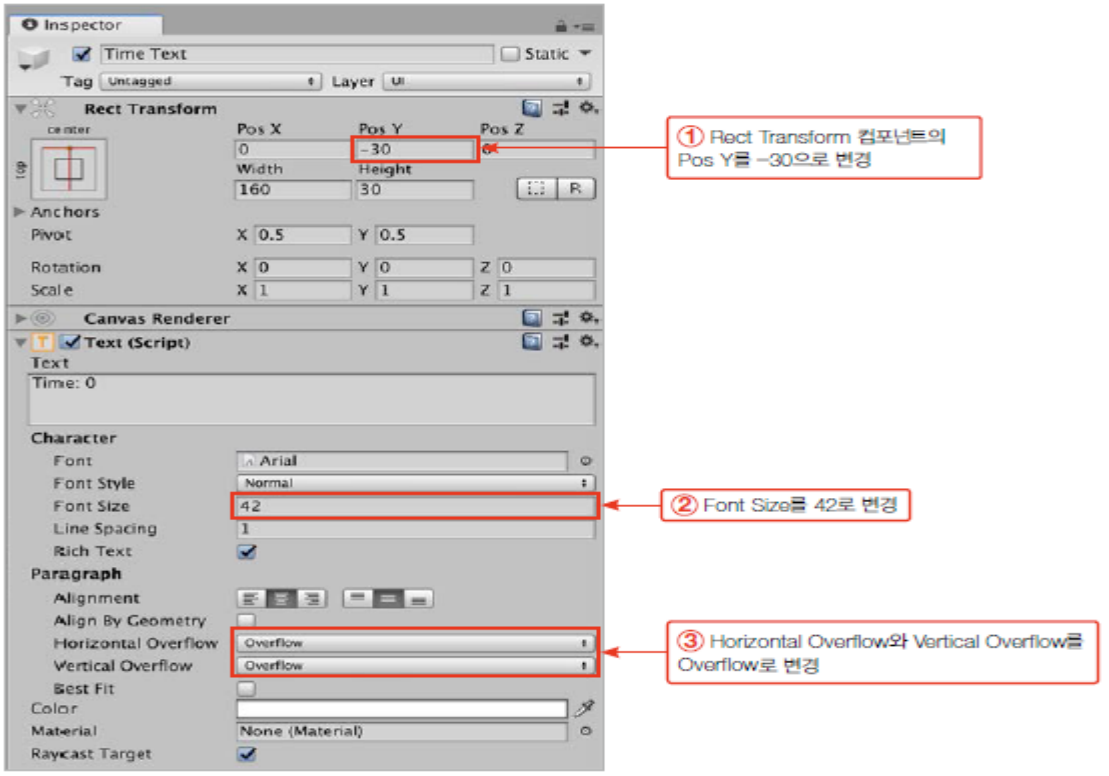
① Time Text 게임 오브젝트의 Text 컴포넌트의 Text 필드 내용을 Time: 0으로 변경
② Text 컴포넌트의 Alignment를 Center, Middle로 변경
③ Color 필드 클릭 > 폰트 컬러를 하얀색(255, 255, 255)으로 변경



▶ Time Text의 텍스트 변경하기

[과정 02] Time Text의 폰트 크기 키우기

① Rect Transform 컴포넌트의 Pos Y를 −30으로 변경
② Text 컴포넌트의 Font Size를 42로 변경
③ Horizontal Overflow와 Vertical Overflow를 Overflow로 변경



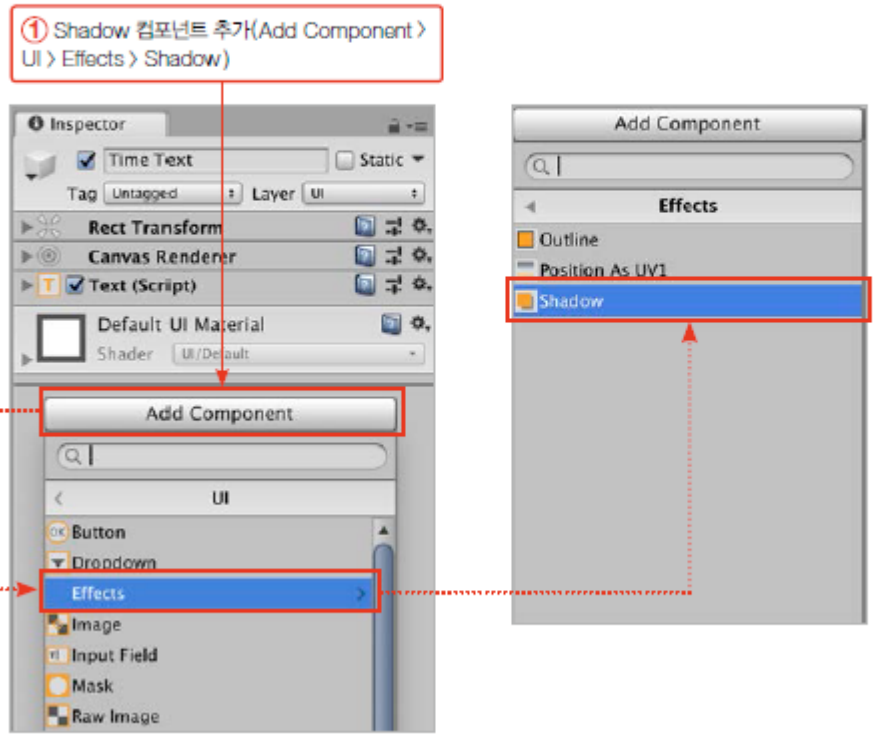
▶ Time Text의 폰트 크기 늘리기

8.3 게임 UI 제작

- 텍스트가 잘 보이도록 그림자 효과를 추가함.

[과정 01] 그림자 효과 추가

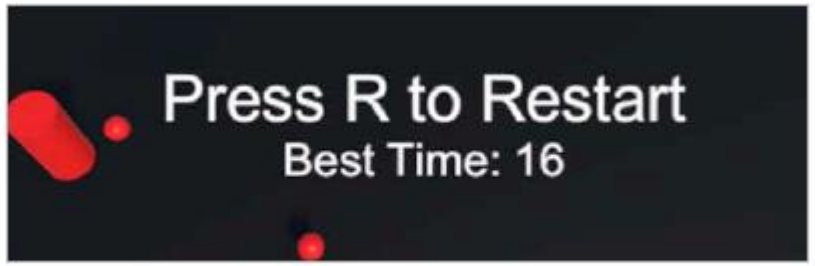
① Time Text 게임 오브젝트에 Shadow 컴포넌트 추가(Add Component > UI > Effects > Shadow)



▶ 그림자 효과 추가

8.3 게임 UI 제작

- 8.3.4 게임오버 텍스트와 최고 기록 텍스트
  - 게임오버되었을 때 활성화할 UI 텍스트를 만듦.



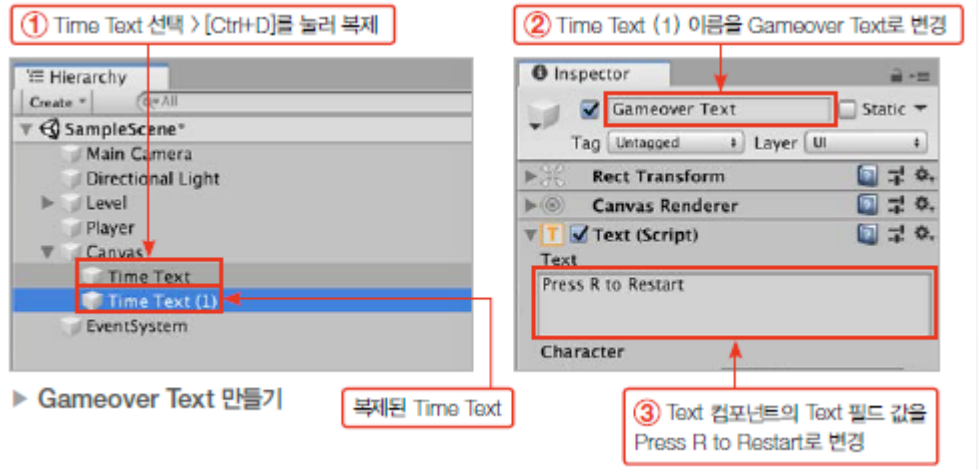
▶ 게임 재시작 안내 텍스트와 최고 기록 텍스트

[과정 01] Gameover Text 만들기

① **하이어라키** 창에서 **Time Text** 선택 > **[Ctrl+D]**를 눌러 복제

② **Time Text (1)** 게임 오브젝트 이름을 **Gameover Text**로 변경

③ **Gameover Text** 게임 오브젝트의 **Text** 컴포넌트의 **Text** 필드 값을 **Press R to Restart**로 변경



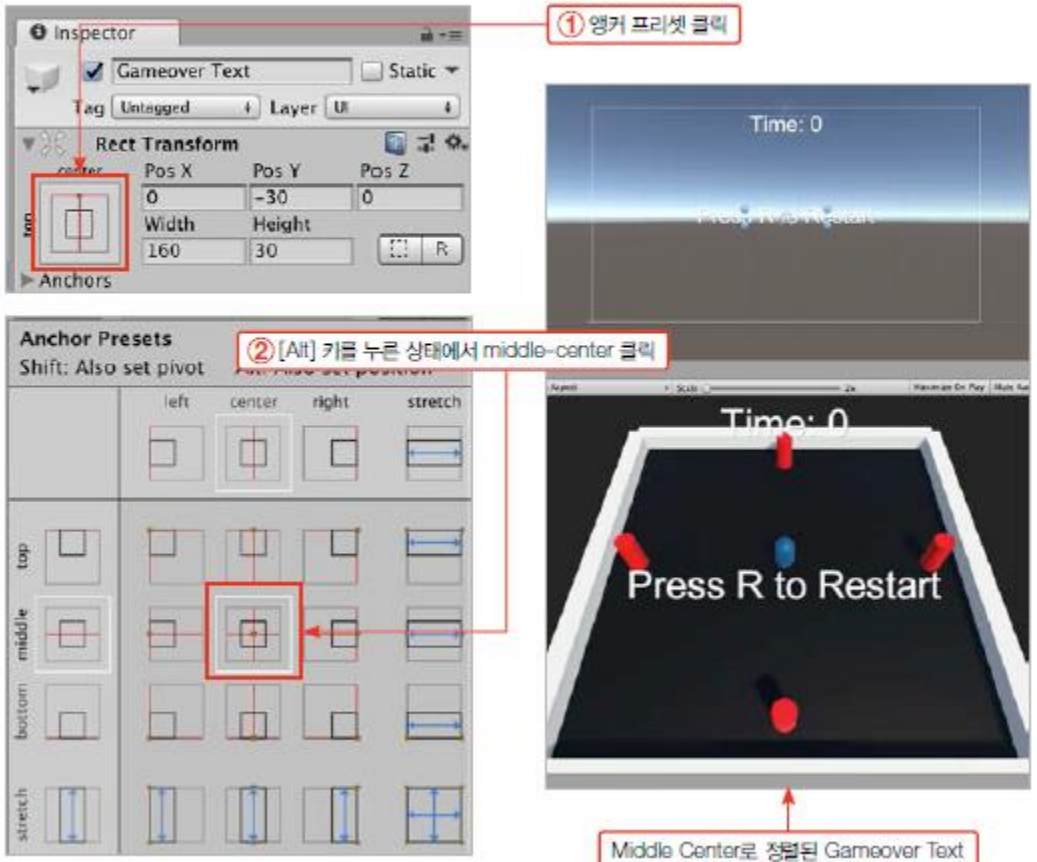
▶ Gameover Text 만들기

8.3 게임 UI 제작

[과정 02] Gameover Text를 화면 정중앙에 배치하기

① **앵커 프리셋** 클릭 → 앵커 프리셋 창이 나타남

② [Alt] 키를 누른 상태에서 **middle-center** 클릭



▶ Gameover Text를 화면 정중앙에 배치하기

[과정 03] Record Text 만들기

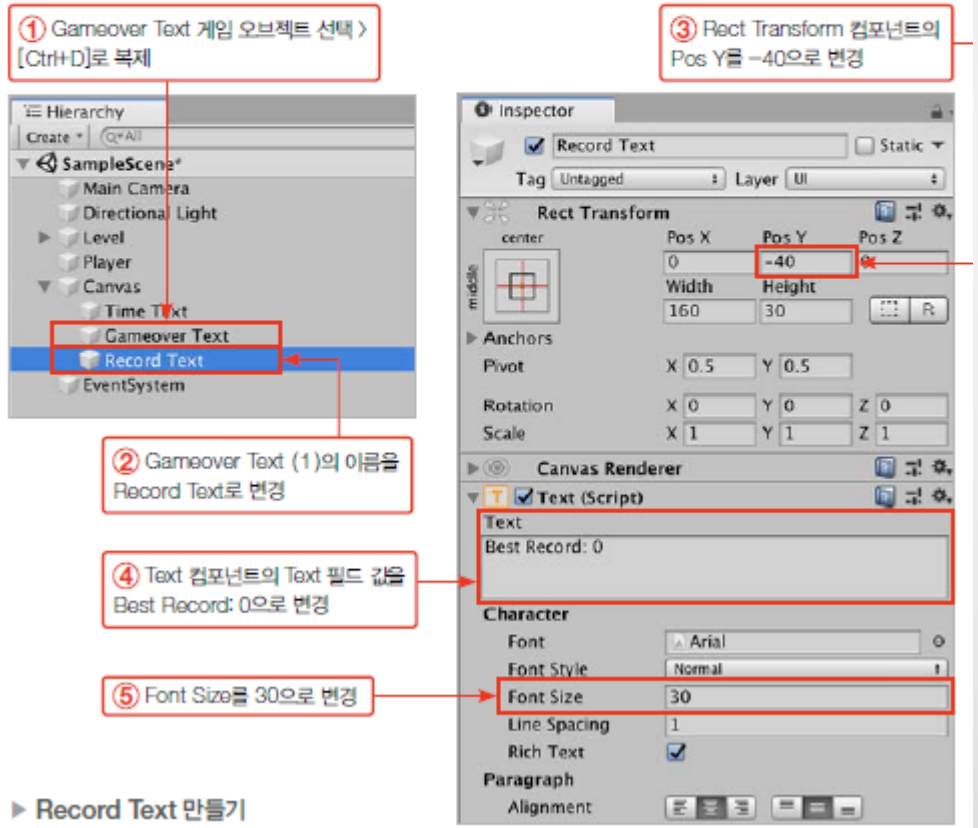
① **하이어라키** 창에서 **Gameover Text** 게임 오브젝트 선택 › **[Ctrl+D]**로 복제

② **Gameover Text (1)** 게임 오브젝트의 이름을 **Record Text**로 변경

③ **Rect Transform** 컴포넌트의 **Pos Y**를 **−40**으로 변경

④ **Text** 컴포넌트의 **Text** 필드 값을 **Best Record: 0**으로 변경

⑤ **Text** 컴포넌트의 **Font Size**를 **30**으로 변경

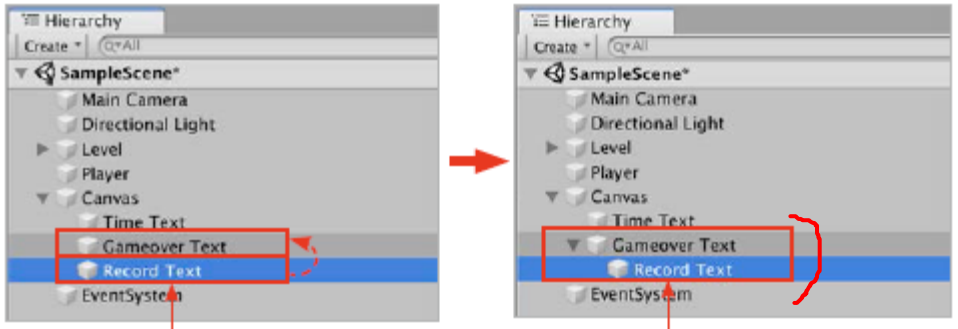


▶ Record Text 만들기

8.3 게임 UI 제작

[과정 04] Record Text를 Gameover Text의 자식으로 만들기

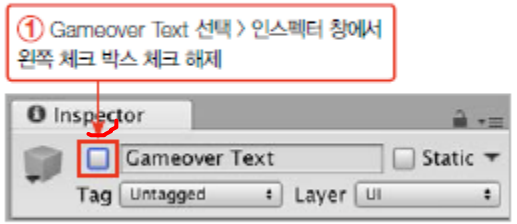
① **하이어라키** 창에서 **Record Text**를 **Gameover Text**로 **드래그&드롭**



▶ Record Text를 Gameover Text의 자식으로 만들기

[과정 05] Gameover Text 비활성화

① **하이어라키** 창에서 **Gameover Text** 선택 › **인스펙터** 창에서 왼쪽 체크 박스 **체크 해제**



▶ Gameover Text 비활성화

8.4 게임 매니저 제작

◦ 8.4.1 GameManager 스크립트 준비

- 게임 매니저로서 동작할 GameManager 스크립트를 준비함.

[과정 01] GameManager 스크립트 만들기

① **프로젝트** 창의 Scripts 폴더에서 Create > C# Script 클릭

② 생성된 스크립트의 이름을 GameManager로 변경하고, 스크립트 열기

[과정 02] 필요한 라이브러리 가져오기

① GameManager 스크립트를 열고 전체 코드를 다음과 같이 수정

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI; // UI 관련 라이브러리
using UnityEngine.SceneManagement; // 씬 관리 관련 라이브러리

public class GameManager : MonoBehaviour {
    void Start() {

    }

    void Update() {

    }
}
```

[과정 03] GameManager에 필요한 변수와 메서드 선언하기

① GameManager 스크립트를 다음과 같이 수정

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI; // UI 관련 라이브러리
using UnityEngine.SceneManagement; // 씬 관리 관련 라이브러리

public class GameManager : MonoBehaviour {
    public GameObject gameoverText; // 게임오버 시 활성화할 텍스트 게임 오브젝트
    public Text timeText; // 생존 시간을 표시할 텍스트 컴포넌트
    public Text recordText; // 최고 기록을 표시할 텍스트 컴포넌트

    private float surviveTime; // 생존 시간
    private bool isGameover; // 게임오버 상태

    void Start() {
        // 생존 시간과 게임오버 상태 초기화
        surviveTime = 0;
        isGameover = false;
    }
    void Update() {

    }

    // 현재 게임을 게임오버 상태로 변경하는 메서드
    public void EndGame() {

    }
}
```

8.4 게임 매니저 제작

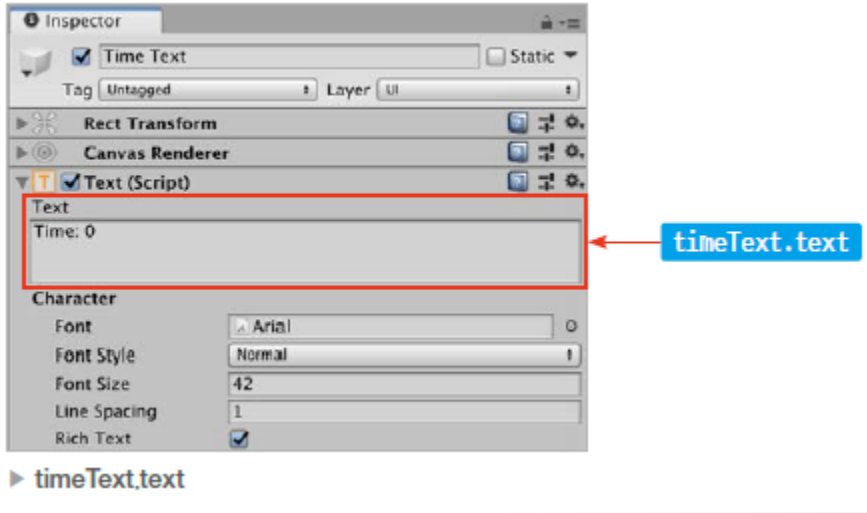
◦ 8.4.2 생존 시간 표시하기

- GameManager 스크립트에 생존 시간을 측정하고 표시하는 기능을 추가함.

[과정 01] 생존 시간 표시하기

① GameManager 스크립트의 Update() 메서드를 다음과 같이 수정

```
void Update() {
    // 게임오버가 아닌 동안
    if (!isGameover)
    {
        // 생존 시간 갱신
        surviveTime += Time.deltaTime;
        // 갱신한 생존 시간을 timeText 텍스트 컴포넌트를 이용해 표시
        timeText.text = "Time: " + (int) surviveTime;
    }
}
```



8.4 게임 매니저 제작

- 8.4.3 게임 재시작 구현

  - 게임오버 상태에서는 특정 키를 눌러 게임을 재시작할 수 있어야 함.

[과정 01] SampleScene을 실시간으로 로드하기

① GameManager 스크립트의 Update() 메서드를 다음과 같이 수정

```
void Update() {
    // 게임오버가 아닌 동안
    if (!isGameover)
    {
        // 생존 시간 갱신
        surviveTime += Time.deltaTime;
        // 갱신한 생존 시간을 timeText 텍스트 컴포넌트를 이용해 표시
        timeText.text = "Time: " + (int) surviveTime;
    }
    else
    {
        // 게임오버 상태에서 R 키를 누른 경우
        if (Input.GetKeyDown(KeyCode.R))
        {
            // SampleScene 씬을 로드
            SceneManager.LoadScene("SampleScene");
        }
    }
}
```

8.4 게임 매니저 제작

- 8.4.4 EndGame( ) 구현

  - GameManager 스크립트에 현재 게임을 게임오버 상태로 만드는 EndGame( ) 메서드를 구현함.

EndGame ( ) 메서드는 다음 기능을 가져야 합니다.

- 게임오버 상태 isGameover를 true로 변경
- 현재 생존 시간 기록과 최고 생존 시간 기록 비교
- 게임오버 UI를 활성화하고 최고 기록 표시

[과정 01] EndGame ( ) 메서드 작성하기

① GameManager 스크립트의 EndGame ( ) 메서드를 다음과 같이 수정

```
public void EndGame() {
    // 현재 상태를 게임오버 상태로 전환
    isGameover = true;
    // 게임오버 텍스트 게임 오브젝트를 활성화
    gameoverText.SetActive(true);
}
```

8.4 게임 매니저 제작

- 8.4.5 PlayerPrefs

  - PlayerPrefs는 Player Preference (플레이어 설정)라고 읽으며, 간단한 방식으로 어떤 수치를 로컬(프로그램을 실행 중인 현재 컴퓨터)에 저장하고 나중에 다시 불러오는 메서드를 제공하는 유니티에 내장된 클래스임.

PlayerPrefs를 사용하는 방법

[PlayerPrefs를 사용해 저장한 키-값 데이터는 로컬에 파일로 저장되어 있음]

키	값

```
PlayerPrefs.SetFloat("Gold",30f);
PlayerPrefs.SetFloat("Score",50f);
```

키	값
Gold	30
Score	50

```
PlayerPrefs.SetFloat("Gold",100f);
```

키	값
Gold	100
Score	50

PlayerPrefs 사용시 유의점

PlayerPrefs는 float 외에도 int와 string을 저장하고 가져올 수 있습니다.

int 저장/가져오기

- PlayerPrefs.SetInt(string key, int value);
- PlayerPrefs.GetInt(string key);

string 저장/가져오기

- PlayerPrefs.SetString(string key, string value);
- PlayerPrefs.GetString(string key);

8.4 게임 매니저 제작

- 8.4.6 최고 기록 저장/읽기 구현

  - GameManager 스크립트에서 작성한 EndGame( ) 메서드로 돌아가 최고 기록을 저장하고 읽는 부분을 추가함.

[과정 01] GameManager 스크립트의 EndGame ( ) 메서드 완성

① GameManager 스크립트의 EndGame ( ) 메서드를 다음과 같이 완성

```
public void EndGame() {
    // 현재 상태를 게임오버 상태로 전환
    isGameover = true;
    // 게임오버 텍스트 게임 오브젝트 활성화
    gameoverText.SetActive(true);

    // BestTime 키로 저장된 이전까지의 최고 기록 가져오기
    float bestTime = PlayerPrefs.GetFloat("BestTime");

    // 이전까지의 최고 기록보다 현재 생존 시간이 더 크다면
    if (surviveTime > bestTime)
    {
        // 최고 기록 값을 현재 생존 시간 값으로 변경
        bestTime = surviveTime;
        // 변경된 최고 기록을 BestTime 키로 저장
        PlayerPrefs.SetFloat("BestTime", bestTime);
    }

    // 최고 기록을 recordText 텍스트 컴포넌트를 이용해 표시
    recordText.text = "Best Time: " + (int) bestTime;
}
```

- 8.4.7 완성된 GameManager 스크립트

  - 지금까지 완성한 GameManager 스크립트의 전체 모습은 다음과 같음.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI; // UI 관련 라이브러리
using UnityEngine.SceneManagement; // 씬 관리 관련 라이브러리

public class GameManager : MonoBehaviour {
    public GameObject gameoverText; // 게임오버 시 활성화할 텍스트 게임 오브젝트
    public Text timeText; // 생존 시간을 표시할 텍스트 컴포넌트
    public Text recordText; // 최고 기록을 표시할 텍스트 컴포넌트

    private float surviveTime; // 생존 시간
    private bool isGameover; // 게임오버 상태

    void Start() {
        // 생존 시간과 게임오버 상태 초기화
        surviveTime = 0;
        isGameover = false;
    }

    void Update() {
        // 게임오버가 아닌 동안
        if (!isGameover)
        {
            // 생존 시간 갱신
            surviveTime += Time.deltaTime;
            // 갱신한 생존 시간을 timeText 텍스트 컴포넌트를 이용해 표시
            timeText.text = "Time: " + (int) surviveTime;
        }
        else
        {
            // 게임오버인 상태에서 R 키를 누른 경우
            if (Input.GetKeyDown(KeyCode.R))
            {
                // SampleScene 씬을 로드
                SceneManager.LoadScene("SampleScene");
            }
            }
        }

        // 현재 게임을 게임오버 상태로 변경하는 메서드
        public void EndGame() {
            // 현재 상태를 게임오버 상태로 전환
            isGameover = true;
            // 게임오버 텍스트 게임 오브젝트를 활성화
            gameoverText.SetActive(true);

            // BestTime 키로 저장된 이전까지의 최고 기록 가져오기
            float bestTime = PlayerPrefs.GetFloat("BestTime");

            // 이전까지의 최고 기록보다 현재 생존 시간이 더 크다면
            if (surviveTime > bestTime)
            {
                // 최고 기록 값을 현재 생존 시간 값으로 변경
                bestTime = surviveTime;
                // 변경된 최고 기록을 BestTime 키로 저장
                PlayerPrefs.SetFloat("BestTime", bestTime);
            }

            // 최고 기록을 recordText 텍스트 컴포넌트를 이용해 표시
            recordText.text = "Best Time: " + (int) bestTime;
        }
    }
```

8.4 게임 매니저 제작

- 8.4.8 PlayerController에서 EndGame( ) 실행
  - PlayerController 스크립트를 편집해서 EndGame( ) 메서드를 실행해야 함.

[과정 01] PlayerController의 Die( ) 메서드 수정

① PlayerController 스크립트 열기 > Die( ) 메서드를 다음과 같이 수정

```
public void Die() {
    // 자신의 게임 오브젝트를 비활성화
    gameObject.SetActive(false);

    // 씬에 존재하는 GameManager 타입의 오브젝트를 찾아서 가져오기
    GameManager gameManager = FindObjectOfType<GameManager>();
    // 가져온 GameManager 오브젝트의 EndGame() 메서드 실행
    gameManager.EndGame();
}
```

8.4 게임 매니저 제작

◦ 8.4.9 게임 매니저 오브젝트 설정

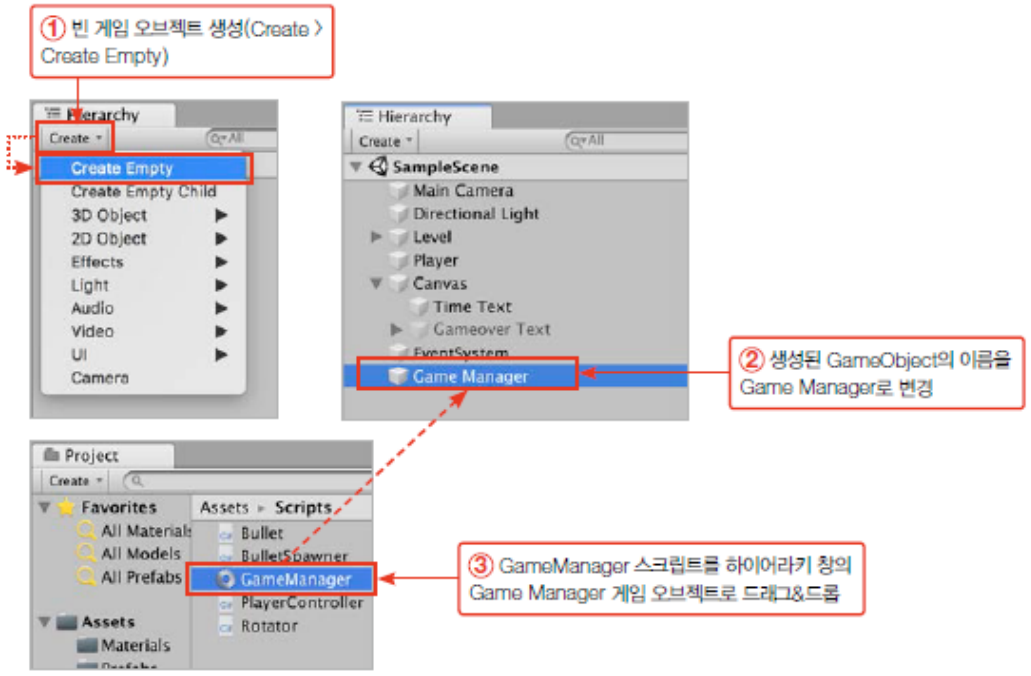
- 게임 매니저 역할을 할 Game Manager 게임 오브젝트를 만들고 GameManager 스크립트를 해당 게임 오브젝트에 추가함.

[과정 01] Game Manager 게임 오브젝트 만들기

① **빈 게임 오브젝트** 생성(**하이어라키** 창에서 **Create** > **Create Empty**)

② 생성된 **GameObject**의 이름을 **Game Manager**로 변경

③ **프로젝트** 창의 **GameManager** 스크립트를 **하이어라키** 창의 **Game Manager** 게임 오브젝트로 **드래그&드롭**



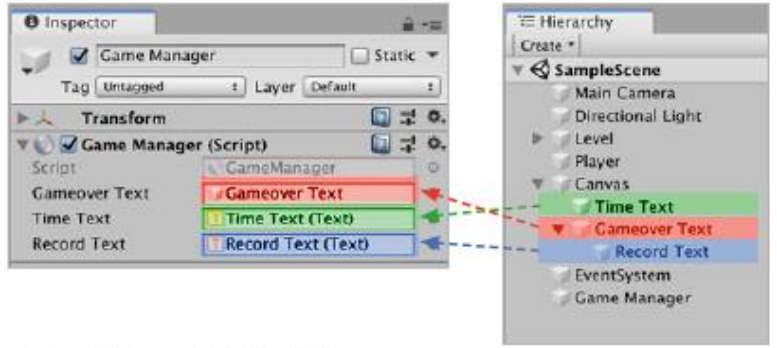
▶ 게임 매니저 만들기

[과정 02] GameManager 컴포넌트 설정

① **하이어라키** 창의 **Gameover Text** 게임 오브젝트를 **GameManager** 컴포넌트의 **Gameover Text** 필드로 **드래그&드롭**

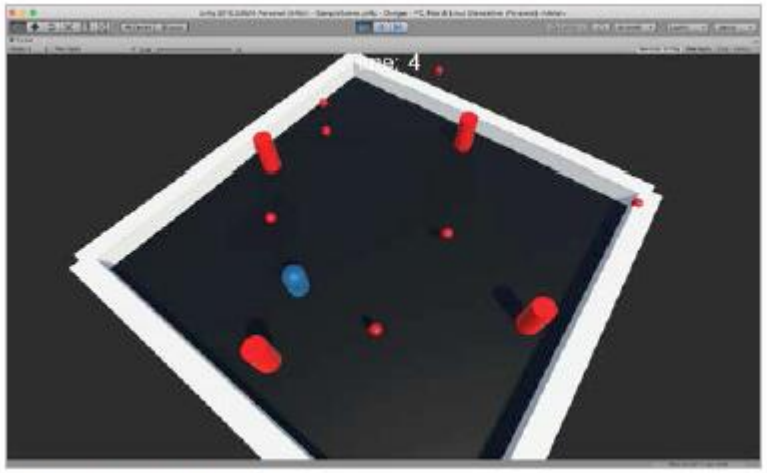
② **하이어라키** 창의 **Time Text** 게임 오브젝트를 **GameManager** 컴포넌트의 **Time Text** 필드로 **드래그&드롭**

③ **하이어라키** 창의 **Record Text** 게임 오브젝트를 **GameManager** 컴포넌트의 **Record Text** 필드로 **드래그&드롭**

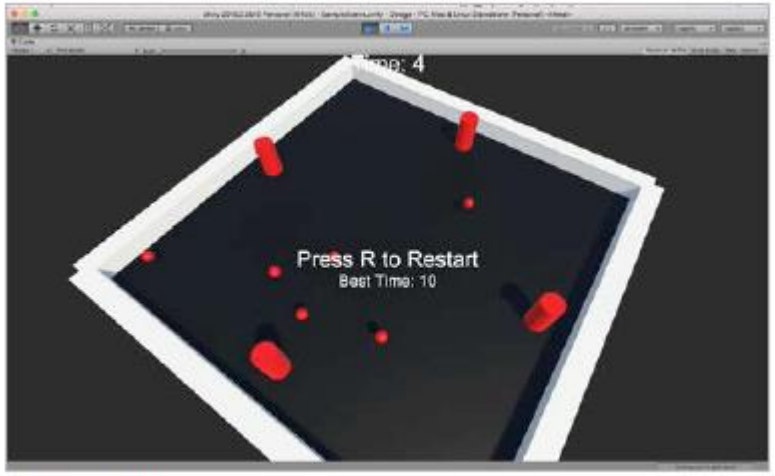


▶ GameManager 컴포넌트 설정

8.4 게임 매니저 제작



▶ 완성된 닷지의 플레이 화면



▶ 게임오버 화면

8.5 빌드하기

- 완성된 유니티 프로젝트 닷지를 다른 사람에게 배포할 수 있는 형태로 빌드함.

[과정 01] 빌드를 저장할 폴더 만들기

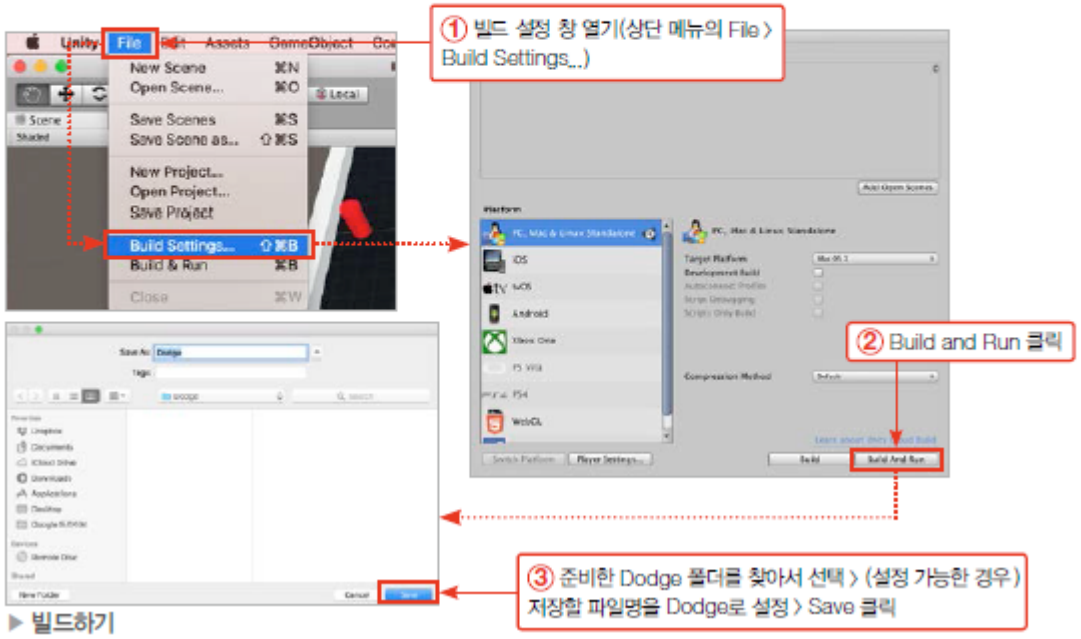
① 바탕 화면이나 내 문서 등의 적당한 위치에 **Dodge**라는 이름의 **새 폴더** 생성

[과정 02] 빌드하기

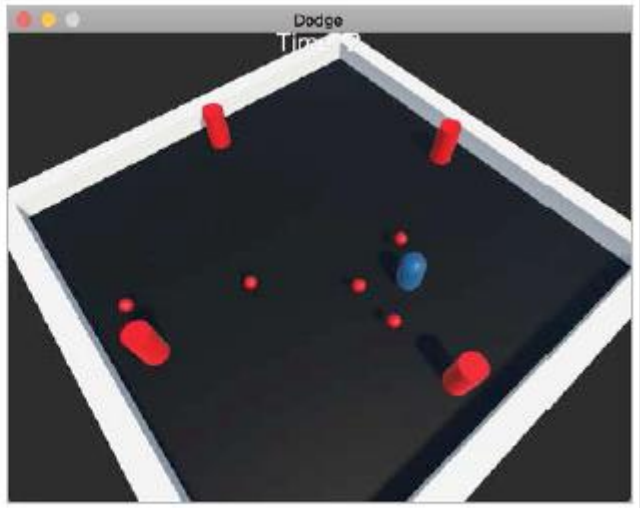
① **빌드 설정** 창 열기(유니티 상단 메뉴의 **File** > **Build Settings...**)

② **빌드 설정** 창에서 **Build and Run** 클릭 → **파일 탐색기**가 실행됨

③ **파일 탐색기**에서 준비한 **Dodge** 폴더를 찾아서 선택 > (설정 가능한 경우) 저장할 파일명을 **Dodge**로 설정 > **Save** 클릭



▶ 빌드하기



▶ 빌드된 게임 플레이하기