



# [5주차] 2D RunGame-2

---

인천정보과학고등학교 3학년 전산과 게임프로그래밍2

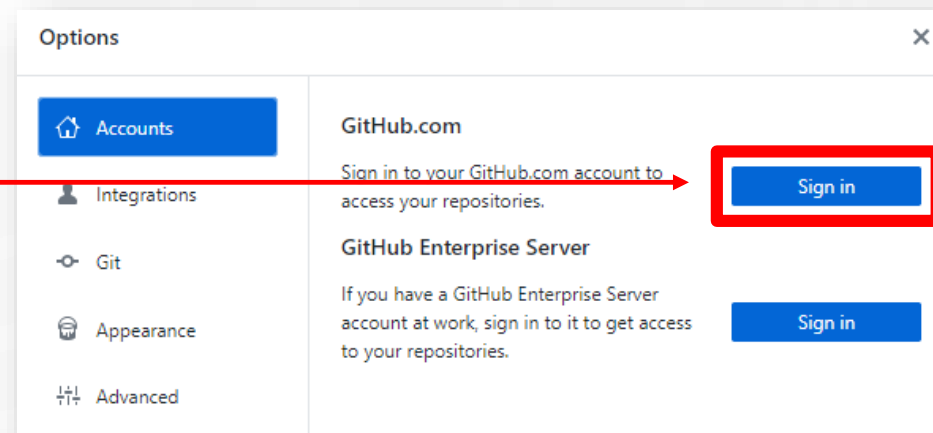
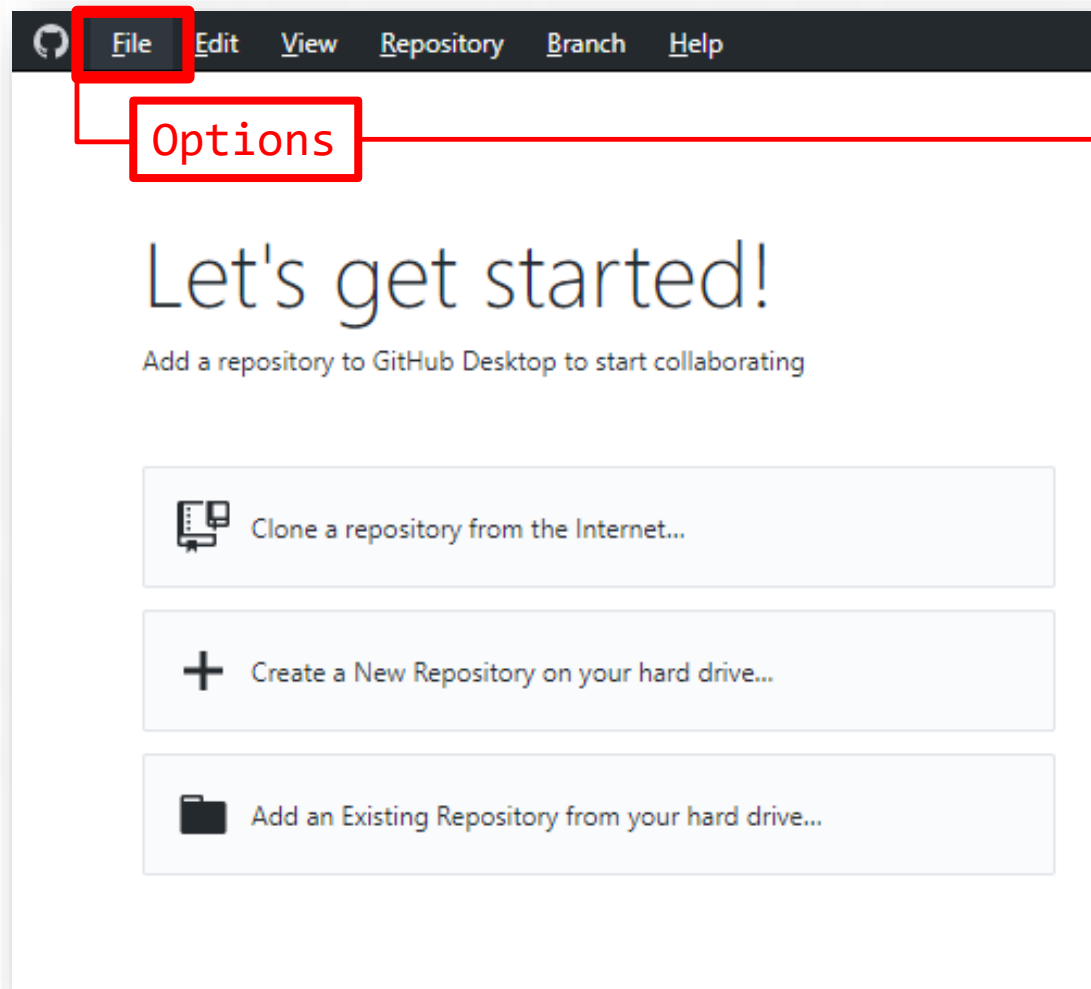
© all right reserved 인천정보과학고 IT소프트웨어과 정수영

# GITHUB CLONE

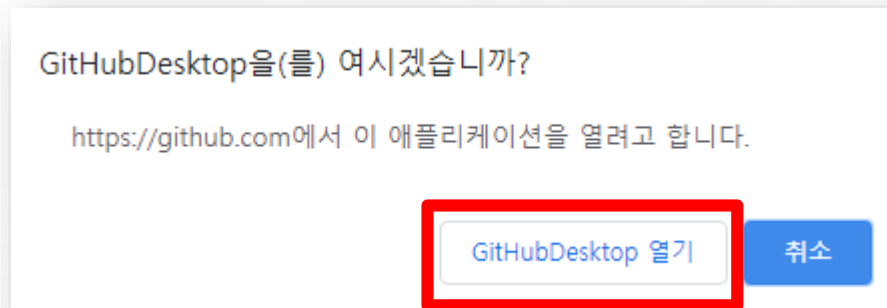
---



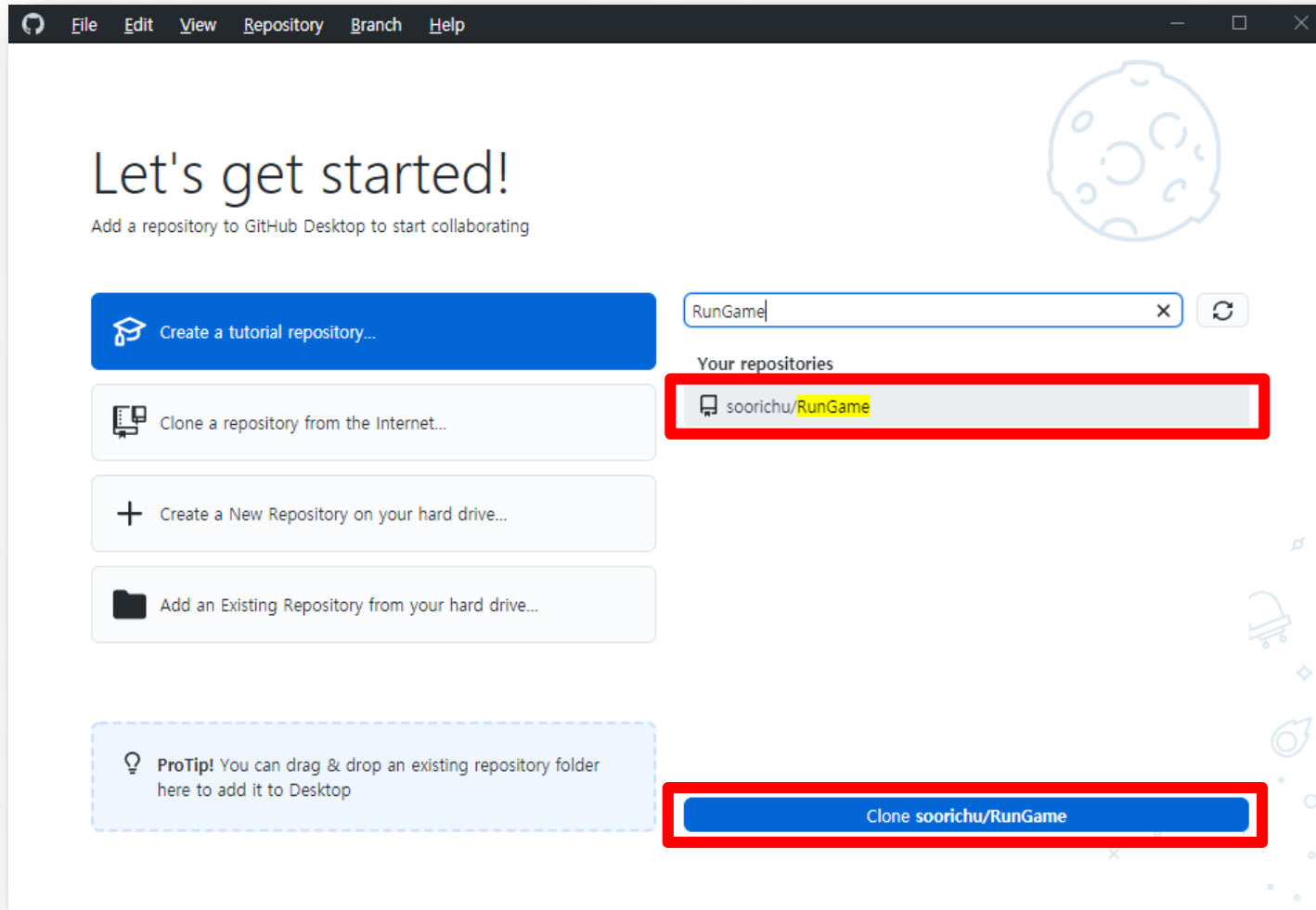
# Git Desktop Login



깃헙 로그인 진행  
크롬에서 다음과 같은 메시지가 나오면



# Clone Repository



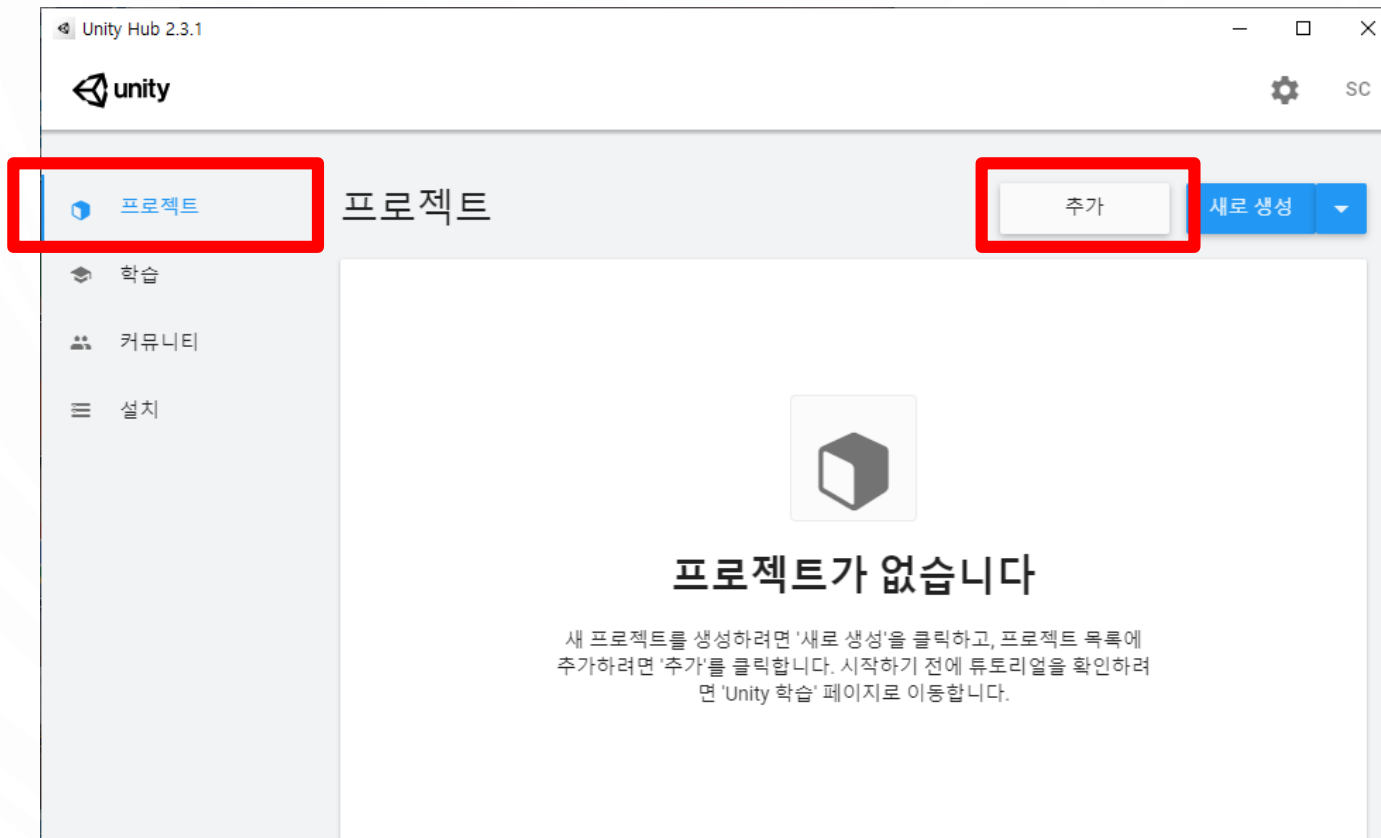
<user>/RunGame 선택

Clone <user>/RunGame 클릭

# Project Open



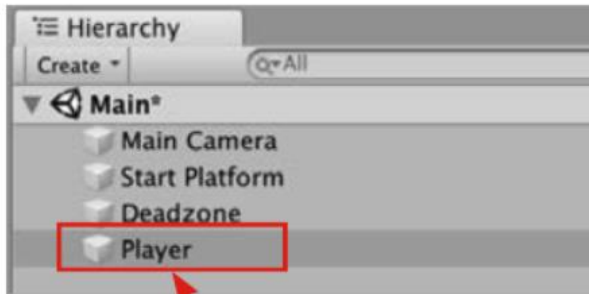
- Unity Hub에서 프로젝트 > 추가 > RunGame 폴더 선택



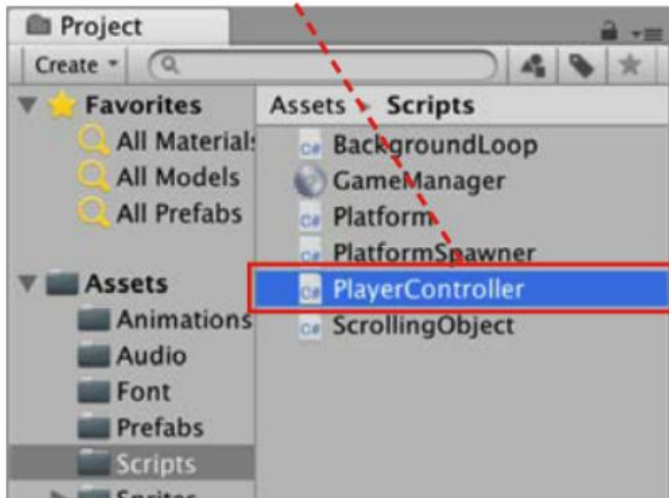
# 플레이어 컨트롤하기

---

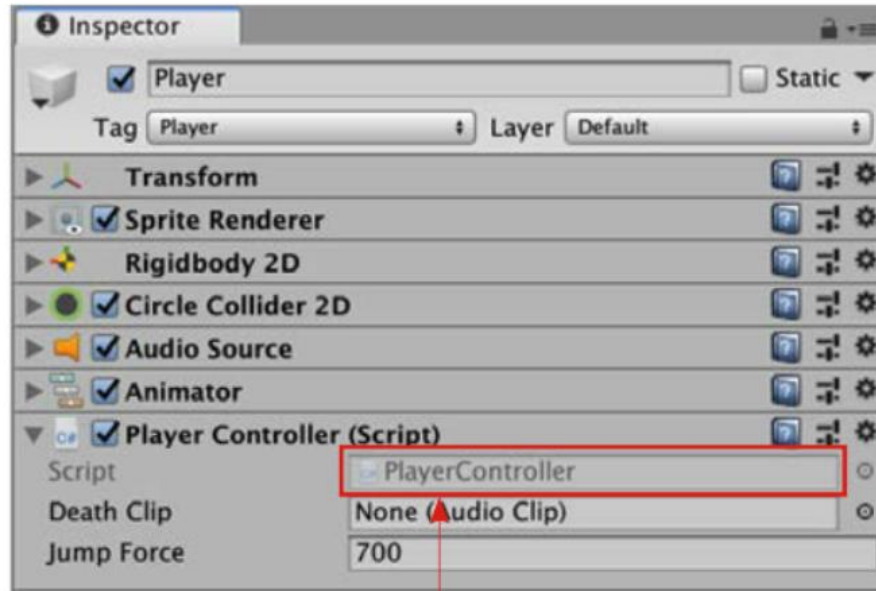
# PlayerController 스크립트 추가



① PlayerController 스크립트를 하이어라키 창의 Player로 드래그&드롭



▶ PlayerController 스크립트 추가



② PlayerController 스크립트를 더블 클릭하여 열기

# PlayerController 스크립트(1)

---

```
using UnityEngine;

// PlayerController는 플레이어 캐릭터로서 Player 게임 오브젝트를 제어함
public class PlayerController : MonoBehaviour {
    public AudioClip deathClip; // 사망 시 재생할 오디오 클립
    public float jumpForce = 700f; // 점프 힘

    private int jumpCount = 0; // 누적 점프 횟수
    private bool isGrounded = false; // 바닥에 닿았는지 나타냄
    private bool isDead = false; // 사망 상태

    private Rigidbody2D playerRigidbody; // 사용할 리지드바디 컴포넌트
    private Animator animator; // 사용할 애니메이터 컴포넌트
    private AudioSource playerAudio; // 사용할 오디오 소스 컴포넌트

    private void Start() {
        // 게임 오브젝트로부터 사용할 컴포넌트들을 가져와 변수에 할당
        playerRigidbody = GetComponent<Rigidbody2D>();
        animator = GetComponent<Animator>();
        playerAudio = GetComponent<AudioSource>();
    }
```



# PlayerController 스크립트(2)

---

```
private void Update() {  
    if (isDead)  
    {  
        // 사망 시 처리를 더 이상 진행하지 않고 종료  
        return;  
    }  
  
    // 마우스 왼쪽 버튼을 눌렀으며 && 최대 점프 횟수(2)에 도달하지 않았다면  
    if (Input.GetMouseButtonDown(0) && jumpCount < 2)  
    {  
        // 점프 횟수 증가  
        jumpCount++;  
        // 점프 직전에 속도를 순간적으로 제로(0, 0)로 변경  
        playerRigidbody.velocity = Vector2.zero;  
        // 리지드바디에 위쪽으로 힘 주기  
        playerRigidbody.AddForce(new Vector2(0, jumpForce));  
        // 오디오 소스 재생  
        playerAudio.Play();  
    }  
}
```

# PlayerController 스크립트(3)

---

```
else if (Input.GetMouseButtonUp(0) && playerRigidbody.velocity.y > 0)
{
    // 마우스 왼쪽 버튼에서 손을 떼는 순간 && 속도의 y 값이 양수라면(위로 상승 중)
    // 현재 속도를 절반으로 변경
    playerRigidbody.velocity = playerRigidbody.velocity * 0.5f;
}

// 애니메이터의 Grounded 파라미터를 isGrounded 값으로 갱신
animator.SetBool("Grounded", isGrounded);
}

private void OnTriggerEnter2D(Collider2D other) {
    if (other.tag == "Dead" && !isDead)
    {
        // 충돌한 상대방의 태그가 Dead이며 아직 사망하지 않았다면 Die() 실행
        Die();
    }
}
```

# PlayerController 스크립트(4)

---

```
private void Die() {  
    // 애니메이터의 Die 트리거 파라미터를 셋  
    animator.SetTrigger("Die");  
  
    // 오디오 소스에 할당된 오디오 클립을 deathClip으로 변경  
    playerAudio.clip = deathClip;  
    // 사망 효과음 재생  
    playerAudio.Play();  
  
    // 속도를 제로(0, 0)로 변경  
    playerRigidbody.velocity = Vector2.zero;  
    // 사망 상태를 true로 변경  
    isDead = true;  
}
```

# PlayerController 스크립트(5)

---

```
private void OnCollisionEnter2D(Collision2D collision) {  
    // 어떤 콜라이더와 닿았으며, 충돌 표면이 위쪽을 보고 있으면  
    if (collision.contacts[0].normal.y > 0.7f)  
    {  
        // isGrounded를 true로 변경하고, 누적 점프 횟수를 0으로 리셋  
        isGrounded = true;  
        jumpCount = 0;  
    }  
}  
  
private void OnCollisionExit2D(Collision2D collision) {  
    // 어떤 콜라이더에서 떴어진 경우 isGrounded를 false로 변경  
    isGrounded = false;  
}  
}
```

# deathClip 오디오 할당

① Player 게임 오브젝트 선택



▶ deathClip에 오디오 클립 할당

② PlayerController 컴포넌트의 Death Clip 필드 옆의 선택 버튼 클릭



③ die 오디오 클립을 더블 클릭

# 테스트

---

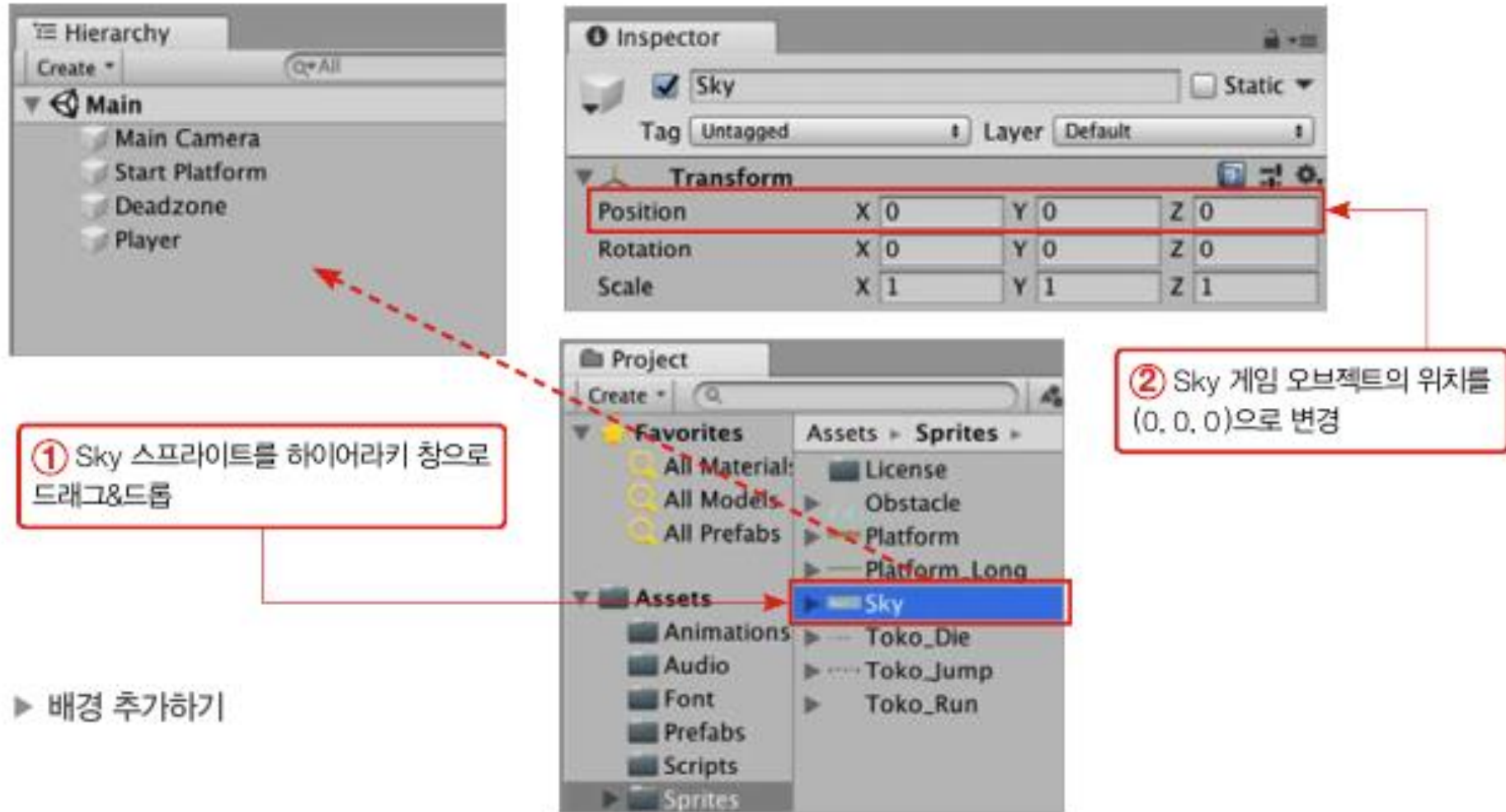
- 이제 ▶ 버튼을 눌러 게임 실행 후
- 마우스 왼쪽 버튼을 누르며 점프가 잘 실행되는지 테스트



# 배경 추가하기

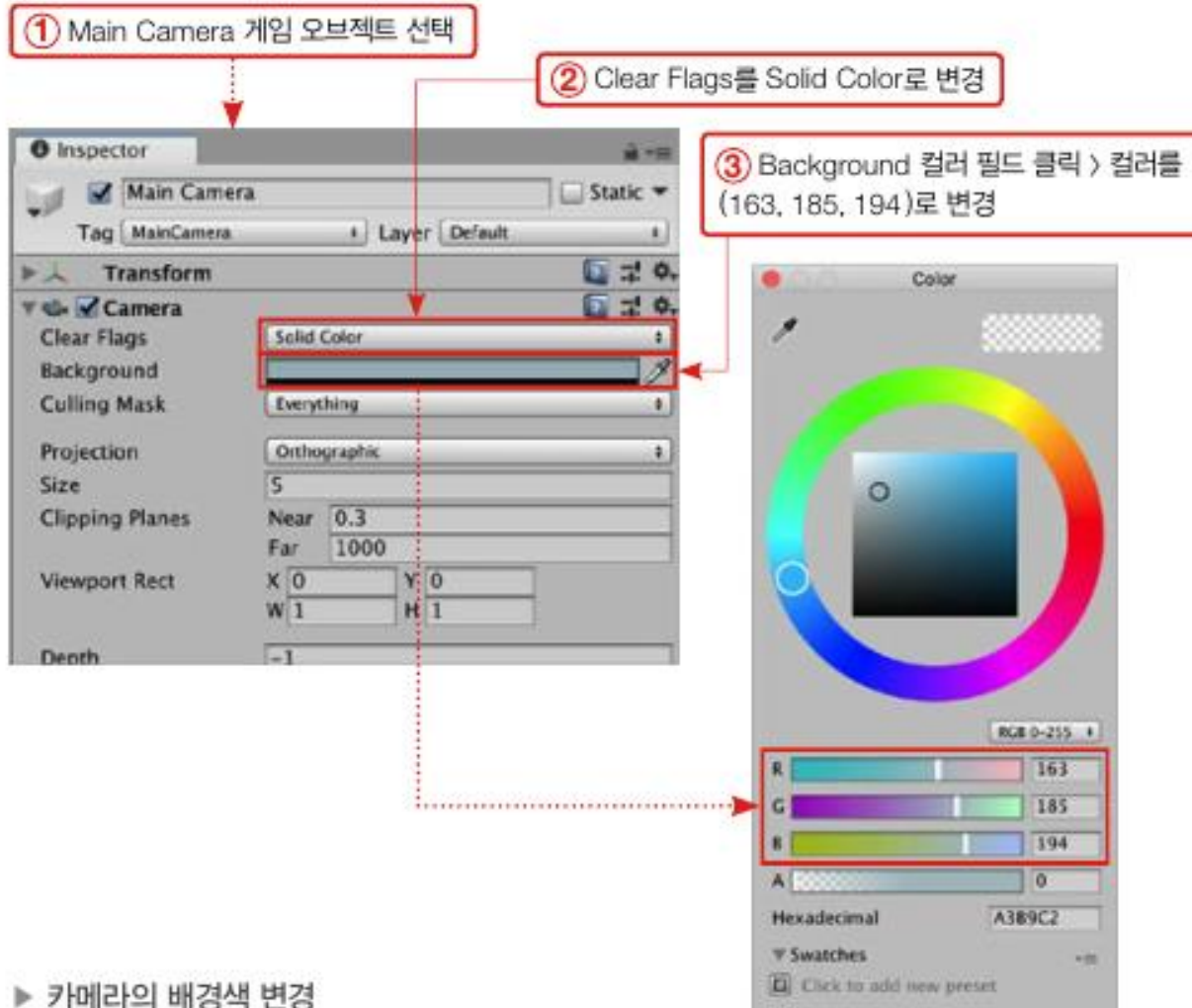
---

# 배경 추가



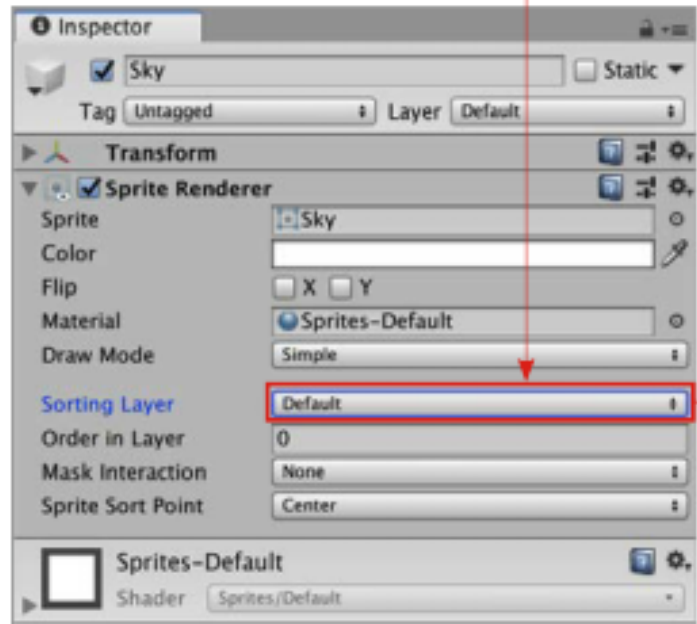


# 카메라 배경색 변경



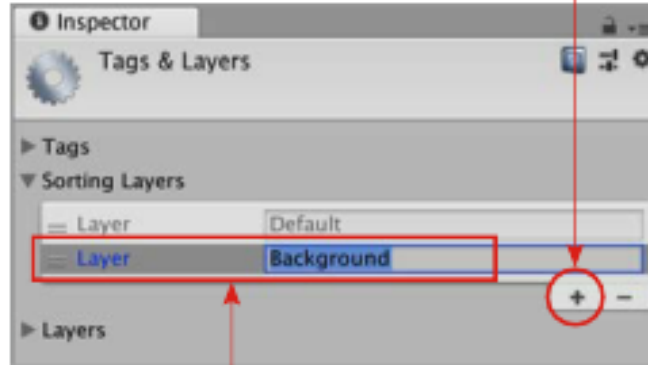
# 정렬 레이어 추가

① Sorting Layer의 Default 클릭 > Add Sorting Layer... 클릭



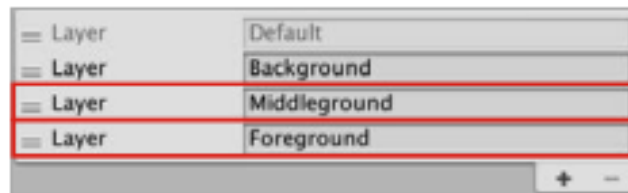
▶ 정렬 레이어 추가

② Sorting Layers 리스트의 + 버튼 클릭



③ 생성된 Layer 이름을 Background로 변경

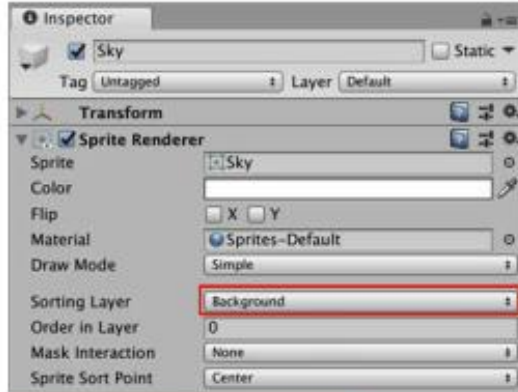
④ + 버튼 클릭 > 생성된 Layer 이름을 Middleground로 변경



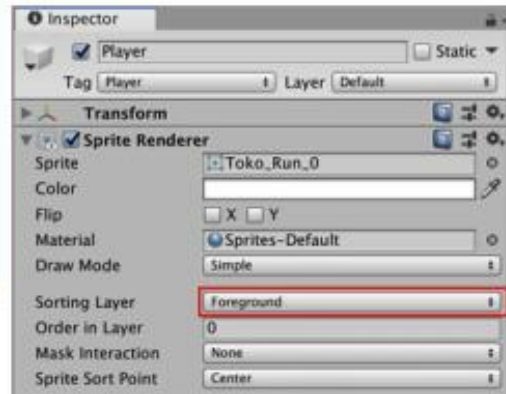
⑤ + 버튼 클릭 > 생성된 Layer 이름을 Foreground로 변경

# 정렬 레이어 할당하기

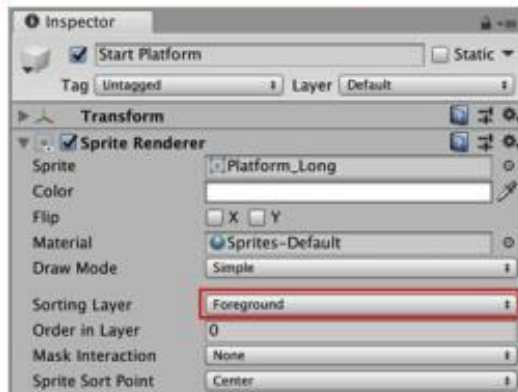
① Sky 게임 오브젝트 선택 > Sorting Layer를 Background로 변경



② Player 게임 오브젝트 선택 > Sorting Layer를 Foreground로 변경



③ Start Platform 게임 오브젝트 선택 > Sorting Layer를 Foreground로 변경



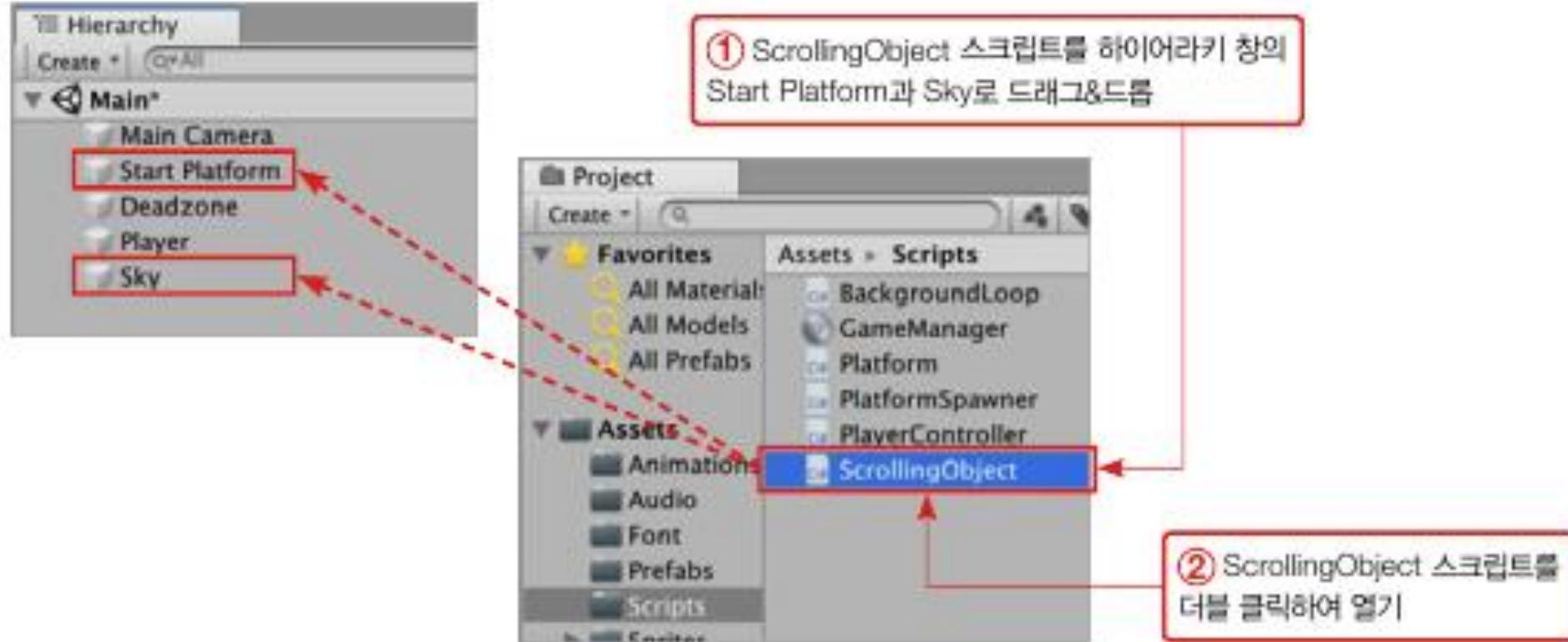
Sky가 Player와 Start Platform 뒤에 그려짐

▶ 정렬 레이어 할당하기

# 배경과 발판 움직이기

---

# ScrollingObject 스크립트 추가



▶ ScrollingObject 스크립트 추가하기

# ScrollingObject 스크립트 완성하기

---

```
using UnityEngine;

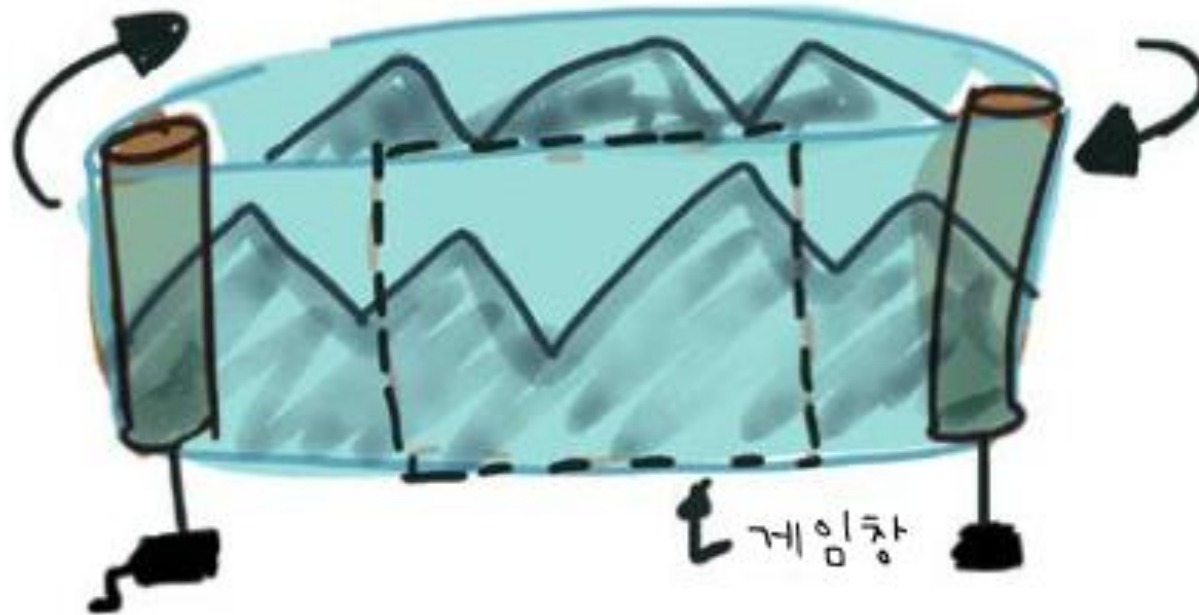
// 게임 오브젝트를 계속 왼쪽으로 움직이는 스크립트
public class ScrollingObject : MonoBehaviour {
    public float speed = 10f; // 이동 속도

    private void Update() {
        // 초당 speed의 속도로 왼쪽으로 평행이동
        transform.Translate(Vector3.left * speed * Time.deltaTime);
    }
}
```

# 반복되는 배경 만들기

---

- 배경은 크래킹 박스와 같이 끊임 없이 반복 스크롤링되어야 합니다.

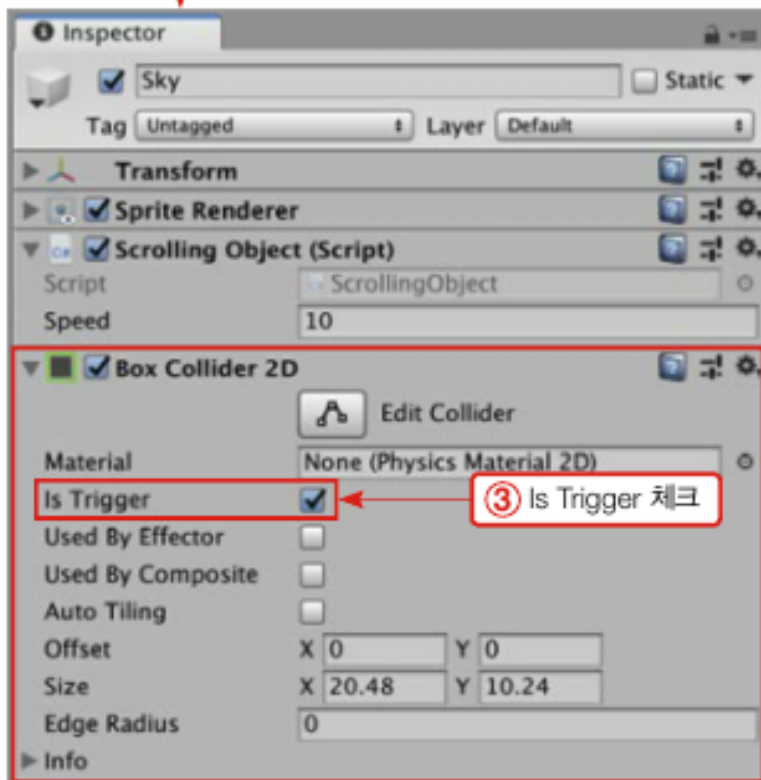




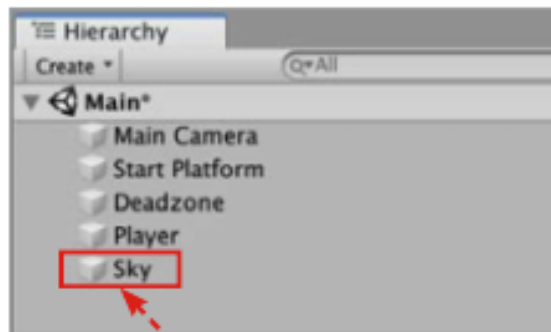
# BoxCollider2D와 BackgroundLoop

① Sky 게임 오브젝트 선택

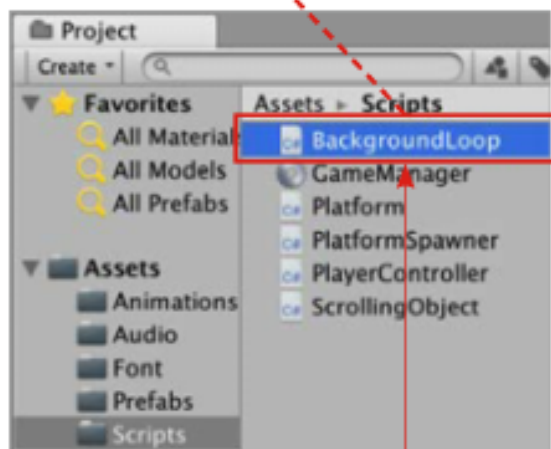
② Box Collider 2D 컴포넌트 추가  
(Add Component > Physics 2D >  
Box Collider 2D)



③ Is Trigger 체크



④ BackgroundLoop 스크립트를 하이어라키  
창의 Sky로 드래그&드롭



⑤ BackgroundLoop 스크립트를  
더블 클릭으로 열기

▶ 박스 콜라이더 2D와 BackgroundLoop 준비하기



# BackgroundLoop 스크립트(1)

---

```
using UnityEngine;
```

```
// 왼쪽 끝으로 이동한 배경을 오른쪽 끝으로 재배치하는 스크립트
```

```
public class BackgroundLoop : MonoBehaviour {
```

```
    private float width; // 배경의 가로 길이
```

```
    private void Awake() {
```

```
        // BoxCollider2D 컴포넌트의 Size 필드의 x 값을 가로 길이로 사용
```

```
        BoxCollider2D backgroundCollider = GetComponent<BoxCollider2D>();
```

```
        width = backgroundCollider.size.x;
```

```
    }
```

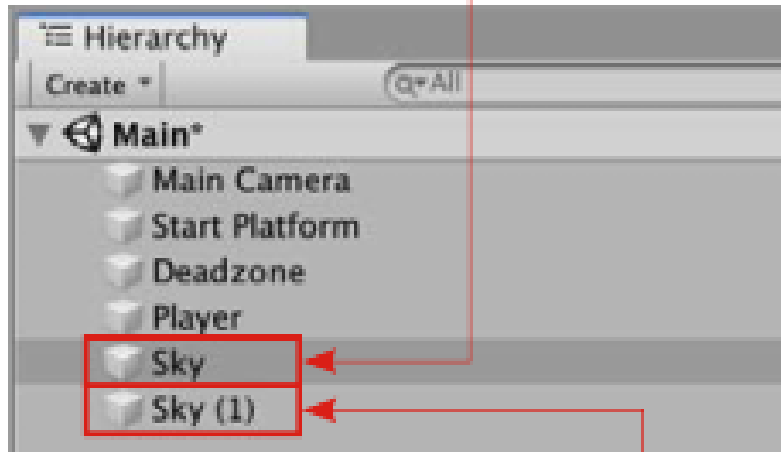
# BackgroundLoop 스크립트(2)

---

```
private void Update() {  
    // 현재 위치가 원점에서 왼쪽으로 width 이상 이동했을 때 위치를 재배치  
    if (transform.position.x <= -width)  
    {  
        Reposition();  
    }  
}  
  
// 위치를 재배치하는 메서드  
private void Reposition() {  
    // 현재 위치에서 오른쪽으로 가로 길이 * 2만큼 이동  
    Vector2 offset = new Vector2(width * 2f, 0);  
    transform.position = (Vector2) transform.position + offset;  
}  
}
```

# Sky 추가 배치

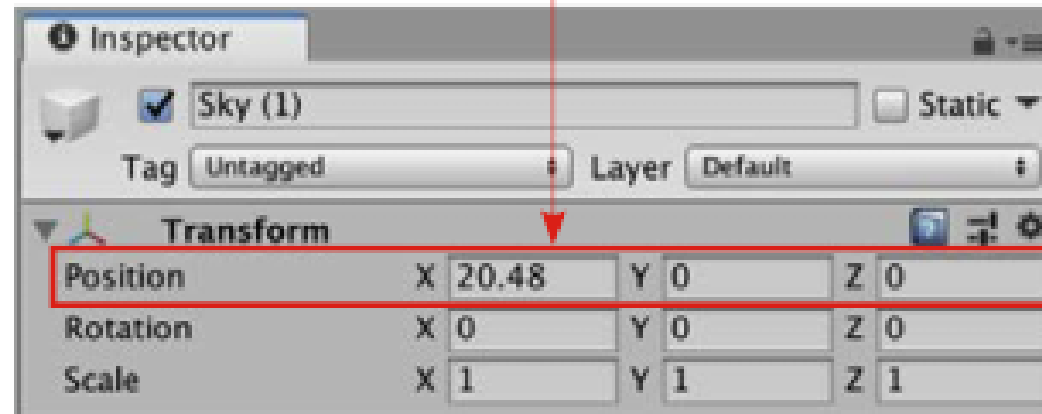
① Sky 게임 오브젝트 선택 > [Ctrl+D]로 복제



▶ Sky 추가 배치

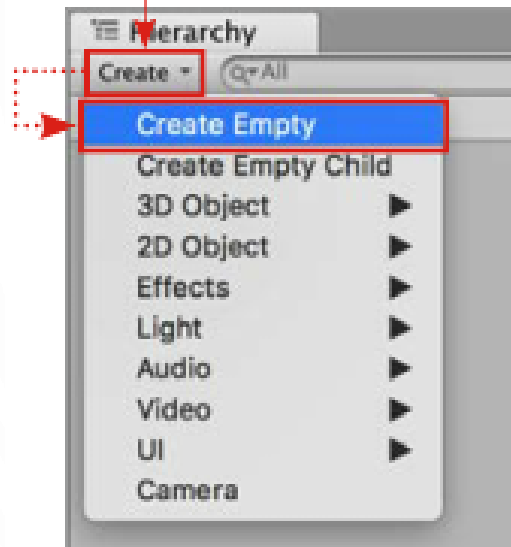
복제된 Sky (1)

② Sky (1) 게임 오브젝트의 위치를 (20.48, 0, 0)으로 변경

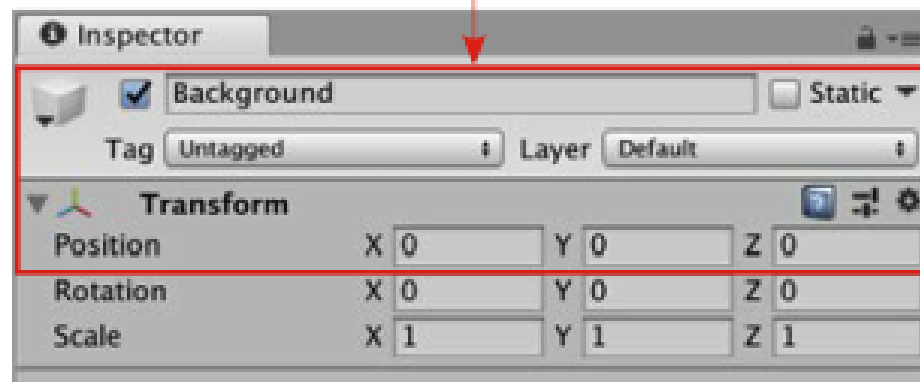


# 배경 오브젝트 정리

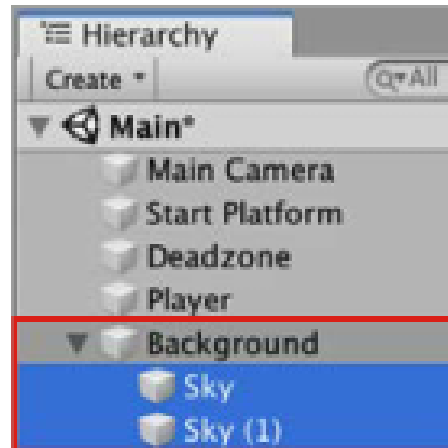
① 빈 게임 오브젝트 생성(Create > Create Empty)



② 이름을 Background로 변경, 위치를 (0, 0, 0)으로 변경



③ Sky와 Sky (1)을 Background의 자식으로 만들기

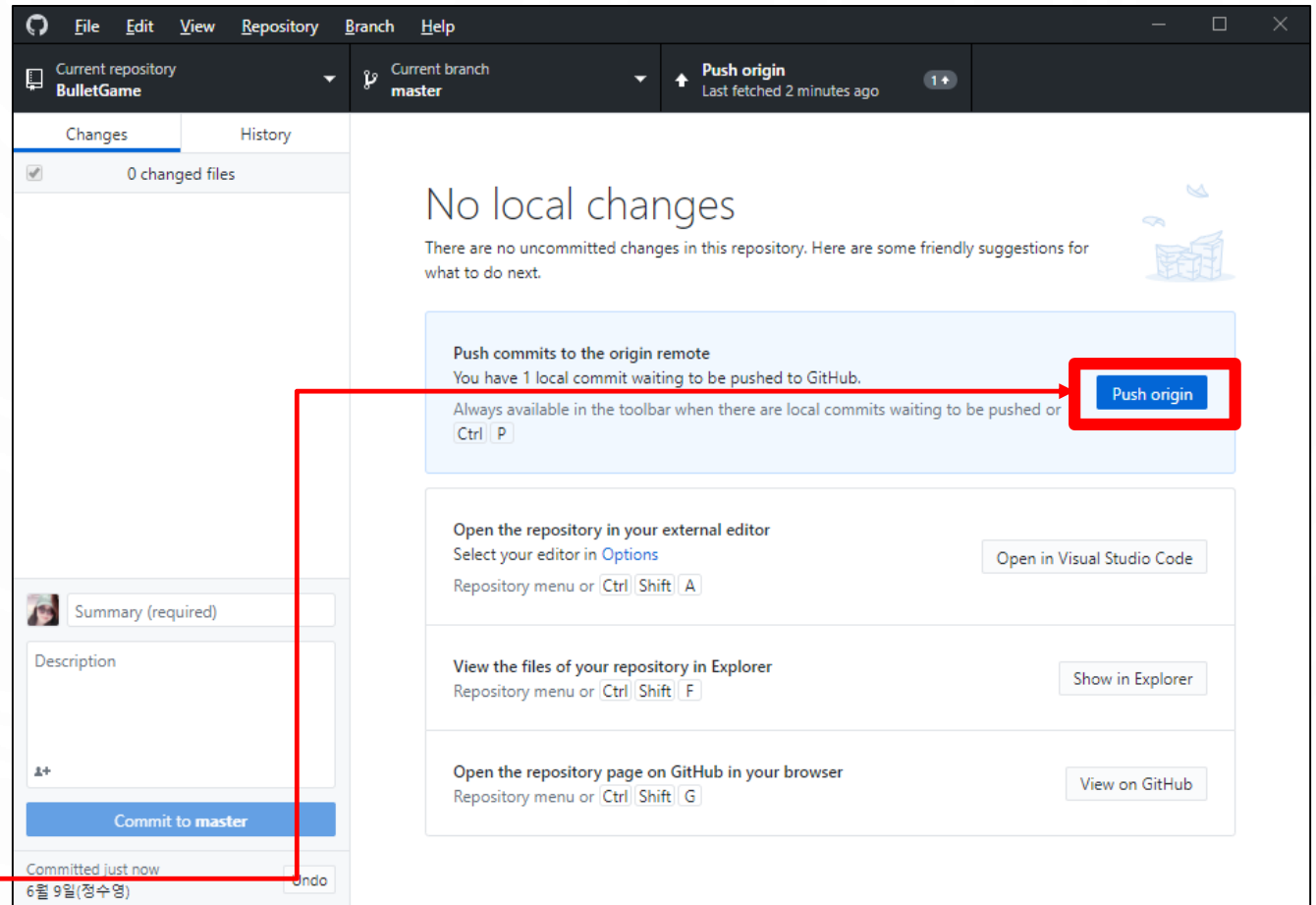
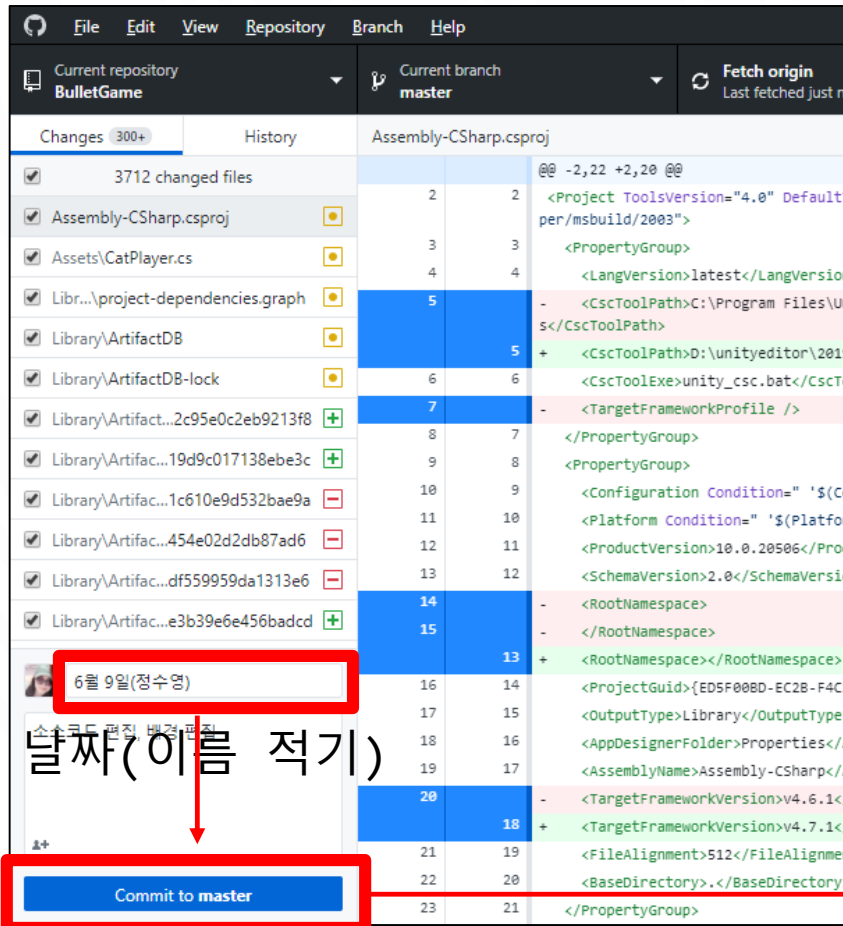


# GITHUB PUSH

---

# 깃허브 올리기

- Unity 프로젝트 저장 후 모두 닫고 Git Desktop 열기



# 에러 날 때!

The image shows the GitHub Desktop application window on the left and its 'Options' dialog box on the right. In the main window, the 'File' menu is highlighted with a red box, and a red arrow points from a box labeled 'Options' to the 'Git' option in the 'Options' dialog. The 'Options' dialog has a sidebar with 'Accounts', 'Integrations', 'Git' (highlighted with a red box), 'Appearance', and 'Advanced'. The main area of the dialog shows 'Name' as 'Soori' and 'Email' as 'soorichu@gmail.com'. At the bottom right of the dialog, the 'Save' button is highlighted with a red box. The main window also displays 'Let's get started!' and three options: 'Clone a repository from the Internet...', 'Create a New Repository on your hard drive...', and 'Add an Existing Repository from your hard drive...'.

그냥 Name과 Email이 잘 채워진 것만 확인하고  
Save 버튼 누르세요.  
이제 다시 Commit > Push 시도해보세요. ^^

# END

---

참고 : 레트로의 유니티 게임 프로그래밍 에센스