ZEROPEN

: 유니티로 만드는 3D마리오파티 미니게임 모작

목차

Chapter 1

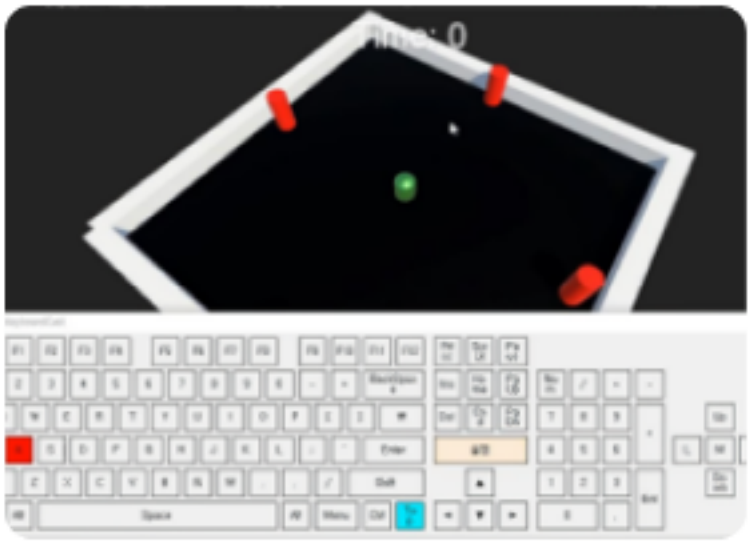
수업에 앞서

01 동아리 소개

동아리 소개하기

Zer0pen은 게임 개발을 전문적으로 다루는 동아리입니다.
Zer0pen에 속한 개발자들은 게임 기획과 Unity 엔진을 이용한 2D 및 3D 게임 개발에 대한 지식을 깊이 있게 배울 수 있습니다.

디자이너들은 게임 디자인과 관련된 지식, 도트 디자인, 모델링 기술 등을 배우게 됩니다.
이러한 능력을 바탕으로 게임을 꾸며나가는 방법 또한 배울 수 있습니다.



▲개발자 작품

▲디자이너 수상작

▲디자이너 작품

▲개발자 작품

02 게임 장르 설명

게임 장르 살펴보기

게임 장르에는 여러 가지 종류가 있습니다. 대표적인 게임 장르는 FPS, 배틀로얄, RPG, 스포츠게임 등이 있습니다.

< 예시 게임들 >

▲에이펙스

▲오버워치

▲발로란트

▲리그오브레전드

▲가디언 테일즈

▲피파23

03 게임엔진 설명

게임엔진 살펴보기

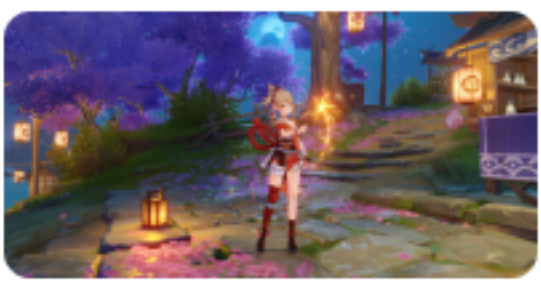
게임은 다양한 게임엔진을 통해서 개발할 수 있습니다. 오디오나 물리엔진, 게임 프레임워크와 같은 필수 요소들을 소스 코드와 연관시켜 쉽게 수정할 수 있게 한 것을 게임엔진이라고 합니다.
그 중 대표적인 두 가지 엔진을 소개해 보겠습니다.

< Unity >

- 직관적인 gui, 만화적인 그래픽, 낮은 요구사양, 간편한 빌드 등의 이유로 인해 1인 개발자가 활용하기 좋음
- 에셋스토어를 통해 스크립트, ai, 플러그인 등을 쉽게 받아올 수 있음
- 사용자층이 넓어 원하는 정보 획득이 편리함

< Unreal >

- 높은 품질의 그래픽, 풍부한 기능, 여러 플랫폼 지원으로 개발이 용이함
- 독자적인 이펙트 시스템인 나이아가라(Niagara)를 활용할 수 있음
- 눈에 띄는 스킬과 시각적 효과를 구현할 수 있음



▲원신

▲발로란트

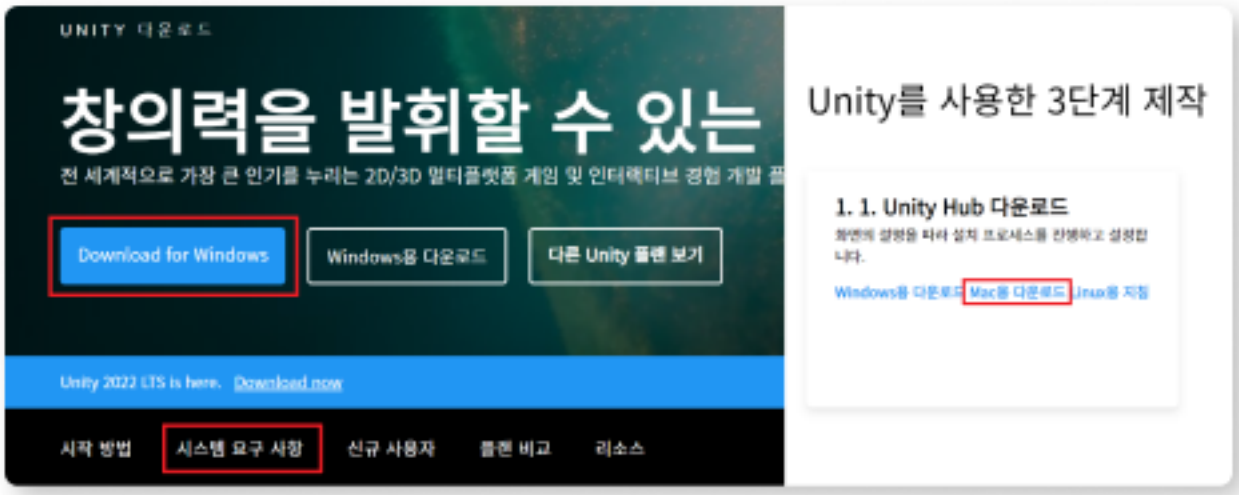
Chapter 2

Unity 사용법

01 Unity 설치

Unity Hub 설치하기

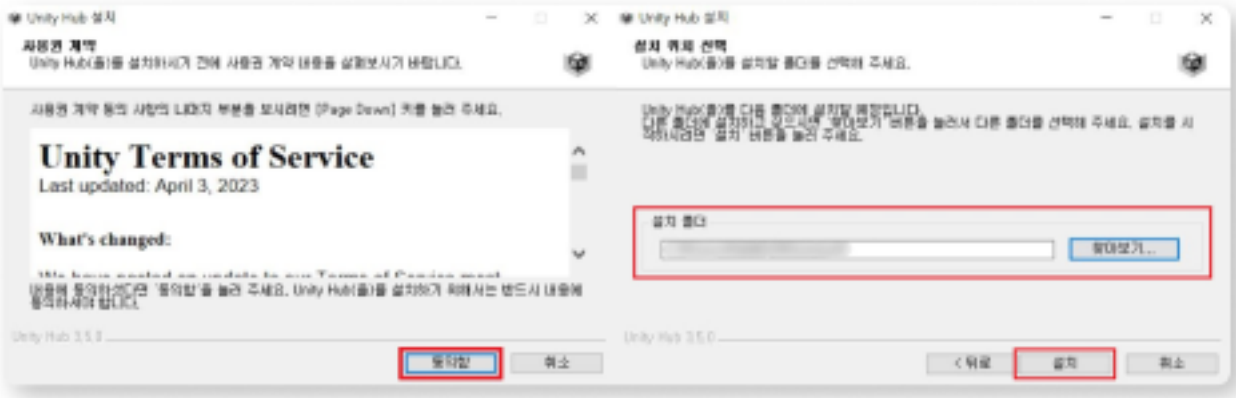
https://unity.com/kr에 접속합니다.
화면 중앙의 [다운로드] 버튼을 클릭하면 다운로드 페이지가 나타납니다.



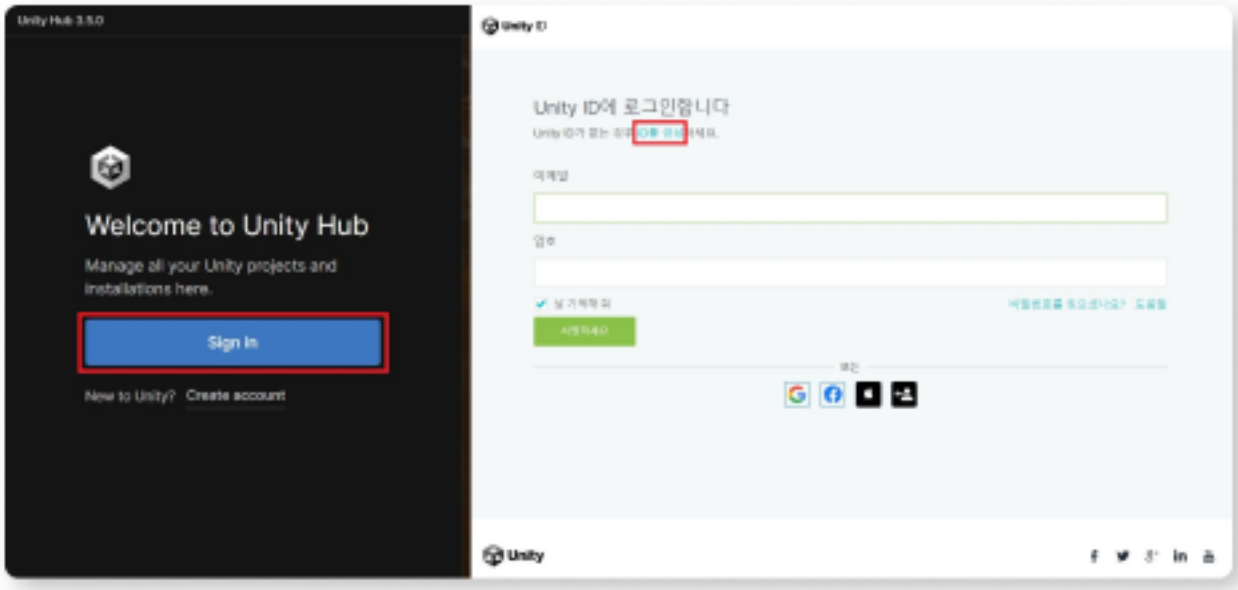
다운로드 페이지에 적혀있는 유니티의 '시스템 요구 사항'을 보고, 내 컴퓨터가 요구조건을 충족하는지 확인합니다.
[Download for Windows] 버튼 혹은 'Mac 다운로드'를 클릭해 유니티 허브를 다운로드합니다.

01 Unity 설치

Unity Hub 설치하기



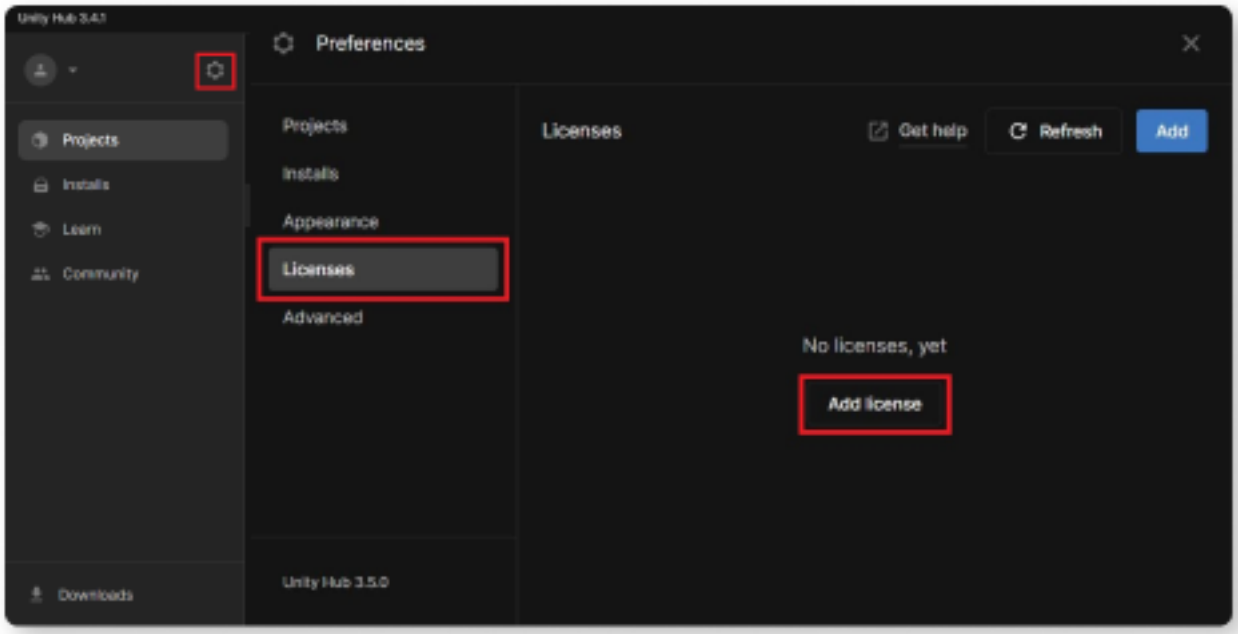
다운로드한 파일을 더블 클릭합니다.
'사용권 계약' 창이 나타나면, [동의함] 버튼을 클릭합니다. 다음으로 나타난 '설치 위치 선택' 창에서도 설치할 폴더를 지정한 후 [설치] 버튼을 클릭합니다.



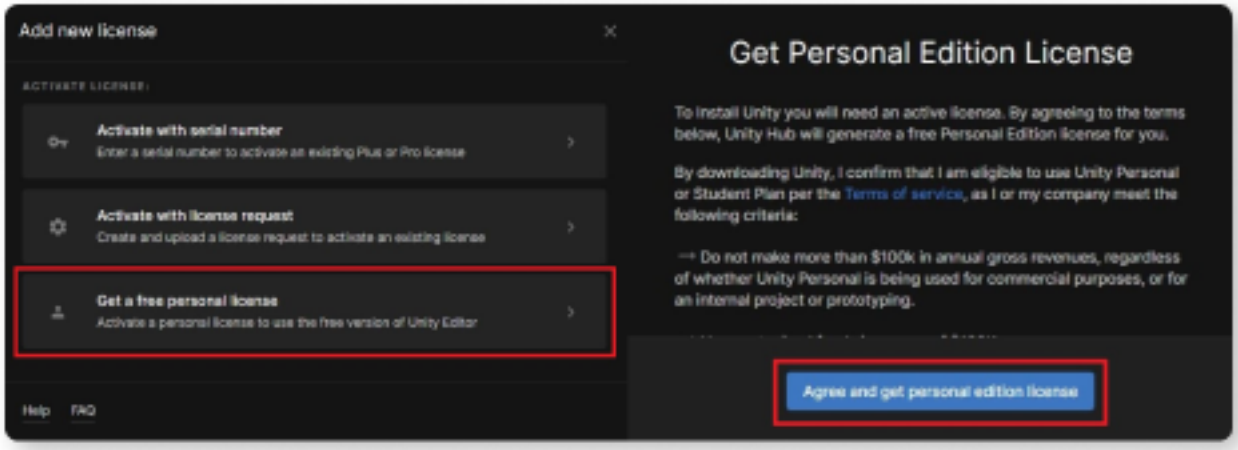
설치가 완료된 후, 'Unity Hub'가 실행되었다면, [Sign in] 버튼을 클릭합니다. 버튼을 클릭하면 Unity ID로 로그인할 수 있는 페이지가 나타납니다.
이미 유니티 ID가 있다면 이메일과 암호를 입력해 로그인하고, 새롭게 계정을 만들려면 'ID를 생성'을 클릭합니다.

01 Unity 설치

Unity License 활성화하기



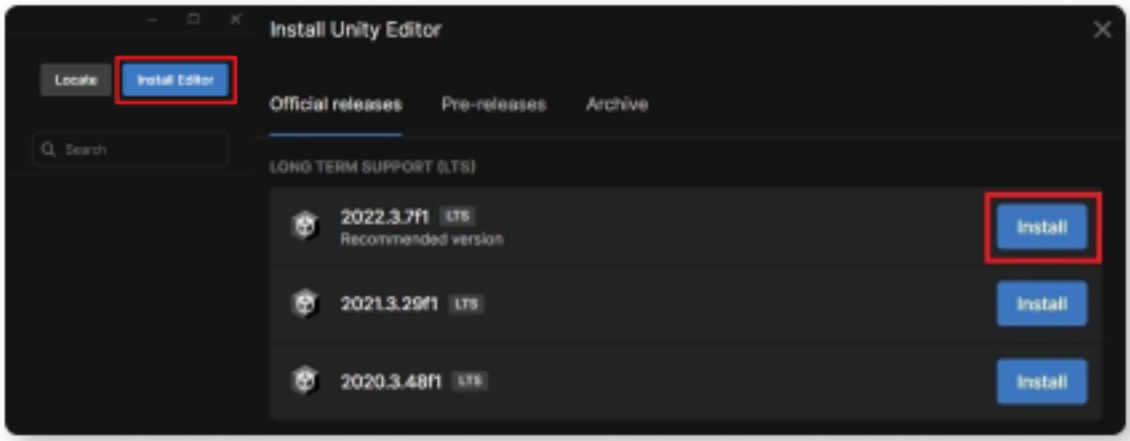
프로필 오른쪽에 위치한 설정 아이콘을 클릭하면 'Preferences' 창이 나타납니다. 왼쪽 메뉴에서 'Licenses'를 선택한 후 [Add license] 버튼을 클릭합니다.



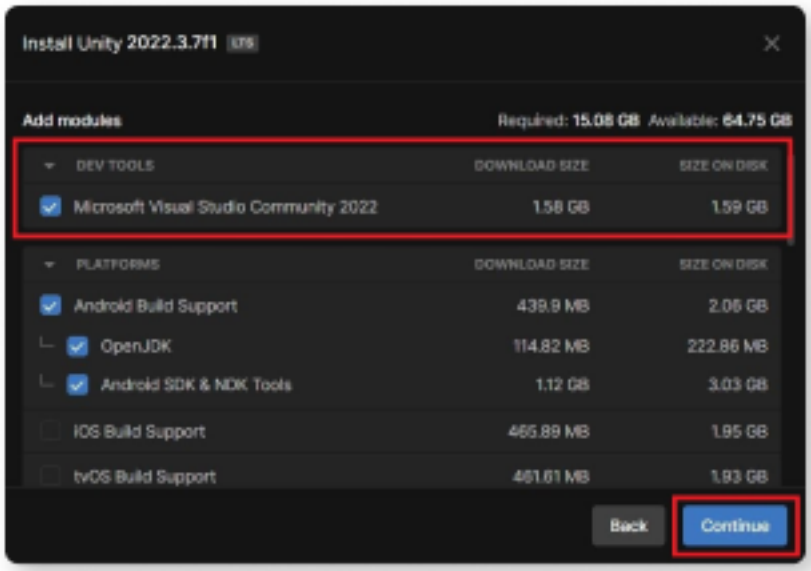
'Get a free personal license' 라이선스를 선택한 후 [Agree and get personal edition license] 버튼을 클릭해 무료 버전을 사용합니다.

01 Unity 설치

Unity Editor 설치하기



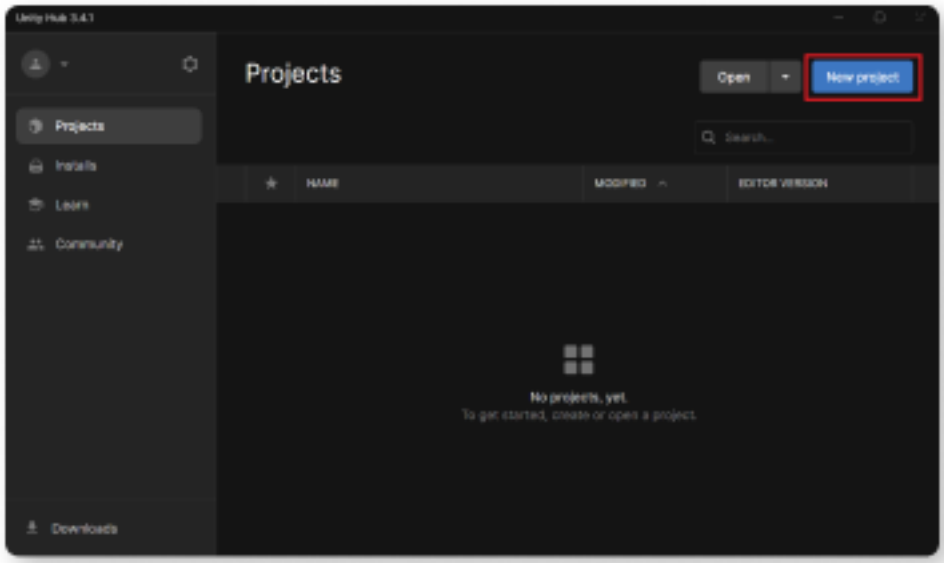
오른쪽 상단에서 'Install Editor' 버튼을 클릭한 후, 설치할 버전의 유니티로 [Install] 버튼을 클릭합니다.
유니티는 디스크 용량을 많이 차지하므로 최소 13GB 정도는 공간을 마련하고 설치를 진행합시다.



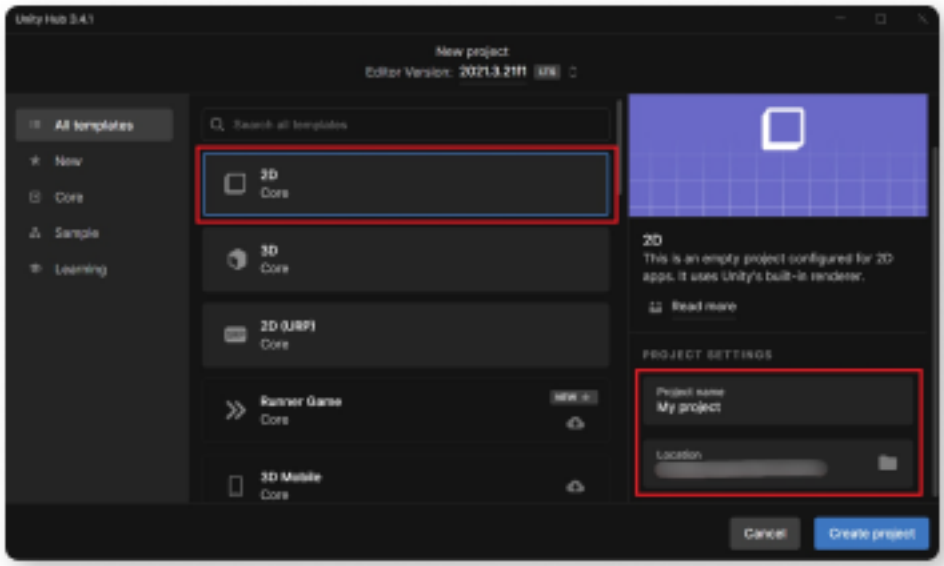
에디터와 함께 설치할 모듈을 선택합니다. 'Visual Studio'라는 에디터는 프로그래밍에 꼭 필요하므로 반드시 포함합니다. 선택이 끝나면 [Continue] 버튼을 클릭합니다.

02 Unity 기초

Unity 새 프로젝트 만들기



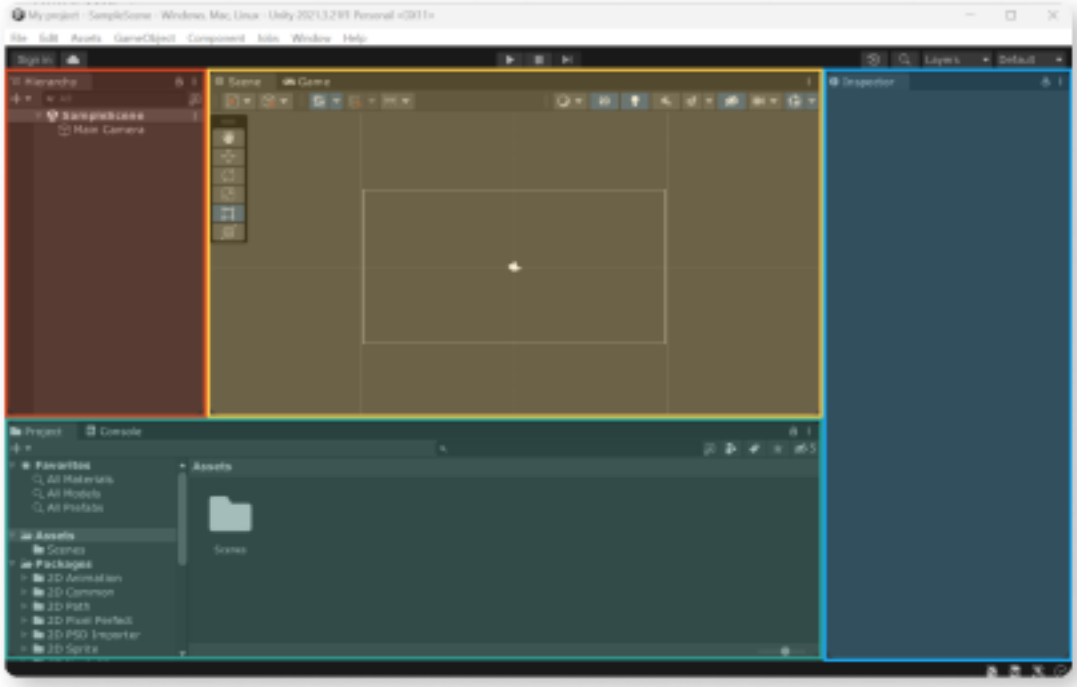
Unity Hub를 실행시키고, [New project] 버튼을 클릭합니다.



'New project' 창에서 프로젝트 이름과 저장할 위치를 정해줍니다.
2D게임을 만들 예정이므로 개발환경은 2D로 설정하고, [Create project] 버튼을 클릭합니다.

02 Unity 기초

Unity 개발화면 이해하기



프로젝트가 생성되면 위와 같은 개발화면이 나타납니다.
Unity 개발화면 속 존재하는 여러 요소에 대해 알아봅시다.

Unity 메인 창 살펴보기

- Hierarchy 창

게임에 필요한 물체나 이펙트, 빛 등의 오브젝트를 생성해 관리할 수 있습니다.
현재 씬에 어떤 물체가 포함되어 있는지 확인할 수 있습니다.

- Inspector 창

선택한 오브젝트의 정보를 수정하고 관리할 수 있습니다.
여러 속성(Component)들을 추가할 수 있습니다.
오브젝트는 컴포넌트가 제공하는 기능을 사용할 수 있습니다.

02 Unity 기초

Unity 메인 창 살펴보기

- Project 창

외부 파일을 불러올 수 있습니다.
종류별로 폴더를 만들어 여러 파일을 관리할 수 있습니다.

- Console 창

로그 혹은 에러를 확인할 수 있습니다.

일반 로그 (회색) : 일반적인 정보가 표시됨
경고 로그 (노란색) : 권장할 만한 수정사항이 표시됨
에러 로그 (빨간색) : 잘못된 문법 등으로 명령을 실행할 수 없을 때 표시됨

- Scene 창

Scene에 존재하는 게임 오브젝트들을 시각적으로 편집할 수 있습니다.

- Game 창

플레이어가 실제로 보게 될 화면을 표시합니다.

Unity 메인 창 살펴보기

아래 툴을 사용할 수 있는 단축키는 순서대로 Q, W, E, R, T, Y 입니다.
- 핸드 툴 : 씬 카메라를 움직입니다.
- 평행이동 툴 : 오브젝트를 이동시킵니다.
- 회전 툴 : 오브젝트를 회전시킵니다.
- 스케일 툴 : 오브젝트의 크기를 조정합니다.
- 렉트 툴 : UI와 오브젝트의 크기를 조정합니다.
- 트렌스폼 툴 : 평행이동, 회전, 스케일 툴을 하나로 합친 툴입니다.

Chapter 3

플레이어 이동 구현

01 자료형과 좌표계 설명

다양한 자료형 살펴보기

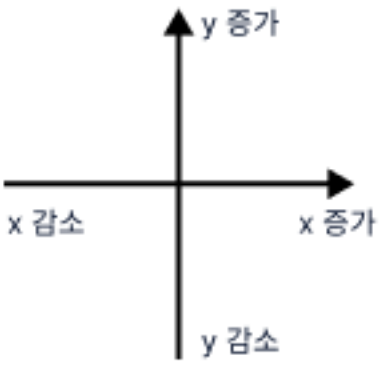
자료형		크기	값의 표현범위
정수형	char	1바이트	-128이상 +127이하
	short	2바이트	-32,768이상 +32,767 이하
	int	4바이트	-2,147,483,648이상 +2,147,483,647이하
	long	4바이트	-2,147,483,648이상 +2,147,483,647이하
	long long	8바이트	-9,223,372,036,854,775,808이상
			+9,223,372,036,854,775,807이하
실수형	float	4바이트	$\pm 3.4\times10^{-37}$ 이상 $\pm 3.4\times10^{+38}$이하
	double	8바이트	$\pm 1.7\times10^{-307}$ 이상 $\pm 1.7\times10^{+308}$이하
	long double	8바이트 이상	double 이상의 표현범위

int, float, double, long, char, bool 등 여러 종류의 자료형이 존재합니다.
이러한 자료형들은 게임에서 변수를 선언할 때 사용됩니다.

- int (정수) ex) -2, 2, 50 ...
- float, double (실수) ex) 3.141592, 4.2 ...
- char (문자) ex) 'A', 'a' ...
- bool (참과 거짓) ex) true, false

Unity 좌표계 살펴보기

좌표계는 간단히 말하면 수학에 나오는 평면 좌표입니다.
평면좌표는 평면에서 x축과 y축으로 원하는 물체의 위치를 좀 더 알기 쉽게 한 것입니다.
물체가 이동하는 방향에 따라 x 좌표, y 좌표가 증가 혹은 감소합니다.



02 플레이어 이동 원리 설명

이동 방식 살펴보기

오브젝트의 이동 방식은 크게 Transform을 이용한 방식과 Rigidbody를 이용한 방식 두 가지가 있습니다.

Transform 방식

Transform 속성은 게임 오브젝트의 위치, 회전, 크기 등을 포함하고 있습니다.
Transform의 대표적인 이동은 두 가지가 있습니다.

```
public GameObject cubeTransform;
void Start ()
{
        // cubeTransform의 x위치 +2 만큼 이동
        cubeTransform.transform.position = new Vector3 (2,0,0);
}
```

단순 오브젝트 이동의 경우 transform.position (Vector 변수)를 사용합니다.

```
public GameObject cubeTransform;
public float speed = 1.0f;
void Update()
{
        float x = Input.GetAxis("Horizontal");
        float z = Input.GetAxis("Vertical");

        // 속도를 조절합니다.
        float translateMove = speed * Time.deltaTime;

        cubeTransform.transform.Translate
        (x * translateMove , 0 , z * translateMove );
}
```

매 프레임 실시간으로 계속 움직이는 경우 transform.Translate (Vector 변수)를 사용합니다.

02 플레이어 이동 원리 설명

Transform 방식의 특징

transform 이동은 당연히 지상에서의 움직임이나 점프 등의 기능을 가진 캐릭터가 아닌 공중에 떠 있는 캐릭터에 적합한 방식입니다. 혹은 캐릭터가 아니더라도 단순 오브젝트의 이동에 적합합니다.

Rigidbody 방식

Rigidbody 속성은 물리 재질(physic material) 등을 이용해 다양한 물리법칙을 만들 수 있습니다.
Rigidbody의 대표적인 이동은 두 가지가 있습니다.

```
public Rigidbody playerRb; // 리지드바디 변수
public float speed = 2.0f;
Vector3 moveDirection;

    void Start()
    {
        // 리지드바디 컴포넌트를 받습니다.
        playerRb.GetComponent<Rigidbody>();
    }

    void Update()
    {
        float x = Input.GetAxis("Horizontal");
        float z = Input.GetAxis("Vertical");

        moveDirection = new Vector3(x, 0, z);
        playerRb.AddForce(moveDirection * speed); // AddForce 적용
    }
```

첫 번째로 Rigidbody.AddForce (힘 더하기)를 사용해 이동할 수 있습니다.

02 플레이어 이동 원리 설명

Rigidbody 방식

```
public Rigidbody playerRb;
public float speed = 2.0f;
Vector3 moveDirection;

    void Start()
    {
        playerRb.GetComponent<Rigidbody>();
    }

    void Update()
    {
        float x = Input.GetAxis("Horizontal");
        float z = Input.GetAxis("Vertical");

        moveDirection = new Vector3(x, 0, z);
        //playerRb.AddForce(moveDirection * speed);
        playerRb.velocity = (moveDirection * speed); // 가속도를 적용
    }
```

두 번째로 Rigidbody.Velocity (가속 더하기)를 사용해 이동할 수 있습니다.

Rigidbody 방식의 특징

Rigidbody가 적용된 캐릭터의 경우 transform.position 명령이 가능하지만, 물리엔진을 사용하기 때문에 마찰로 인해 지상에서 비정상적인 움직임을 가집니다.

Rigidbody는 transform과 character controller와는 달리 중력 여부 / 정도를 조절할 수 있기 때문에 별도의 코드 없이 간단하게 물리엔진을 사용할 수 있지만, 물리엔진에 필요한 연산이 많이 들어가기에, 최적화에 따른 여러 가지 공부가 필요합니다.

03 플레이어 이동 코드 구현

플레이어 이동 코드 살펴보기

```
// Point 1.
h = Input.GetAxis("Horizontal");       // 가로축
v = Input.GetAxis("Vertical");         // 세로축

// Point 2.
transform.position += new Vector3(h, 0, v) * Speed * Time.deltaTime;
```

플레이어 1의 이동방식은 화살표나 wasd를 입력을 받아 캐릭터를 모든 방향으로 움직일수있게 해놨습니다. h,v는 입력받는 키입력에 따라 1,0,-1로 값이 바뀝니다. 그로인해 속도와Time.deltaTime을 곱해줍니다.

```
if (Input.GetKey(KeyCode.Keypad4)) {
    transform.Translate(-speed * Time.deltaTime, 0, 0);
}
if (Input.GetKey(KeyCode.Keypad6)) {
    transform.Translate(speed * Time.deltaTime, 0, 0);
}
```

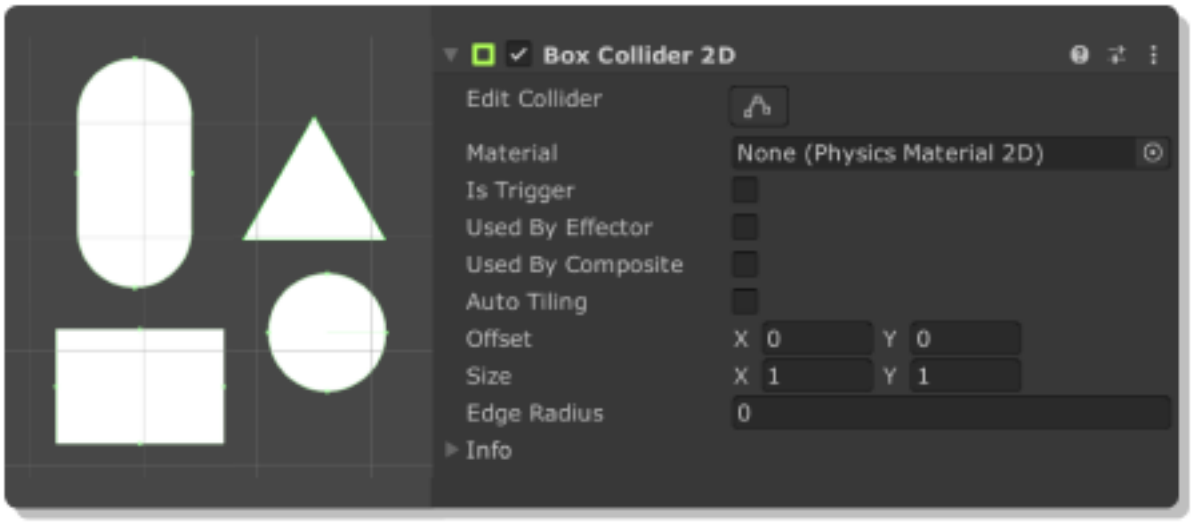
플레이어 2의 이동 방식은 키보드 숫자판에있는 4와 6입니다.
4를 눌렀을때는 속도에 음수를 곱해서 왼쪽으로 이동하게 하였고
6을 눌렀을때는 속도에 양수를 곱해서 오른쪽으로 이동하게 하였습니다.

Chapter 4

플레이어 충돌 설명

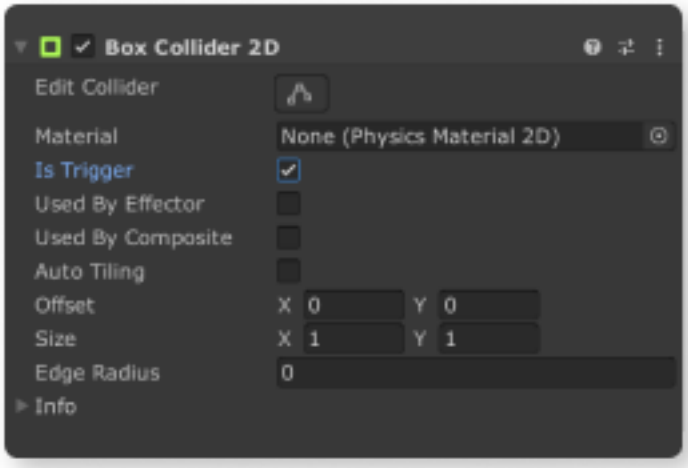
01 플레이어 충돌 원리 설명

Collider 살펴보기



Collider 컴포넌트를 이용하면 충돌을 체크할 수 있습니다. 위 사진에서 보이는 초록색 테두리가 이 물체에 무언가 닿았을 때 인식할 수 있는 범위입니다.

OnTriggerEnter() 살펴보기

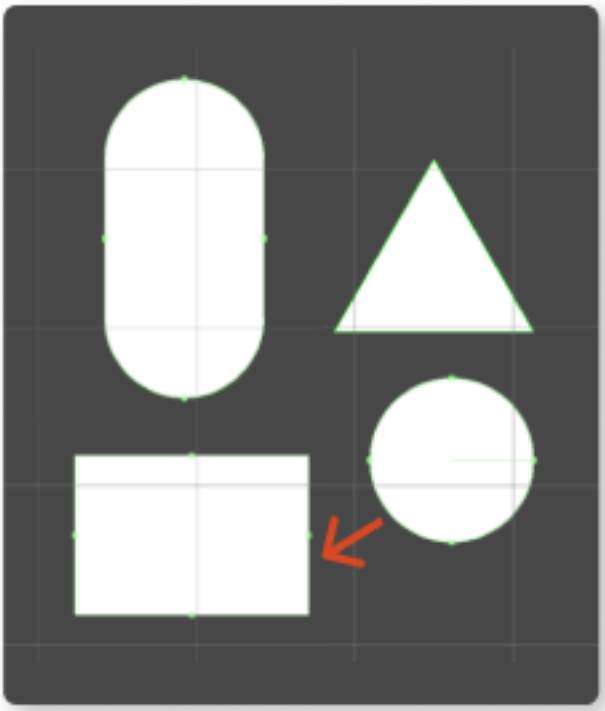


OnTriggerEnter() 함수를 사용하는 경우 충돌하는 두 개의 게임 오브젝트 모두 Collider 컴포넌트를 가지고 있어야 합니다.
그리고 그중 하나는 isTrigger가 활성화되어 있어야 합니다. (Collider 함수에서 isTrigger를 체크)

또 적어도 하나의 게임 오브젝트가 Rigidbody 컴포넌트를 가지고 있어야 합니다. 그래야 OnTriggerEnter함수에서 닿은 물체가 트리거인 것을 인식하고 그다음 명령을 실행할 수 있습니다.

01 플레이어 충돌 원리 설명

적에 닿았을 때 판정 처리



Collider 함수에서 isTrigger가 체크 되어 있는 동그란 물체가 사각형 물체의 인식 범위에 닿았을때 OnTriggerEnter()함수를 사용해 다음 명령어를 실행하는 방식으로 충돌 처리를 해줍니다.

이러한 방식을 이용하면 총알이 플레이어에 닿았을 때 죽는 처리, 플레이어가 바닥에 닿았을 때 점프 쿨타임을 초기화해 주는 처리 등 여러 코드를 쉽게 짤 수 있습니다.
이 게임에서는 캐릭터가 공중에서 적으로 떨어지고 적에게 닿았을 때 적을 파괴해 주는 코드에서 이 방식을 사용합니다.

Chapter 6

마무리

01 게임 시스템 구현

게임 시스템 구현

```
if (transform.position.y < -1)
{
    panel.SetActive(true);
}
if(transform.position.y > 7.5)
{
    d.text = "GameClear";
    d.color = Color.green;
    panel.SetActive(true);
    panel2.SetActive(false);
}
if (Input.GetKey(KeyCode.R))
{
    panel .SetActive(false);
    transform.position = new Vector3(1.19f, 0.8f, -0.85f);
}
```

만약 플레이어가 떨어졌을 경우에 게임오버 창을 띄어줍니다.
그리고 플레이어가 적이 있는 위치에 닿았을 경우에 게임클리어 화면을 띄어줍니다.
게임오버창일 경우에 R을 누르면 재시작할수있게 됩니다.

01 게임 시스템 구현

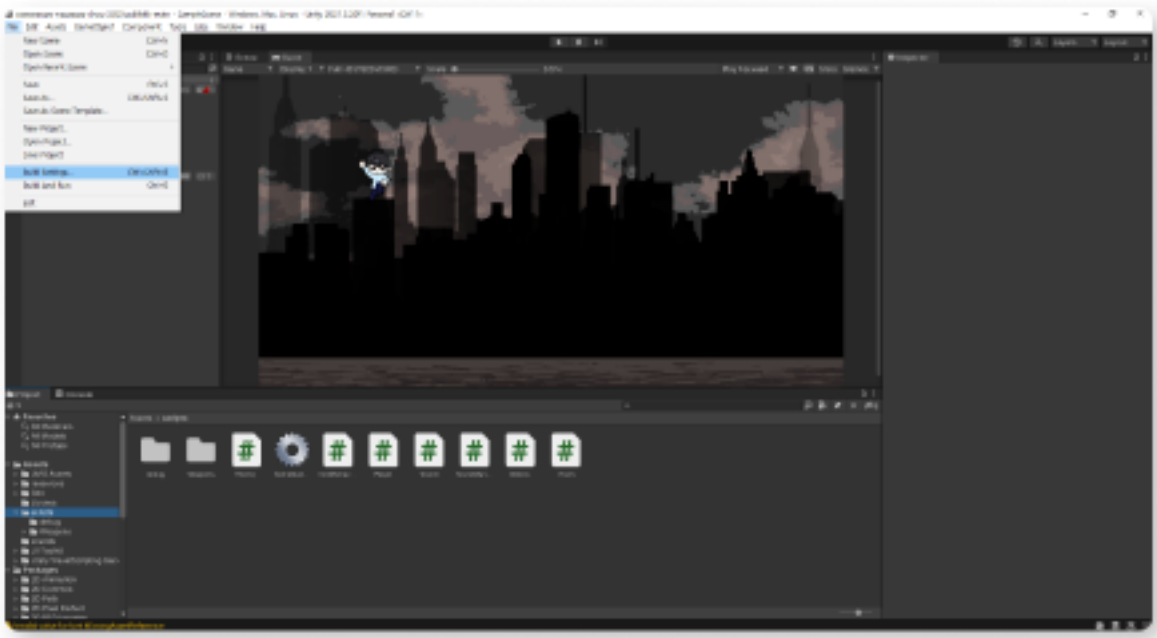
게임 시스템 구현

```
if (Input.GetKey(KeyCode.Keypad5))
{
    if (time > 3f)
    {
        Instantiate(go, transform.position - new Vector3(0, 0, 2), Quaternion.identity);
        time = 0.0f;
    }
}
```

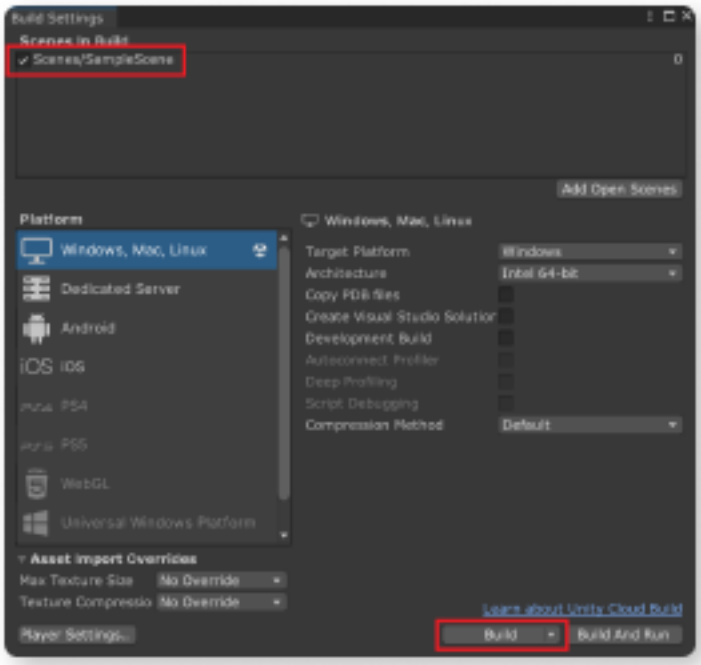
스페이스바를 눌러서 적이 돌을 생성할 수 있게 됩니다.
돌은 플레이어보다 질량이 100배 높아 위에서 돌이 내려오면 플레이어가 밀릴수밖에 없는 구조입니다. 그리고 돌을 생성하는데 3초 쿨타임을 주게 됩니다.
그리고 돌은 적의 좌표보다 z축이 2적은 위치에서 생성되게 됩니다.

01 프로젝트 빌드 방법

프로젝트 빌드 방법



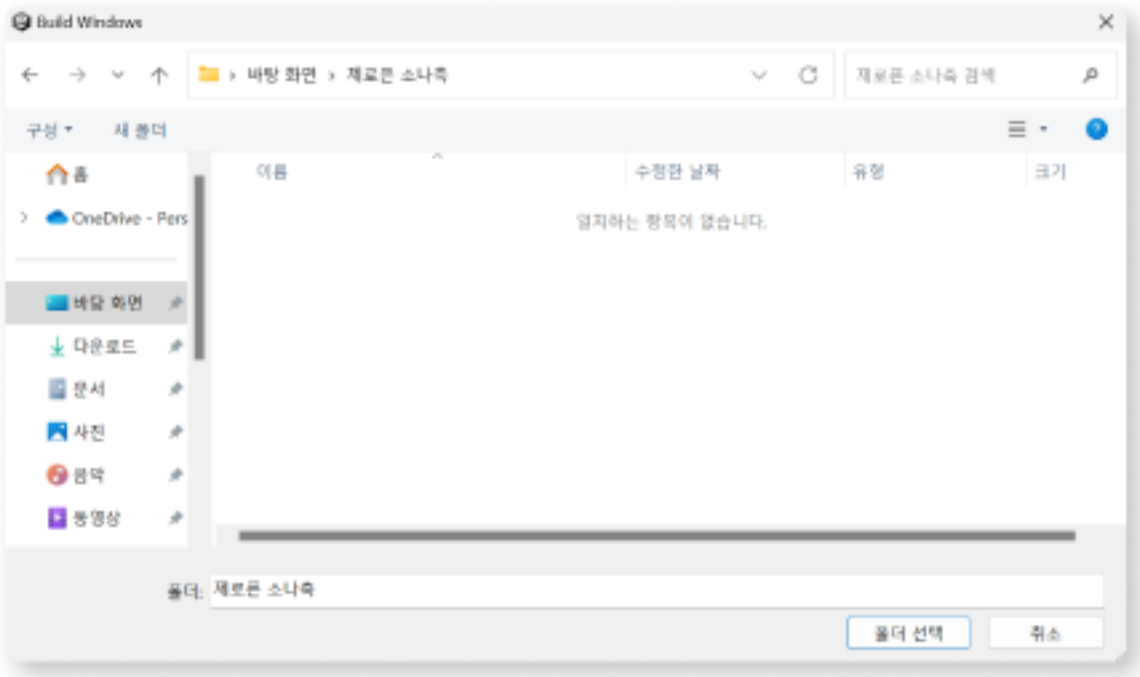
Ctrl + Shift + B 키를 눌러서 'Build Settings' 창을 열어줍니다.



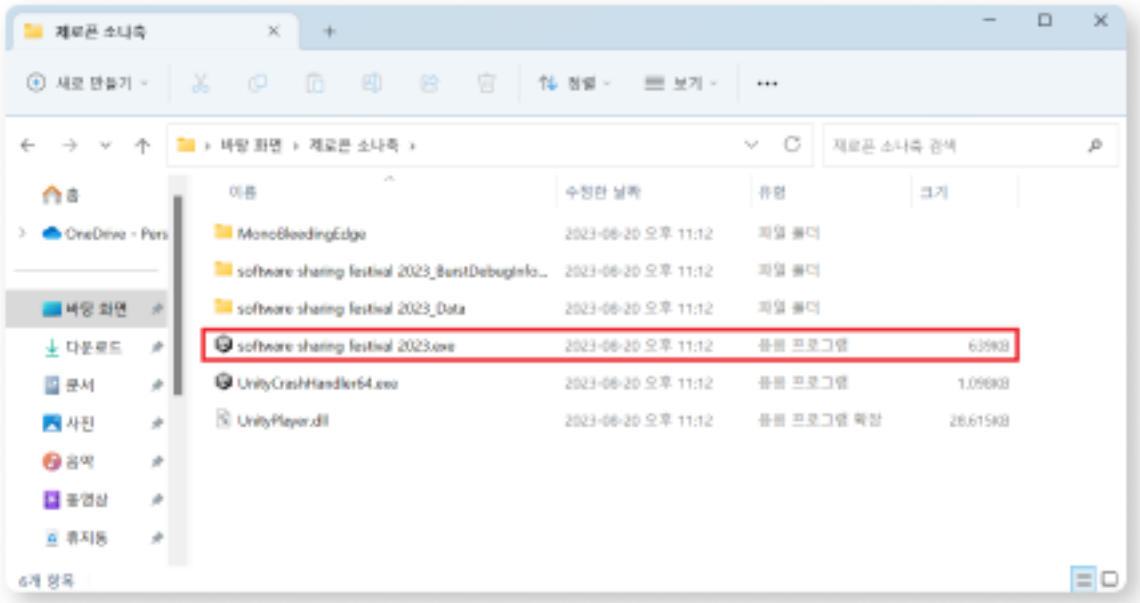
'Build Settings' 창에서 SampleScene를 체크해주고 [Build] 버튼을 눌러줍니다.

01 프로젝트 빌드 방법

프로젝트 빌드 방법



빌드할 프로그램을 저장할 폴더를 지정해 줍니다.



지정해준 폴더에서 빌드한 프로그램을 찾습니다. 이때 확장자는 .exe파일로 저장됩니다. .exe파일을 두 번 클릭하면 프로그램을 실행시킬수 있습니다.

디자이너 : 윤지연 조연진 차윤슬

개발자 : 김범근 김성태 이강민 이정훈 윤찬호 송혜인 최주성 황태건