버전 : 2021.3.34f1

hub 버전 : 3.7.0

1. 유니티 설명

2. 오브젝트 제어

**프로젝트 생성**

New project -> 2D Core 선택 -> 오른쪽에서 프로젝트 이름, 저장경로 설정 -> 눌러서 실행

**창 설명**

창 이름은 각 창 위쪽에 적혀있는 이름이 있음

유니티는 보통 Scene 단위로 제작이 이루어지는데 특정 Object 요소를 저장해놓고 쓸수도 있음

Scene View : 현재 Scene의 게임오브젝트가 표시되고 배치, 크기조절 등 보통 이걸 보면서 제작함

Game View : 실제 게임을 실행할때 카메라를 통해 보이는 모습

Hierarchy View : '하이라키 뷰' 라고 읽고, 해당 Scene에 있는 모든 오브젝트가 표시됨

Project View : 오브젝트, 텍스처, 코드, 음악 등 게임에 사용될 모든 요소가 표시됨, 하나의 패키지라고 생각하면 됨

Inspector View : 오브젝트에 대한 구성이나 설정, 스크립트 등 설정 가능

Console View : 프로그램 메시지, 오류, 경고 등 컴파일애 대한 요소가 표시됨

창에 대한 추가는 상단의 툴바에서 Window -> General에 있음

창 이름을 잡고 드래그해서 원하는 곳에 놓으면 됨

**기본 세팅**

Project view -> Assets 클릭 -> 마우스 우클릭 -> Create -> Folder

기본적으로 Scenes, Prefabs, Scripts, Textures(images, Resources 등 디자인 요소)로 구성되고 Sounds, Animations 등 피룡한 것은 그때마다 추가해서 사용

**게임오브젝트 생성**

Hierarchy view -> 마우스 우클릭 -> 2D Object -> sprites -> Square

생성된 Square를 선택하면 Scene view와 Inspector view에 해당 object에 대한 정보가 나옴

**컴포넌트 추가**

기본 Component 설명

Transform - 오브젝트의 위치, 회전, 크기를 수치로 조정 가능

Renderer - 오브젝트 모양에 대해 렌더링을 해줌, 옵션중 sprite로 물체의 모양을 변경함

Meterial - 색 또는 질감 조절을 해줌

Inspector view -> Add Component 클릭 -> Rigidbody 2D 추가

Rigidbody - GameObject가 물리 제어로 동작하게 함

**게임오브젝트 제어**

1. Scene view

왼쪽에 컨트롤 Tool 들이 있는데 화면 이동, 오브젝트 이동, 오브젝트 회전, 오브젝트 크기조절등의 작업을 할 수 있음

2. Inspector view

Transform을 직접 조절 가능(값 조절, x y z 글자 클릭&드래그로 조절 가능)

3. Script

c# 코드를 이용해 조절

**Script 생성, 적용**

Scripts 폴더 -> 마우스 우클릭 -> Create -> C# Script

이름은 ObjectControl로 작성

스크립트 적용 방법

1. Hierachy view에서 game Object 클릭 -> Inspector view의 Add Component 클릭 -> Script 이름 검색

2. Hierachy view에서 game Object 클릭 -> Project view에서 Scripts 클릭 -> 적용할 스크립트를 Project view에서 Inspector View로 드래그&드롭

파일 명을 바꾸면 파일명과 class명을 동일하게 해야됨

**Script 설명**

기본 상속되는 MonoBehaviour 클래스가 있음

MonoBehaviour는 기본적인 게임 오브젝트 연결에 대한 프레임워크를 제공함

Start : 게임 오브젝트가 로드되면 호출함(한번)

Update 전에 한번만 실행된다고 보면 됨

오브젝트 초기 설정, 오브젝트 연결, 변수 설정 등의 작업 수행

Update : 스크립트가 활성화 상태일때 매 프레임마다 호출함(반복)

게임이 실행되는 동안 오브젝트를 계속 바꾸는 모든 작업 수행

다른 시점에 호출되는게 있는데 나머지는 최적화 설명할때 같이 설명

**물체 이동 코드**

보통 쓰이는 3가지 방식

1. transform.translate

2. transform.position

3. rigidbody.AddForce

4. rigidbody.velocity

translate는 방향을 입력받아서 로컬 좌표를 기준으로 해당 방향으로 가게 하는 방법

position은 Inspector View에서 볼수 있는 월드 좌표를 직접 조절해서 위치를 옮기는 방법

AddForce는 물리적인 힘을 가해서 물체를 이동시킴 -> 질량에 따른 속도 차이가 생기고 관성의 영향을 받게됨

(position에 가속도 변수를 추가해서 AddForce와 비슷한 동작을 하도록 만들수는 있음)

velocity는 rigidbody의 속력을 변경해서 이동

**Script 코드 설명**

변수 선언할 때 public, private를 지정 가능

public으로 선언하면 Inspector view에서도 변수 확인, 값 조절 가능

변수로 선언 가능한 자료형은 int, float같이 기본적인 것과 GameObject, 오브젝트 요소 같은 것도 가능함

Vector2.up - 월드 좌표에서 위쪽 방향(1,0)을 나타냄, up, down, right, left가 가능하고 3차원 공간을 나타내는 Vector3에서는 forward, back까지 동작이 가능함

transform.up 같은 것도 가능한데 이건 물체의 회전에 따라 방향이 달라짐 -> 보통 3D에서 많이 사용됨

Time.deltaTime - 프레임 사이의 간격을 float형식으로 반환함 -> 오브젝트가 불규칙한 Update 호출에서 일정한 속도로 이동할수 있게 해줌

AddForce - (Vector3, mode)를 파라미터로 입력받는데 필요한 효과에 따라 여러가지 모드로 설정 가능

rigidbody도 velocity로 속도 조절을 값으로 직접할 수 있음

직접 값을 조절하는 방식은 대각선으로 갈 경우 속도가 빨라지게 됨 -> rigidbody.velocity는 피타고라스 정의에 따라서 값이 조정되기 때문에 안정적으로 대각선 이동이 가능함

**물체 회전**

회전을 설정하는 두가지 방식이 있음

1. Euler angle(오일러 각)

2. Quaternion(쿼터니언)

오일러 각은 x, y, z 축을 기점으로 절대적인 값의 회전을 나타내고

쿼터니언 각은 한 방향에서 다른 방향으로의 상대 회전을 나타냄

유니티에서 모든 rotation은 Quaternion이 사용돼서 우리가 생각하는 x, y, z축 형식인 오일러 각을 쿼터니언 값으로 변경해야 됨

transform.Rotate은 연속적으로 회전

transform.rotation은 지정된 값을 정적으로 회전

**혼자 해볼내용**

1. 다른 키 입력으로 물체 제어

2. 특정 공간을 물체가 벗어나지 못하도록 제어