Unity_Intro

서울 5반 이민아

Index

Unity Install

1. Unity Hub

> `https://unity3d.com/kr/get-unity/download`

- 유니티 허브 다운로드
- 여러 가지의 버전의 유니티들을 정리해서 관리해주는 매니저인 버전 관리 플랫폼
- 스마트폰에서 빌드 할 경우 IOS와 Android 구분 (Android Build Support를 체크)
- 로그인 후 라이센스 발급

2. Unity Project

- 버전 추가 후 프로젝트 생성
- 템플릿을 3D로 설정
- 프로젝트 경로 및 이름은 띄어쓰기 불가하고 영문만 가능 (UnityPractice)

Unity Menu

1. Menu

(1) File

- file 씬을 나타내는 메뉴
- build PC 버전(exe 실행), 안드로이드, iOS 등 빌드 가능
- 오큘러스 퀘스트 2 안드로이드 기반으로 빌드 Run Device 에서 Switch platform 선택
- 플레이어세팅 Company Name = 프로젝트 명

(2) Edit

- 에디터 플레이모드
    - 실행 (Play 버튼 클릭하여 플레이모드로 전환)
    - 중지 (Pause, 일시정지)
    - 단계별 진행 (Step, 한번에 한프레임씩 진행)
- Play mode 에서 생성한 게임 오브젝트는 Play mode 종료 시 삭제 됨
- Scene의 중요한 변경 사항은 Play mode를 해제한 상태에서 추가할 것
- Main Camera, Directional Light 기본 값
- 우클릭 오브젝트 추가 가능
- 3개의 화살표로 오브젝트 이동 가능

2. Transform Tools

- 오브젝트들을 움직이거나 회전하기 위해 최상단 Tool Box
- **단축키: q w e r t y**

- Hand Tool
    - 씬 화면 이동 가능
    - 게임 화면은 움직이지 않음 카메라는 그대로 있기 때문
- Move Tool (평행이동 툴)
    - 평행이동 툴의 화살표를 드래그하면 해당 방향으로 오브젝트를 평행이동 함
    - X축-빨간색, Y축-초록색, Z축-파란색
- Rotate Tool (회전 툴)
    - 회전 축을 드래그하여 축 기준으로 **회전**
- Scale Tool
    - 오브젝트의 **크기**를 조절 가능 (중앙 축을 드래그하면 XY 동시에)
- Rect Tool
    - 선택한 오브젝트의 직사각형을 편집(가로, 세로)
    - 2D 게임 오브젝트나 UI 게임 오브젝트를 주로 편집할 때 사용
    - 스케일과 크기, 포지션 값을 같이 조절
- Move, Rotate or Scale selected objects (Transform Tool)
    - 3가지 툴을 합친 것
- Available Custom Editor Tools
    - 사용자 정의 툴

3. View

(1) Inspector View

- 게임 오브젝트들의 기능들을 관찰하는 창
- 게임 오브젝트 클릭하면 우측의 Inspector창
- 모든 게임 오브젝트는 `Transform`이라는 기능을 가지고 있다
  - 최소한 자신의 World 상의 위치 또는 자신의 크기를 정보 정도
  - `Transform`이라는 기능은 위치, 회전, 크기와 관련된 것을 조정
- `Inspector > Add Component > Physics > Rigidbody` 컴포넌트 추가시 중력 추가 가능

(2) Project View

- Project창은 유니티 전용 파일 탐색기
- Asset(게임을 위해 쓸 파일) 확인 가능
- 파일들의 확장자가 보이지 않음(image.jpg파일은 image)

(3) Console View

- 프로젝트 옆에 탭을 눌러서 접속
- 디버깅할 때 많이 사용

(4) Hierarchy View

- Scene 안의 모든 게임 오브젝트들을 볼 수 있는 창
- 게임 오브젝트들의 부모-자식 관계 확인
- `Hierarchy > Create > 3D > Cube` 게임 오브젝트 생성

(5) Scene View

- 구성하고 있는 장면을 보여주는 창
- 조작을 하고 게임 오브젝트들을 배치
- Scene파일의 확장자는 `.unity`
- 조작모드
  - Q, W, E, R, T, Y로 모드 전환
  - Scroll
  - 우측 마우스 누르고 후 드래그
  - 좌측 마우스 누르고 후 드래그
  - 게임 오브젝트 누르고 F
  - Shift + Arrow
  - Ctrl + Arrow

(6) Game View

- 게임을 실제 실행했을 때 보이는 창
- Scene에서 카메라가 비추고 있는 장면

Unity Words

1. Game Objects

(1) 정의

- 게임 오브젝트는 여러 가지 기능들을 구성요소로 하는 `Box`
- 부착된 컴포넌트들에 의해서 동작

(2) 구성

- 3D Object , Effects, Lights 등의 효과 설정 가능
- 빛의 범위를 바꾸거나 빛의 색, 강도, 멀티플라이어 등을 조절

(4) 카메라

- 카메라를 한 씬에 두개 이상 두면 플레이 도중에 전환 가능 (Tag 기능에서 MainCamera 변경)
- 카메라가 오브젝트를 계속 따라오도록 설정할 가능 (오브젝트를 드래그해서 다른 오브젝트에 넣어 부모 자식 관계)
- Inspector > Mover(Script)

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

public class Mover : MonoBehaviour
{

    float m_fSpeed = 5.0f;

    void Update()
    {
        float fHorizontal = Input.GetAxis("Horizontal");
        float fVertical = Input.GetAxis("Vertical");

        transform.Translate(Vector3.right * Time.deltaTime * m_fSpeed *
fHorizontal, Space.World);
      transform.Translate(Vector3.forward * Time.deltaTime * m_fSpeed *
fVertical, Space.World);
    }
}
```

- 메인 카메라(Main Camera)에 오브젝트를 따라가는 스크립트를 추가

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

public class CFollowCamera : MonoBehaviour
{
    public Transform target;        // 따라다닐 타겟 오브젝트의 Transform

    private Transform tr;               // 카메라 자신의 Transform

    void Start()
    {
        tr = GetComponent<Transform>();
```

```
    }

    void LateUpdate()
    {
        tr.position = new Vector3(target.position.x - 0.52f, tr.position.y,
target.position.z - 6.56f);

        tr.LookAt(target);
    }
}
```

- Inspector > C Follow Camera(Script) > Target > 오브젝트 (Cube(Transform))

2. Component

(1) 정의

- 게임 오브젝트의 구성요소이며 각각의 기능들을 유니티에서는 `Component`
- 여러 가지의 값(변수)들로 구성

(2) 종류

- Collider 게임 오브젝트가 충돌할 수 있게 하는
- Camera 제가 게임을 실행했을 때의 볼 창
- Sound Listener 사운드가 들리는 곳을 설정
- Transform 위치, 회전, 크기, 등 (회전이나 크기를 표현하기 위해 X, Y, Z를 표현하기 위한 값(변수))
- Rigidbody 물리적인 상호작용
- Script 내가 정의하는 기능

3. Script

(1) 정의

- Component의 설계 도면
- 유니티에서 제공해주는 기본 Component가 아니라 우리가 만들어서 쓰는 것
- 키보드나 조이스틱, 등의 입력에 반응할 수 있고 특정 시간에 이벤트를 발생

(2) 메서드

- `Start()`와 `Update()` 메서드가 자동으로 생성

4. 변수

(1) 자료형

- int / float
- string / char (문자 하나만 저장하며 작은 따옴표 사용)

```
string my_string = "String"
char my_char = "C"
```

- bool

(2) Unity 변수

- 오브젝트 GameObject obect
- 위치정보 Tranform transform
- 물리엔진 정보 Rigidbody rigidbody
- 충돌체 정보 Collider collider

(3) 작성법

- 영문 숫자 혼합 가능
- 언더바 사용 가능
- 중복 변수명 불가
- 숫자 먼저 불가
- 공백 불가
- 특수 문자 불가(#)

Visual Studio Script

> Input에 반응하고 Component들의 값을 조정하며 원하는 게임

1. method

(1) Start() method

- 유니티가 자동으로 호출해주는 메서드 중 하나
- 게임이 시작할 때 딱 한 번만 호출
- `Update()` 메서드가 호출되기 직전에 호출
- 주로, 초기화와 같은 작업을 할 때

(2) Update() method

- 유니티가 자동으로 호출해주는 메서드 중 하나
- 매 프레임마다 호출
- FPS(Frame Per Second) (60 FPS이면 1초에 60번 `Update` 메서드가 호출)

(3) console

```
void Start()
{
    Debug.Log("Start() 가장 먼저 한 번 출력")
}

void Update()
{
    Debug.Log("Update() 프레임마다 매번 출력")
}
```

(4) Time.deltaTime

- 이전 프레임부터 현재 프레임이 일어나기까지 걸린 시간
- 컴퓨터 성능과 무관하게 게임이 작동하게 만들려면 필요
- 성능과 무관하게 프레임간의 간격에 비례하여 움직이게

```
void Update()
{
    // 내 캐릭터를 컴퓨터 성능과 무관하게 움직여라
    // 1초마다 일정한 간격(아래의 코드에서는 1미터)으로 움직여라
    transform.Translate(Vector3.right * Time.deltaTime);
}
```

(5) 변수 초기화

- Public이라면 인스펙터창에서 초기화
- (Asset)Project창에 있는 파일들도 Drag and Drop

```
// Public이라면 인스펙터창에서 초기화
public GameObject myGameObject;
public Transform myTransfrom;
public Vector3 myVector;
public string myName;
public int hp;

void Update()
{
    // 초기화한 변수들을 출력
    Debug.Log(myGameObject.name);
    Debug.Log(myTransform);
    Debug.Log(myVector);
    Debug.Log(myName);
    Debug.Log(hp);
    // 내가 가져온 오브젝트를 1유니티미터씩 오른쪽으로 움직여라
    myTransfrom.Translate(Vector3.right);
}
```

2. Transform (Component)

(1) 정의

- 모든 오브젝트가 가지는 특성
- 위치, 회전 그리고 스케일 저장 및 조작 (위치, 회전, 크기)
- Transform 컴포넌트는 유니티에서 자동으로 transform이라는 변수에 가져와준다

(2) Translate

- `Translate(백터)`
- 매 프레임마다 벡터만큼 위치 움직임
- 상대 좌표 사용
- 월드 기준으로 이동시키려면 2번째 인자로 Space.World

3. Rigidbody (Component)

(1) 정의

- 오브젝트가 물리 제어로 동작
- 중력을 받거나 속도를 부여하거나 물체끼리 충돌
- rigidbody는 직접 컴포넌트를 불러와야 한다

(2) 구성

- Mass 질량
- Drag 공기저항
- Angular Drag 토크로 회전할 때 공기 저항이 영향을 미치는 정도
- Use Gravity 중력 작용 여부
- Is Kinematic 물리 엔진으로 제어되지 않고 오로지 Transform으로만 조작

(2) Velocity

- `GetComponent<Rigidbody>().Velocity = 백터;`
- transform.Translate함수와 달리 속도는 한 번만 바꾸면 되므로 `Start()`

(3) Force

- `GetComponent<Rigidbody>().AddForce(백터);`
- 매 프레임마다 Force를 가해서 가속도가 생긴다 `Update()`

4. Find

(1) Find & Tag

- 오브젝트를 찾는 함수는 Update()와 같이 프레임마다 호출하는 것은 좋지 않다
- Tag를 사용하는 편이 속도가 더 빠르기 때문에 쓸 수 있는 상황에서 `GameObject.FindWithTag`
- 자주 활성, 비활성을 작업을 해주는 오브젝트는 활성화된 부모 오브젝트를 두어 계속 접근 `Transform.Find`

(2) GameObject Class Method

함수 이름	설명
Find	오브젝트 이름으로 검색하여 가장 처음에 나오는 오브젝트를 GameObject로 반환한다.
FindWIthTag	태그 이름으로 검색해서 나타난 오브젝트 여러개를 GameObject 배열로 반환한다.
FindGameObjectsWithTag	태그 이름으로 검색해서 나타난 오브젝트 여러개를 GameObject 배열로 반환한다. 같은 태그를 가진 Object들을 GameObject[] 형태로 반환한다.
FindObjectOfType	오브젝트형(혹은 컴포넌트의 형)으로 검색해서 가장 처음 나타난 오브젝트를 GameObject로 반환한다.(유효한 오브젝트만)
FindObjectsOfType	오브젝트형(혹은 컴포넌트의 형)으로 검색해서 가장 처음 나타난 오브젝트 여러개를 GameObject 배열로 반환한다.(유효한 오브젝트만)

```
GameObject obj1 = GameObject.Find("Name");
// Object의 이름으로 대상을 찾음, 이름이 같을 경우 가장 처음 검색된 Object 반환
GameObject obj2 = GameObject.FindWithTag("Tag");
// 태그로 대상을 찾음, 이름이 같을 경우 가장 처음 검색된 Object 반환
GameObject obj3 = GameObject.FindObjectsWithTag("Tag");
// 같은 태그를 가진 Object들을 GameObject[] 형태로 반환
```

(3) Transform Class Method

함수 이름	설명
Find	Object의 이름으로 자식 오브젝트를 검색해, 가장 처음에 나타난 자식 오브젝트를 반환한다.
GetComponentInChildren	컴포넌트 형으로 자식 오브젝트를 검색해서 처음 나타난 자식 오브젝트를 반환한다.
GetComponentsInChildren	컴포넌트 형으로 자식 오브젝트를 검색해서 나타난 자식 오브젝트들의 배열을 반환한다.
GetComponentInParent	컴포넌트 형으로 부모 오브젝트를 검색해, 가장 처음에 나타난 부모 오브젝으를 반환한다.
GetComponentsInParent	컴포넌트 형으로 부모오브젝트를 검색해서 나타난 부모 오브젝트들의 배열을 반환한다.
FindObjectOfType	오브젝트형(혹은 컴포넌트의 형)으로 검색해서 가장 처음 나타난 오브젝트를 반환한다.(유효한 오브젝트만)
Transform.FindObjectsOfType	오브젝트형(혹은 컴포넌트의 형)으로 검색해서 나타난 여러개의 Object들을 배열의 형태로 반환한다.(유효한 오브젝트만)

Unity Asset Store

1. Asset Store

> `https://assetstore.unity.com/?locale=ko-KR`

(1) Asset Store

- `Windows > Asset Store`
- Ctrl + 9

(2) Import

- download 는 내 PC 에 내려받는 기능
- import 는 현재 열어놓은 유니티 프로젝트의 에셋에 추가하는 기능
- `Windows > Package Manager`
- 한번 다운로드한 패키지는 패키지 매니저에 이미 보관되어 있기 때문에 아래 메뉴를 선택하시면 에셋 스토어에 접근하지 않고 바로 불러오기

(3) Object

- fbx(애니메이션 확장자) 또는 프리셋에서 모델링을 Drag & Drop
- Hirarchy view 확인
- Create Empty 생성 후 오브젝트들을 자식 오브젝트로 넣어주면 한 번에 관리 및 이동 가능
- 오브젝트 더블클릭 오브젝트로 가까이 다가감

2. Workflow

> Import, Create, Build, Distribute, Load

(1) Import

- Asset Store에서 Import

(2) Create

- Scenes의 GameObjects를 script 사용해서 추가

(3) Build

- `.EXE` file 생성

Unity XR

1.Package Manager

- 대부분의 Unity XR 기능은 패키지에서 이용
- `Window > Package Manager`를 통해 설치
- Package Manager 창에는 각 패키지에 대한 상세 정보