Unity 메타버스 과정

2022 Unity 3.14f 기준

CONTENTS >

M1. Unity 이해하기

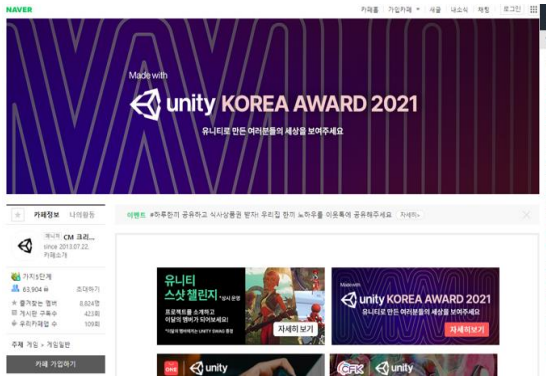
✓ 이 워크북은 Unity MR Education용으로만 사용 가능합니다.
✓ Unity는 게임, 영상, XR, AI등의 제작 플랫폼입니다.
✓ Unity는 초, 중, 고 교육용은 무료입니다.
✓ Unity의 실행을 위해서는 반드시 관련 계정이 있어야합니다.
✓ 자세한 정보를 얻기 위해서는 다음의 사이트에 접속합니다.
✓ https://unity.com/

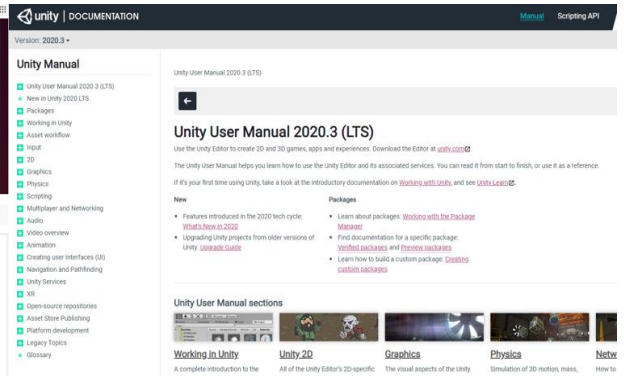
✓ 2004년 8월 Unity Technologies가 개발한 게임 엔진입니다.
✓ Unity Technologies는 2004년 덴마크 코펜하겐에서 설립되었습니다. 현재는 미국 샌프란시스코로 본사를 이전했습니다.
✓ 주로 저 사양/소규모 게임의 개발에 적합하며, 2005년 6월 8일에 처음 발표되었습니다.
✓ Unity의 시작은 플래시로 구현이 힘든 3D 시장 공략을 노린 3D 타겟 웹미디어 제작 툴 이였습니다.
✓ 현재까지 'Unity Web Player'라는 이름으로 남아있습니다.
✓ 이후 타 엔진에 비해 사용법이 쉽다는 점에 착안하여 개발자들이 이를 이용해 게임엔진으로 방향을 선회하였습니다.
✓ 여타 고급엔진들로 대형 프로젝트를 개발하는 것에 비해 단순한 게임을 만들어 내기에는 비교적 쉬운 편이라 모바일시대로 들어오면서 승승장구 하였습니다.
✓ Unity는 Asset Store라는 생태계의 조성에 힘입어 넓은 사용자 풀이 형성되면서 모바일 게임 시장의 1위 제작 플랫폼으로 자리 잡고 있습니다.

☺ **과정 목표**

Unity에서 기본적으로 제공하는 정보공유, 학습에 참고 될 항목을 확인합니다.

✓ **1. Unity 공식사이트**





- Unity 공식 카페입니다.
- 최신 업데이트 정보를 확인 할 수 있습니다.
- https://unity.com/

✓ **2. Unity Document**

- Unity Version별로 제공됩니다.
- 최신 업데이트 된 내용도 포함되어 있습니다.
- 한글 번역은 순차적으로 진행이 되고 있습니다.
- 개발할 때 1순위로 참고해야 되는 사이트입니다.
- https://docs.unity3d.com/Manual/index.html

✓ **Unity 공식 카페**

- Unity Korea가 직접 운영하는 Naver카페입니다.
- 만명의 개발자들이 자유롭게 토론하고 정보를 공유함에 있어서 유익하고자 만들어진 카페입니다.
- 각종 교육정보 및 업데이트 소식, Q&A 게시판 등이 있습니다.
- https://cafe.naver.com/unityhub

M2. Unity 설치하기

☺ **과정 목표**

Unity와 다운받고 설치하는 과정을 살펴보고, 설치를 진행합니다.

✓ 1. Unity를 설치하기 위하여 다음사이트로 접속합니다.

https://unity.com/은 Unity 공식 사이트입니다.

✓ 2. 다음 사이트에서 Unity를 설치합니다.

https://unity.com/download에서 Download Unity Hub를 클릭합니다.

✓ UnityHubSetup.exe를 실행하여 UnityHub를 인스톨합니다.

☺ **과정 목표**

Unity Hub에 대한 기능과 역할을 탐구합니다.

✓ Unity Hub의 구성은 다음과 같습니다.

✓ A. 프로젝트

-프로젝트를 생성 관리를 할 수 있습니다.

✓ B. 학습

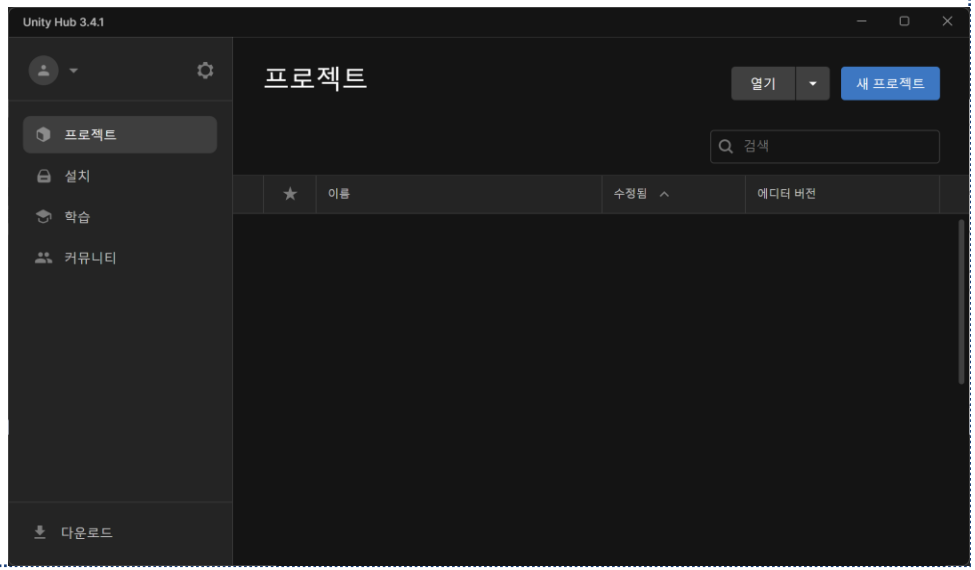
-학습 예제프로젝트와 튜토리얼을 학습할 수 있습니다.

✓ C. 커뮤니티

-Unity에서 공식적으로 제공하는 블로그, 포럼 등을 제공합니다.

✓ D. 설치

-Unity Version별로 관리 하는 공간입니다.



☺ **과정 목표**

Unity Hub의 Setting설정에 대한 기능과 역할을 탐구합니다.

✓ Unity Hub Setting의 구성은 다음과 같습니다.

✓ A. 일반

-Unity editor의 저장 위치 및 언어를 설정할 수 있습니다.

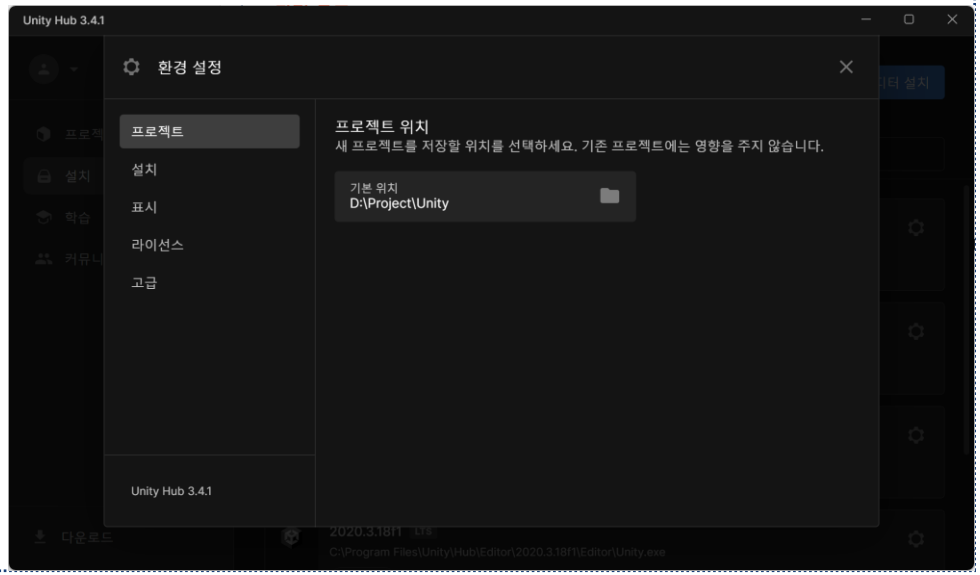
✓ B. 라이선스 관리

-라이선스를 관리하는 창입니다.

-라이선스를 활성화하기 위하여 로그인 하고 새 라이선스 활성화를 꼭 클릭 후 활성화 해야 Unity를 사용 가능합니다.

✓ C. 고급

-정식, 베타 버전을 선택할 수 있습니다.



☺ **과정 목표**

Unity Version 선택하고 설치함을 탐구합니다.

✓ Unity Hub 버전 추가의 구성은 다음과 같습니다.

✓ A. 권장 릴리스

-Unity에서 권장 하는 릴리스 버전입니다.

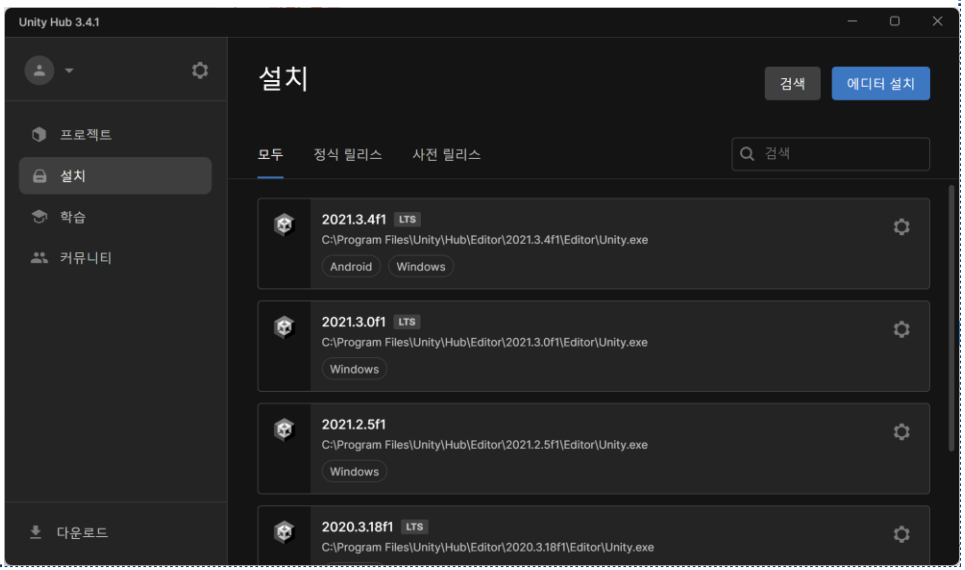
✓ B. 정식 릴리스

-Unity에서 출시한 정식 릴리스 버전입니다.

✓ C. 사전 릴리스

-최신 버전의 Unity editor(알파 버전)입니다.

✓※버전은 개발에 필수적인 기능을 포함한 버전으로 선택하면 됩니다.



☺ **과정 목표**

Unity 설치에 필요한 Unity 모듈의 종류와 기능을 탐구합니다.

✓ Unity Hub 버전 추가의 구성은 다음과 같습니다.

✓ A. Dev tool

-개발에 필요한 Visual Studio를 설치할 수 있습니다.

-이미 설치되어 있다면 굳이 설치할 필요는 없습니다.

✓ B. Platforms

-원하는 build Platform의 support를 설치할 수 있습니다.

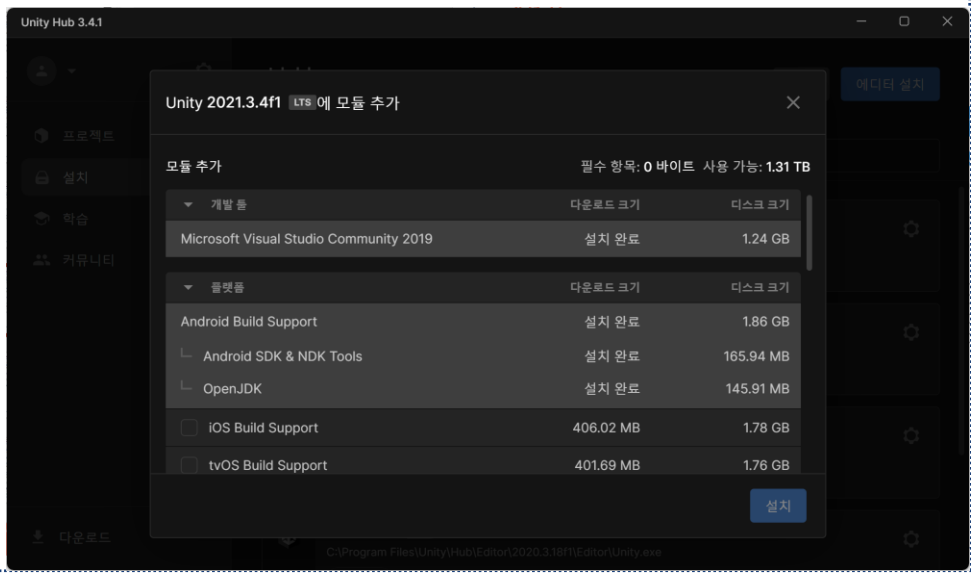
✓ C. Documentation

-Unity Manual을 설치할 수 있습니다.

✓ D. Language Packs(Preview)

-언어 팩을 설치할 수 있습니다.(한국어 지원)

✓※게임 개발을 위해서는 Visual Studio를
꼭 설치해 주십시오.



☺ 과정 목표

프로젝트를 생성하는 과정을 탐구합니다.

✓ Unity Project 추가 방법은 다음과 같습니다.

✓ A. 추가

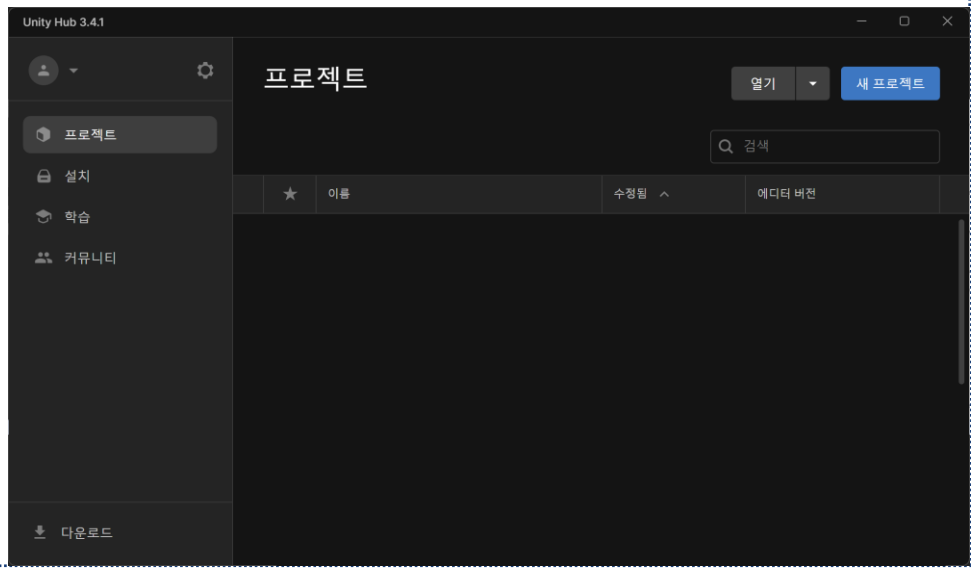
-기존 프로젝트를 Hub에 추가합니다.

✓ B. 새로 생성

-프로젝트를 새로 생성합니다.

✓ C. ▼

-새로 생성할 프로젝트의 버전을 선택할 수 있습니다.



☺ **과정 목표**

프로젝트를 생성하는 과정을 탐구합니다.

✓ Unity Project 추가 시 기본적으로 제공되는 템플릿은 다음과 같습니다.

✓ A. 2D

-2D프로젝트에 필요한 기능을 Default 템플릿으로 제공합니다.

✓ B. 3D

-3D프로젝트에 필요한 기능을 Default 템플릿으로 제공합니다.

✓ C. High Definition RP

-HDRP 파이프라인을 사용하는 프로젝트에 필요한 기능을 Default 템플릿으로 제공합니다.

✓ D. Universal Render Pipe Line

-URP 파이프라인을 사용하는 프로젝트에 필요한 기능을 Default 템플릿으로 제공합니다.

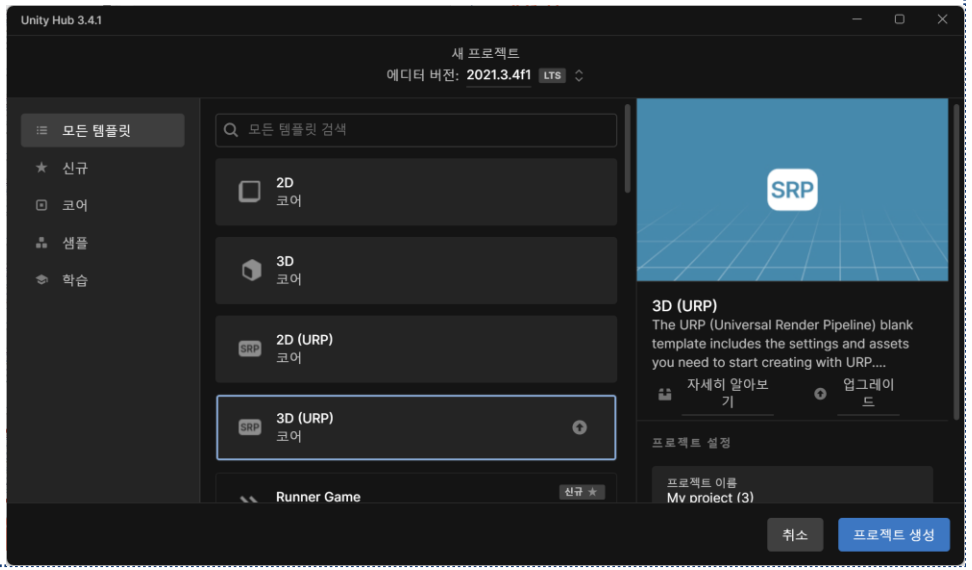
✓ E. 프로젝트 이름

-프로젝트 이름을 설정할 수 있습니다.

✓ F. 저장 위치

-프로젝트 저장 위치를 설정할 수 있습니다.

✓ ※ Unity Version에 따라 제공되는 템플릿이 추가되어 있으니 Unity공식 사이트를 참고 하세요.



M3. 메타버스 알아보기

☺ **과정 목표**

메타버스에 대한 정의에 대해 알아보고 탐구합니다.

✓메타버스의 대한 정의

가상 공간을 나타내는 메타버스는 1992년 닐 스티븐 슨(Neal Stephenson)의 소설
《스노우 크래쉬》에서 처음 등장한 개념과 용어입니다.

이는 확장 가상세계라고 불리우며, 초월에 의미인 Meta 와 특별한 세계 ,우주의 의미인 Universe를
합성한 신조어입니다.

3차원에서 실제 생활과 법적으로 인정되는 활동직업 금융 학습 등이 연결 된
가상세계로 일컬으며, 폭넓게 나아가서는 정치나 경제 사회 문화의 전반적 측면,
또는 현실과 비현실이 공존하는 생활 또는 게임형 가상 세계라고도 불리우고 있습니다.

다만, 메타버스라는 개념의 아직까지 뚜렷하게 정의되지는 않았습니다.
일반적으로는 '현실세계와 같은 사회적·경제적 활동이 통용되는 3차원 가상공간'
정도의 의미로 사용되고 있으나, 학자나 기관마다 나름의 정의를 내리고 있어
넓은 의미로 통용되고 있습니다.

☺ **과정 목표**

메타버스에 대한 4가지의 유형에 대해 알아보고 탐구합니다.

✓메타버스는 크게 4가지 유형으로 각각 증강현실(AR), 일상기록(Life Logging), 거울세계(Mirror Worlds), 가상세계(VR)으로 정의하고 있습니다.

✓증강현실 (AR)

- 증강현실은 현실공간에 2D 또는 3D로 표현한 가상의 겹쳐 보이는 물체를 통해 상호작용 할 수 있는 환경을 의미하며, 일반적으로 사용자가 단말기 카메라로 촬영하면 디지털로 구축 된 과거의 건물이 사용자 단말기에 중첩해서 보여주는 장면이 증강현실의 사례입니다.

✓ 일상기록 (Life Logging)

- 사물과 사람에 대한 일상적인 경험과 정보를 캡처하고 저장하고 묘사하는 기술로 일상생활에서 일어나는 모든 순간을 사진, 텍스트, 영상 등으로 그 내용을 서버에 저장하여 다른 사용자들과 공유할 수 있는 기능입니다. 일반적으로 사용하는 Health Care, Instagram 등이 해당 사례에 포함됩니다.

☺ **과정 목표**

메타버스에 대한 4가지의 유형에 대해 알아보고 탐구합니다.

✓거울세계 (Mirror Worlds)

- 실제 세계를 가능한 한 사실적으로 반영하되, 정보적으로 확장된 가상세계를 일컫는 표현입니다.
  대표적으로 구글 어스(Google Earth)를 예로 들 수 있습니다.
  Google Earth는 세계 전역의 위성사진을 모조리 수집하여 일정 주기로 사진을 업데이트 하면서,
  변화하는 현실세계의 모습을 반영하고 있습니다.

✓가상세계 (Virtual Worlds)

- 가상세계는 현실과 유사하거나 혹은 완전히 다른 대안적 세계를 디지털로 구현한 세계를 일컫는 표현입니다.
  가상세계에서 사용자들은 아바타를 통해 현실세계의 경제적, 사회적인 활동이나, 유사한 활동을 할 수 있다는
  특징이 있습니다.
  가상세계는 우리에게 가장 친숙한 형태의 메타버스로서, 일반적으로 게임이나 세컨드 라이프와 같은
  생활형 가상세계를 칭하는 개념입니다.

☺ **과정 목표**

메타버스에 대한 문제점에 대해 알아보고 탐구합니다.

✓메타버스 내의 불법 행위와 사법권

- 메타버스 내 가상세계에서의 도박, 사기, 매춘 등 범죄가 발생하며 현실세계의 법질서를 가상세계에도 동일하게 적용하자는 견해가 나왔으나, 가상세계에서는 물리적 장소 개념을 적용하지 못하므로, 법적 문제로 이관 될 시 재판 관할에 문제가 발생하기에 현실세계의 법질서를 다루기 힘들다는 문제가 있습니다.

✓가상화폐의 현금화

- 가상세계의 경제 규모가 커지면서, 가상화폐의 현금화에 관한 논쟁이 발생 중으로, 국내에서는 게임산업진흥법에 의거하여 가상화폐 환전은 불법으로 취급하고 있기에, 문제가 발생 할 수 있습니다. 또한, 가상화폐를 새로운 거래수단으로 인정할지에 대한 문제로, 일부에서는 가상경제 활성화라는 긍정적인 효과를 기대하는 반면, 게임중독, 불법거래 등 탈세에 대한 우려로 인해 부정적인 측면도 있는 상황이다.

✓가상세계 중독

- 가상세계, 특히 현실과 사회경제적 활동 양상이 비슷한 메타버스의 경우, 기존 온라인 게임과 달리 일상생활로 인식하며 중독성 심화 가능성이 높은 편입니다. 가상세계에 지나친 몰입으로 현실 일상을 뒤로하고, 정체성 장애가 발생할 수 있는 문제가 있을 수 있습니다.

☺ **과정 목표**

Unity에서 정의한 메타버스 자료와 실제 메타버스 적용 사례에 대해 알아봅니다.

✓Unity에서 정의한 메타버스 자료

https://kr.object.ncloudstorage.com/ebook202110metaverse/UTK_All_ebook_Intro-to-the-Metaverse%20%281%29.pdf

✓메타버스 적용 사례

- 스페이셜 (Spatial) – VR을 이용한 가상 회의 (Unity 제작)

  참고 링크 : 스페이셜 영상 보기

- AltSpaceVR – 가상 회의 및 원격수업 행사 등(MS)

  참고 링크 : AltSpaceVR 영상 보기

- 이프랜드 – 모바일 플랫폼 소셜 커뮤니티 (국내SKT)

  참고 링크 : 이프랜드 영상 보기

- 게더타운 – 2D 메타버스 플랫폼 화상 회의 및 원격 수업

  참고 링크 : 게더타운 영상 보기

- 컴투버스 – 컴투스

  참고 링크 : 컴투스 영상 보기

M4. Object
Move & Rotation

☺ **과정 목표**

Object에 움직임을 주는 방법에 대해 알아보고 탐구합니다.

✓Unity Engine 안에서 Object 움직임을 주는 방법에 대해 알아봅니다.

1. Object를 움직이는 방법은 여러 방법이 있으나 그 중 간단하게 3가지에 대해 알아봅니다.

A. Transform.position 값을 조정하여 움직이는 방법.

- 현재 Object의 x,y,z좌표 값을 직접 조절하여 움직이는 방법이 있으며 일반적으로 사용 시,
  Transform.position += Vector3.forward * Time.deltaTime; 으로 움직임을 정합니다.

  위에 Vector3.forward는 실제 Vector3(0,0,1)과 같은 값으로 Unity상 정면 방향을 나타냅니다.
  단 실제 World 좌표를 이용하여 움직이기 때문에 캐릭터가 World상 Z축 방향을 바라보고 있지 않을 경우,
  캐릭터가 바라보는 방향으로 움직이지 않는 단점이 존재합니다.
- 실제 Key값을 받아와 설정하여 해당 키를 눌렀을 경우 방향을 설정하는 식으로 사용하나, 매 프레임마다
  각 축 값을 더하기에 약간 끊김 현상이 발생 할 수도 있습니다 (자연스러운 움직임이 나오지 않을 수 있음)

☺ **과정 목표**

Object에 움직임을 주는 방법에 대해 알아보고 탐구합니다.

B. Transform.translate() 메소드를 이용하는 방법.

- Transform.translate 메소드는 Local좌표 기준으로 이동해주는 메소드입니다.
  실제 사용 시 메소드 내 인자 값으로 이동 할 위치 값을 넣음으로써, 해당 값으로 이동 할 수 있습니다.
- 실제 사용 예)

  Transform.translate(Vector3.forward * Time.deltaTime);

  이 또한 앞에 설명한 position값 이동처럼 실제 Key값을 받아와 설정하여 해당 키를 눌렀을 경우 방향을 설정하는 식으로 사용하나, 각 축마다 해당 값을 입력해 줘야 하는 번거로움이 있습니다.
  또한 position 이동과 해당 메소드의 차이점은 position은 단순 위치 값을 변화하는 것인 반면, translate는 연속적인 이동이기에 position 이동과 달리 끊김 현상이 발생하지 않습니다.

☺ **과정 목표**

Object에 움직임을 주는 방법에 대해 알아보고 탐구합니다.

C. Rigidbody Component를 이용한 방법

- 실제 Object에 물리적인 기능을 추가하여 이동하는 방법입니다.
- Rigidbody Component를 Object에 추가 시 물리가 적용되기에 기본적으로 중력과 관성의 영향을 받습니다.
  그리하여 움직임을 주었을 때 Object는 가장 안정 된 상태를 찾아가게 되기에 Capsule Object의 경우에는 관성에 의해 넘어지는 것을 볼 수 있습니다.
  그리하여 해당 현상을 방지하기 위해 회전 축을 제한 시킨 후 이동을 진행 합니다.
- 축 제한 방법은 Rigidbody 속성의 Constraints 속성의 Rotation X축과 Z축을 Check하여 축을 제한합니다.
  (Y축은 Capsule의 좌우 회전을 담당하기에 따로 제한하지 않습니다)
- 이후 Script에서 Rigidbody 변수를 선언한 후 해당 변수를 찾아 초기화를 진행합니다.
  초기화 후 Rigidbody AddForce 메소드를 활용하여 Object를 이동 할 수 있습니다.
  다만, AddForce의 경우 물리적인 힘을 지속적으로 주어지기에 점점 관성에 의해 가속도가 붙는 현상이 발생하기에 velocity 프로퍼티에 Vector값을 직접 넣어주어 이동을 진행 할 수 있습니다.

☺ **과정 목표**

Object에 움직임을 주는 방법에 대해 알아보고 탐구합니다.

✓Unity Engine 안에서 Object 움직임을 주는 방법에 대해 알아봅니다.

2. 위 이동 방법을 이용한 실제 이동기능 구현해보기 (Transform.position)

A. Transform.positon 이동기능을 통해 실제 메타버스에서 캐릭터가 이동하도록 기능을 만들어 봅니다.

- Window – Asset Store을 클릭 후 [Search online] 버튼을 클릭하여 Asset Store에 접속합니다.
- Asset Store에서 [**Character Pack: Free Sample]** 을 검색 후 캐릭터를 다운받습니다.
- 다운로드 이후 Package Manager로 접속하여 다운받은 캐릭터를 Project로 Import 합니다.
- Project창에서 Assets\Supercyan Character Pack Free Sample\Prefabs\Base 경로 안에 있는 MaleFree1 캐릭터를 드래그 하여 Scene View에 배치합니다.
- Project창에서 마우스 우 클릭 후 Create – C# Script 항목을 선택하여 새로운 Script를 생성합니다.
- 생성 후 C# Script 이름을 MovePosition으로 변경합니다.
- MaleFree1 캐릭터를 선택 후 Inspector창에 Add Component 항목을 선택 한 뒤 MovePosition Script를 추가합니다.
- MovePosition Script를 열고 아래와 같이 이동 코드를 작성합니다.

☺ **과정 목표**

Object에 움직임을 주는 방법에 대해 알아보고 탐구합니다.

```
void Update()
  {
    if(Input.GetKey(KeyCode.W)) // Input Class로 W 키를 눌렀을 때 true값 반환
    {
       transform.position += Vector3.forward;  //Vector3.forward == Vector3(0,0,1)
    }
    if (Input.GetKey(KeyCode.S))  // Input Class로 S 키를 눌렀을 때 true값 반환
    {
       transform.position += Vector3.back;  //Vector3.back == Vector3(0,0,-1)
    }
    if (Input.GetKey(KeyCode.A))  // Input Class로 A 키를 눌렀을 때 true값 반환
    {
       transform.position += Vector3.left;  //Vector3.forward == Vector3(-1,0,0)
    }
```

☺ **과정 목표**

Object에 움직임을 주는 방법에 대해 알아보고 탐구합니다.

```
    if (Input.GetKey(KeyCode.D))  // Input Class로 D 키를 눌렀을 때 true값 반환
    {
       transform.position += Vector3.right;  //Vector3.forward == Vector3(1,0,0)
    }
  }
}
```

- 다만 현재 이 코드로 실행했을 경우 캐릭터가 엄청 빠르게 이동하는 것을 확인 할 수 있는데 이는 업데이트 안에서 실행하고 있기에 실제 매 프레임마다 1씩 이동하는 거와 같습니다. 그래서 이러한 현상을 방지하기 위해 Time.deltaTime (현재 프레임을 값을 나눠 매 초 마다 실행시키도록 하는 Time Class)를 곱해주도록 합니다.

☺ **과정 목표**

Object에 움직임을 주는 방법에 대해 알아보고 탐구합니다.

✓Unity Engine 안에서 Object 움직임을 주는 방법에 대해 알아봅니다.

2. 위 이동 방법을 이용한 실제 이동기능 구현해보기 (Transform.Translate())

B. Transform.translate 이동기능을 통해 실제 메타버스에서 캐릭터가 이동하도록 기능을 만들어 봅니다.

- Project창에서 마우스 우 클릭 후 Create – C# Script 항목을 선택하여 새로운 Script를 생성합니다.
- 생성 후 C# Script 이름을 MoveTranslate으로 변경합니다.
- MaleFree1 캐릭터를 선택 후 Inspector창에 Add Component 항목을 선택 한 뒤 MoveTranslate Script를 추가합니다.
- MoveTranslate Script를 열고 아래와 같이 이동 코드를 작성합니다.

☺ **과정 목표**

Object에 움직임을 주는 방법에 대해 알아보고 탐구합니다.

```
void Update()
    {
        if(Input.GetKey(KeyCode.W))
        {
            transform.Translate(Vector3.forward * Time.deltaTime);
        }
        if (Input.GetKey(KeyCode.S))
        {
            transform.Translate(Vector3.back * Time.deltaTime);
        }
        if (Input.GetKey(KeyCode.A))
        {
            transform.Translate(Vector3.left * Time.deltaTime);
        }
```

☺ **과정 목표**

Object에 움직임을 주는 방법에 대해 알아보고 탐구합니다.

```
        if (Input.GetKey(KeyCode.D))
        {
            transform.Translate(Vector3.right * Time.deltaTime);
        }
    }
```

- position 이동과 달리 Translate이동은 Local 좌표, 즉 Object 기준 좌표이기 때문에 World좌표와 달리 캐릭터 기준의 z축 좌표 기준이 정면으로 정의됩니다.
  또한, Translate를 사용하여 월드 축 기준으로 변경 하고 싶을 때 에는 인자 값에 Space.World를 추가 시 World 좌표 기준으로도 변경이 가능합니다.

☺ **과정 목표**

Object에 움직임을 주는 방법에 대해 알아보고 탐구합니다.

✓Unity Engine 안에서 Object 움직임을 주는 방법에 대해 알아봅니다.

2. 위 이동 방법을 이용한 실제 이동기능 구현해보기(Rigidbody.Addforce)

C. Rigidbody 이동기능을 통해 실제 메타버스에서 캐릭터가 이동하도록 기능을 만들어 봅니다.

- Project창에서 마우스 우 클릭 후 Create – C# Script 항목을 선택하여 새로운 Script를 생성합니다.
- 생성 후 C# Script 이름을 RigidbodyMove으로 변경합니다.
- MaleFree1 캐릭터를 선택 후 Inspector창에 Add Component 항목을 선택 한 뒤 RigidbodyMove Script를 추가합니다.
- 추가로 물리적으로 이동하기 위해 Rigidbody Component도 추가합니다.
- 현재 캐릭터에 Collider가 없으므로 Capsule Collider Component도 추가합니다.
- RigidbodyMove Script를 열고 아래와 같이 이동 코드를 작성합니다.

☺ **과정 목표**

Object에 움직임을 주는 방법에 대해 알아보고 탐구합니다.

```
public class RigidbodyMove : MonoBehaviour
{
    Rigidbody rigid;  // Rigidbody Component를 사용하기 위해 변수 선언
    float speed;  // 실제 캐릭터가 이동 할 물리적인 속도
    // Start is called before the first frame update
    void Start()
    {
        speed = 5.0f;  // 속도는 5
        rigid = GetComponent<Rigidbody>();  // rigid변수에 Rigidbody Component 할당 (초기화)
    }
```

☺ **과정 목표**

Object에 움직임을 주는 방법에 대해 알아보고 탐구합니다.

```
void Update()
  {
    if(Input.GetKey(KeyCode.W))
    {
      rigid.AddForce(Vector3.forward * speed);
    }

    if (Input.GetKey(KeyCode.A))
    {
      rigid.AddForce(Vector3.left * speed);
    }
```

☺ **과정 목표**

Object에 움직임을 주는 방법에 대해 알아보고 탐구합니다.

```
if (Input.GetKey(KeyCode.D))
    {
       rigid.AddForce(Vector3.right * speed);
    }

    if (Input.GetKey(KeyCode.S))
    {
       rigid.AddForce(Vector3.back * speed);
    }
 }
```

☺ **과정 목표**

Object에 움직임을 주는 방법에 대해 알아보고 탐구합니다.

✓Unity Engine 안에서 Object 움직임을 주는 방법에 대해 알아봅니다.

3. 위 이동 방법 중 하나를 통하여 간단하게 이동 제어하기

A. 각 키에 대해 이동방향을 직접 설정하려면 코드가 길어지고 보기에도 복잡해 지는 경향이 있기에 간단하게 정리하여 이동 할 수 있도록 만들어 보도록 하겠습니다.
아래 코드를 참조하여 만들어 봅시다.

```
public class CharacterMove : MonoBehaviour
{
   float speed;   // 속도 변수
   float horizontal;  // X 값을 받을 변수
   float vertical;  // Z값을 받을 변수

void Start()
   {
      speed = 5;  // 이동속도 5 초기화
   }
```

☺ **과정 목표**

Object에 움직임을 주는 방법에 대해 알아보고 탐구합니다.

```
// Update is called once per frame
   void Update()
   {
      horizontal = Input.GetAxisRaw("Horizontal");  //Input Class에 내장되어 있는 값 변수에 초기화 X값
      vertical = Input.GetAxisRaw("Vertical"); ");  //Input Class에 내장되어 있는 값 변수에 초기화 Z값
      Vector3 vec = new Vector3(horizontal, 0, vertical).normalized * Time.deltaTime; // 새로운 vector 변수 선언 후 값 저장

      if(horizontal !=0 || vertical !=0) // 현재 값이 둘 중 하나라도 0이 아니라면
      {
         transform.Translate(vec * speed); // 캐릭터 이동
      }
```

☺ **과정 목표**

Object에 움직임을 주는 방법에 대해 알아보고 탐구합니다.

3. Character 키보드로 이동 시 회전 기능을 추가로 만들어 봅시다.

- Translate 이동 시 Character Local좌표로 이동하기에 이동 부분은 forward 방향만 남겨둡니다.
- 각 방향 선택 후 눌렀을 때 캐릭터를 회전 시키도록 아래와 같이 코드를 작성합니다.

```
public class Character : MonoBehaviour
{
    float speed;  // 기존 Speed 변수
    float rotate;  // 초기 캐릭터의 회전 축 Y값
    float curRotateY;  // 현재 캐릭터의 회전 축 Y값

    // Start is called before the first frame update
```

☺ **과정 목표**

Object에 움직임을 주는 방법에 대해 알아보고 탐구합니다.

```
void Start()
  {
    speed = 5.0f;
    rotateY = transform.rotation.eulerAngles.y;
    curRotateY = rotateY;
  }

  // Update is called once per frame
  void Update()
  {
    if (Input.anyKey)
    {
      transform.Translate(Vector3.forward * Time.deltaTime * speed);
      transform.rotation = Quaternion.Euler(new Vector3(0, curRotateY, 0));
    }
```

☺ **과정 목표**

Object에 움직임을 주는 방법에 대해 알아보고 탐구합니다.

```
    if (Input.GetKey(KeyCode.W))
{
    curRotateY = rotateY;
}
if (Input.GetKey(KeyCode.A))
{
    curRotateY = rotateY - 90;
}
if (Input.GetKey(KeyCode.S))
{
    curRotateY = rotateY - 180;
}
if (Input.GetKey(KeyCode.D))
{
    curRotateY = rotateY + 90 ;
}
```

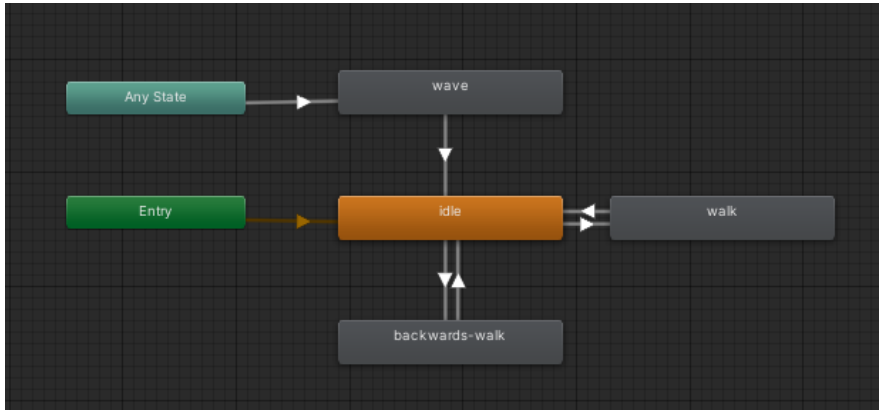
M5. Character Animation 추가

☺ **과정 목표**

Unity Animation의 기능과 사용법을 탐구합니다.

✓해당 캐릭터에 이동기능을 추가하였지만 이동할 때 부자연 스러운 부분이 많으므로,
캐릭터에 애니메이션을 추가하여 움직임을 주는 방법에 대해 알아봅시다.

1. Character Animation기능을 추가하기 위에 Animation Controller에 Clip 적용하기
   - Project 창에서 마우스 우 클릭 후 Create -> Animation Controller 항목을 선택합니다.
   - 선택 후 해당 Controller의 이름을 CharacterAnimController으로 명명합니다.
   - 해당 Controller를 더블클릭 후 컨트롤러를 엽니다.
   - Assets₩Supercyan Character Pack Free Sample₩Animations 해당 경로에 Character의 Animation을 가져와 아래와 같이 Animation Controller에 배치합니다.



☺ **과정 목표**

Unity Animation의 기능과 사용법을 탐구합니다.

2. Character Animation 기능을 추가하기 위해 Transition에 Parameter 설정하기

- Parameters 탭으로 이동 후 Bool Type으로 각각 FrontMove , BackMove 변수를 생성합니다.
- Trigger Type의 Wave 변수를 생성합니다.
- Entry 항목에서 Animation Clip에 이어지는 Transition을 클릭 하여 Conditions 값에 파라미터를 추가합니다.

  Idle <-> walk = FrontMove

  Idle <-> Backwards = BackMove

  Ani State -> wave = Wave
- Script로 제어 할 예정이기에 각 Transition의 Has Exit Time은 비활성화 합니다.
- 기존 Character Script로 가서 추가로 코드를 작성해 줍니다.

☺ **과정 목표**

Unity Animation의 기능과 사용법을 탐구합니다.

```
public class Character : MonoBehaviour
{
    float speed;
    float horizontal;
    float vertical;

    Animator animator;  // Animator Controller Component를 사용하기 위해 변수 선언

    // Start is called before the first frame update
    void Start()
    {
        speed = 5;
        animator = GetComponent<Animator>();  // Animator Controller Component를 변수에 할당
    }
```

☺ **과정 목표**

Unity Animation의 기능과 사용법을 탐구합니다.

```
void Update()
  {
      CharacterAnimation();  // 기존 코드에 Animation 호출 함수 추가

      horizontal = Input.GetAxisRaw("Horizontal");
      vertical = Input.GetAxisRaw("Vertical");
      Vector3 vec = new Vector3(horizontal, 0, vertical).normalized * Time.deltaTime;

      if(horizontal !=0 || vertical !=0)
      {
          transform.Translate(vec * speed);
      }
  }
```

☺ **과정 목표**

Unity Animation의 기능과 사용법을 탐구합니다.

```
void CharacterAnimation()  // 함수 생성
  {
      if (vertical > 0)  // z축이 0보다 클 경우
      {
          animator.SetBool(“FrontMove”, true);  // FrontMove 애니메이션 실행
          animator.SetBool("BackMove", false);  // BackMove 애니메이션 중지
      }
      else if (vertical < 0)  // z축이 0보다 작을 경우
      {
          animator.SetBool("FrontMove", false);  // FrontMove 애니메이션 중지
          animator.SetBool("BackMove", true);  // BackMove 애니메이션 실행
      }
```

☺ **과정 목표**

Unity Animation의 기능과 사용법을 탐구합니다.

```
        else  // 그 외 vertical값이 0 이라면
        {
            animator.SetBool("FrontMove", false);  // 애니메이션 중지
            animator.SetBool("BackMove", false);   // 애니메이션 중지
        }

        if(Input.GetKeyDown(KeyCode.Space))   // space키를 눌렀을 때
        {
            animator.SetTrigger("Wave");  Wave 애니메이션 실행
        }
    }
```

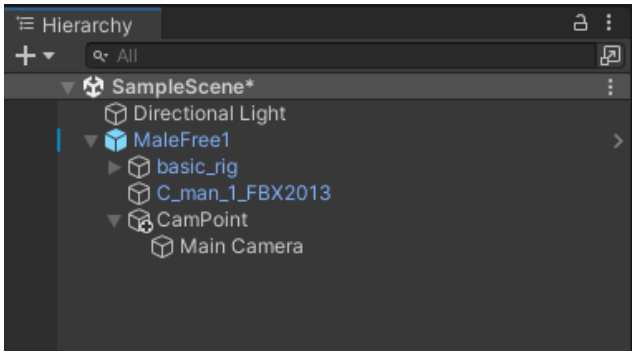
M6. Unity Camera
Follow & Rotation

☺ **과정 목표**

Camera가 Character를 따라다니며 사용자에 의해 회전하는 방법에 대해 알아보고 탐구합니다.

✓현재 캐릭터가 이동과 회전은 가능하지만 카메라 시점이 고정되어 있어 카메라가 캐릭터를 따라다니면서, 마우스를 이용하여 캐릭터를 회전 시키도록 하는 방법에 대해 알아봅니다.

1. 기존 Object에 새로운 Empty Object를 추가하여 CamPoint으로 이름을 변경합니다.
2. CamPoint Object에 Main Camera Object를 배치합니다.
3. 새로운 C# Script를 생성하여 이름을 CameraScript으로 변경 후 카메라 Object에 추가 합니다.
4. 해당 Script를 열고 아래와 같이 코드를 작성합니다.




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


public class CameraScript : MonoBehaviour
{
    public GameObject character;  // 외부에서 초기화 할 수 있도록 character 변수 선언
    public GameObject camPoint;  // 외부에서 초기화 할 수 있도록 camPoint 변수 선언
```

☺ **과정 목표**

Camera가 Character를 따라다니며 사용자에 의해 회전하는 방법에 대해 알아보고 탐구합니다.

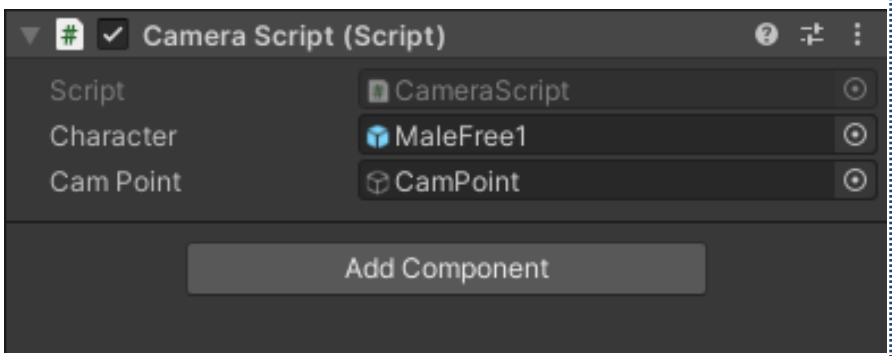
```
// Start is called before the first frame update
    void Start()
    {


    }

    // Update is called once per frame
    void Update()
    {
        float mouseX = Input.GetAxis("Mouse X");
         //Mouse X축의 -1 부터 1 사이의 값을 받아 mouseX변수에 초기화
        float mouseY = Input.GetAxis("Mouse Y");
         //Mouse Y축의 -1 부터 1 사이의 값을 받아 mouseY변수에 초기화
        Vector3 camAngle = camPoint.transform.rotation.eulerAngles;
         // camPoint Object의 Vector3 회전 값을 받아 camAngle 변수에 저장(초기화)
```

☺ **과정 목표**

Camera가 Character를 따라다니며 사용자에 의해 회전하는 방법에 대해 알아보고 탐구합니다.

```
        if (Input.GetMouseButton(1)) // 마우스 오른쪽 버튼을 눌렀을 때
        {
            character.transform.rotation = Quaternion.Euler(0, camAngle.y, 0);  //캐릭터의 회전 값을 적용
            camPoint.transform.rotation = Quaternion.Euler(camAngle.x - mouseY, camAngle.y + mouseX,
camAngle.z);
        // camPoint의 회전을 적용 X축은 3D Object 회전 축에서 상 하 를 나타냄으로 마우스 Y값을 빼서 X축 적용,
           같은 원리로 Y축은 좌우를 나타냄으로 mouseX축 (좌우) 를 추가하여 좌우를 나타냄
           여기서 각 축의 값을 빼거나 더하는 것은 방향을 나타냅니다.
        }
    }
}
```

- 이후 우측 사진과 같이 각 Object를 할당한 후,
  Play 시 마우스 회전 방향대로 캐릭터가 회전하는 것을
  볼 수 있습니다.



M7. Unity Collider

☺ **과정 목표**

Collider Component의 기능과 사용법을 탐구합니다.

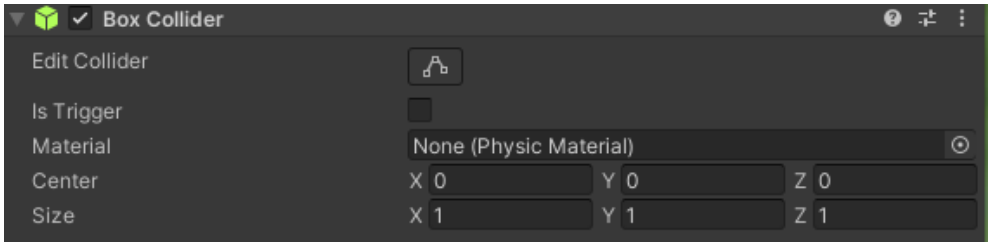
✓ 물리 충돌 처리를 위한 오브젝트의 형태를 정의 합니다.

✓ Collider는 보이지 않는 요소이므로 오브젝트의 메시와 정확히 동일한 모양일 필요는 없으며, 실제로는 게임 플레이시에는 대략적인 근사치로도 크게 구분되지 않으며 더 효율적입니다.

☺ **과정 목표**

Trigger Component의 기능과 사용법을 탐구합니다.

✓ 일반적으로 Collider와 비슷하지만 다른 형태로 Collider의 경우 충돌 처리를 위해 동작하지만, Trigger의 경우 실제로 오브젝트와 충돌이 일어나진 않지만 해당 충돌 시점을 감지하고, OnTRiggerEnter()함수를 호출하여 이벤트를 발생할 수 있습니다.



✓ Edit Collier - Collider Box의 사이즈를 수동으로 조절하실 수 있습니다.

✓ Is Trigger - 이 옵션을 활성화하면 이 Collider는 이벤트를 Trigger하는 데 사용되고 물리 엔진의 적용을 받지 않습니다.

✓ Material - Collider가 다른 Collider와 상호작용하는 방법을 결정하는 물리 Material에 대한 레퍼런스입니다.

✓ Center - 오브젝트의 로컬 공간에서 Collider의 포지션입니다.

✓ Size - X, Y, Z 방향에 있는 Collider의 크기.

☺ **과정 목표**

Rigid body Component의 기능과 사용법을 탐구합니다.

✓ Rigid body는 Game Object가 물리 제어로 동작하게 합니다.

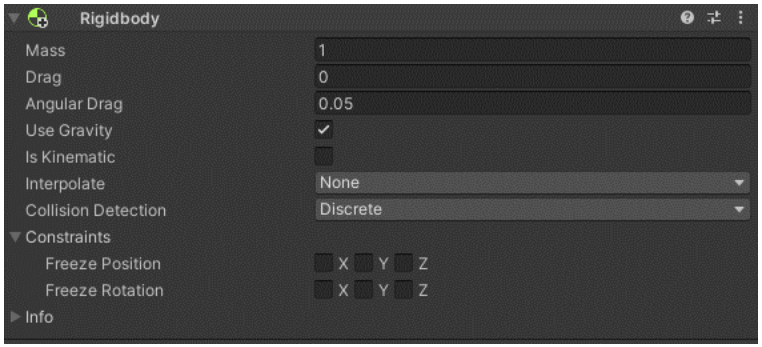
✓ 힘과 토크를 받아 오브젝트가 사실적으로 움직이도록 해줍니다.

✓ Rigid body가 포함된 모든 Game Object는 중력의 영향을 받으며, Scripting을 통해 가해진 힘으로 움직이거나 물리 엔진을 통해 다른 오브젝트와 상호 작용할 수 있습니다.

✓ Rigid body는 Game Object가 물리 제어로 동작하게 합니다.

☺ **과정 목표**

Rigid body Component의 기능과 사용법을 탐구합니다.



✓ Mass - Collider Box의 사이즈를 수동으로 조절 하실 수 있습니다.

✓ Drag - 이 옵션을 활성화하면 이 Collider는 이벤트를 Trigger하는데 사용되고 물리 엔진의 적용을 받지 않습니다.

✓ Angular Drag -Collider가 다른 Collider와 상호작용하는 방법을 결정하는 물리 Material에 대한 Reference입니다.

✓ Use Gravity - 오브젝트의 로컬 공간에서 Collider의 포지션입니다.

✓ Is Kinematic - X, Y, Z 방향에 있는 Collider의 크기.

✓ Interpolate - Rigid body의 움직임이 어색해 보일 경우 다음 옵션 중에서 변경하여 Setting합니다.

✓ Collision Detection - 빠르게 움직이는 오브젝트가 충돌의 감지 없이 다른 오브젝트를 지나쳐가는 것을 방지합니다.

☺ **과정 목표**

Collider Component의 기능과 사용법을 탐구합니다.

✓ 실제 충돌이 이루어지고 있는지 코드에서 확인하기 예제

1. 실제 충돌이 정상적으로 이루어지고 있는지 확인하기 위해 Script를 생성하여 확인해 봅시다.
2. Project 항목에서 마우스 우 클릭 후 Create – C# Script 항목을 선택하여 Script를 생성합니다.
3. 해당 Object에 Add Component 항목을 클릭 후 Rigidbody Component를 추가합니다.
4. 생성한 파일 이름은 ColliderCheck 으로 명명 후 스크립트를 열어줍니다.
5. Collider가 정상적으로 충돌이 이루어지고 있는지 확인하기 위해 3가지의 함수를 생성하여 줍니다
   - [OnCollisionEnter()] : 처음 Collider가 충돌됐을 때 호출되는 함수
   - [OnCollisionStay()] : Collider가 충돌되고 있을 때 호출되는 함수
   - [OnCollisionExit()] : Collider의 충돌이 끝났을 때 호출되는 함수
6. 함수 생성 후 메시지함수인 [Debug.Log()] 함수를 사용하여 다음과 같이 코드를 작성하여 줍니다.

☺ **과정 목표**

Collider Component의 기능과 사용법을 탐구합니다.

✓ Collider System을 이용하여 Item을 획득 할 때 Score가 증가되는 간단한 System을 만들어 봅니다.

1. Hiererchy창에 마우스 우 클릭 후 새로운 Sphere Object를 생성합니다.
2. Sphere Object를 적절한 위치에 배치 후 Inspector창의 Sphere Collider Component의 Is Trigger 항목을 Check 합니다. (Is Trigger 항목을 Check하게 되면 물리적으로의 충돌을 발생하지 않습니다.)
3. 아이템을 획득 할 대상(Player)의 Inspector 창의 Tag를 Player 으로 변경합니다.
4. Hierarchy창에 우 클릭 후 Text UI를 클릭하여 생성한 후 적절한 위치에 배치합니다.
5. Project창에 새 Script를 생성 후 ItemScript으로 변경 후 Script를 열어 코드를 작성합니다.

☺ **과정 목표**

Collider Component의 기능과 사용법을 탐구합니다.

```
public class ItemScript : MonoBehaviour
{
    float score = 20.0f;  // 스코어 점수

    private void OnTriggerEnter(UnityEngine.Collider other)  // 현재 Item Object와 Trigger 충돌 발생 시 실행 함수
    {
        if(other.tag =="Player")  // 충돌 된 객체의 Tag가 Player라면 실행
        {
            other.GetComponent<Character>().Score(score); // 닿은 객체가 가지고있는 Component 안의 함수 실행
            Destroy(this.gameObject);  // 현재 Script를 가지고있는 Object 삭제
        }
    }
}
```

☺ **과정 목표**

Collider Component의 기능과 사용법을 탐구합니다.

6. Character Script 에서도 Score를 확인 할 수 있도록 아래와 같이 코드를 추가합니다.

```
using UnityEngine.UI;  // UI Class 를 사용하기 위해 추가
using TMPro;  // TextMeshPro 기능을 사용하기 위해 추가

public class Character : MonoBehaviour
{
    public TextMeshProUGUI scoreText;  //TMP를 사용하려면 TextMeshProUGUI Class 사용 지금은 TMP으로 사용
    //public Text scoreText;  // 일반 Text를 사용하려면 Text Class 사용
    float totalScore; // 총 점수를 합산 할 변수
```

☺ **과정 목표**

Collider Component의 기능과 사용법을 탐구합니다.

```
void Start()
    {
        Score(totalScore); // Score함수를 실행하여 UI 값 초기화 (1회 실행)
    }

    public void Score(float score)
    {
        totalScore += score;  // 해당 함수가 실행 될 때마다 총 점수에 매개변수로 들어온 점수를 더함
        scoreText.text = string.Format("Score : {0}", totalScore);  // ScoreText에 총 누적 점수를 출력
    }
}
```

M8. Unity Video Player

☺ **과정 목표**

Unity에서 Video Player의 기능과 사용법을 탐구합니다.

✓Unity에서 Video Player를 재생하는 방법에 대해 알아봅니다.

✓기본적으로 UI에서 Play 하는 방법과 Object에서 Play하는 방법 2가지에 대해 알아봅니다.

1. UI에서 Video Play하는 방법
   - 먼저 재생시킬 영상 파일을 준비합니다. (mp4 또는 mov파일 권장)
   - Hierarchy창에서 마우스 우 클릭 후 UI - RawImage를 생성합니다.
   - 생성 된 RawImage UI를 적절한 위치에 배치시킵니다.
   - Hierarchy창에서 마우스 우 클릭 후 Video - Video Player를 생성합니다.
   - Video Player Object의 Inspector 창에서 Video Clip 부분에 준비한 영상 파일을 할당합니다.
   - Project창에서 마우스 우 클릭 후 Create – Render Texture를 생성합니다.
   - 해상도를 1920 1080 Size로 맞춰줍니다. (기본 영상 Size와 동일하게)
   - Video Player Object의 Inspector 창에서 Target Texture부분에 생성한 Render Texture를 할당합니다.
   - RawImage Object의 Inspector 창에서 Texture부분에 Render Texture를 할당합니다.
   - Play하여 정상적으로 영상이 출력되는지 확인합니다.

☺ **과정 목표**

Unity에서 VideoPlayer의 기능과 사용법을 탐구합니다.

2. 코드로 제어하여 동작하는 방법

- Video Player Component에서 Play On Awake 항목을 체크 해제합니다.
- Project창에 우 클릭 후 Create – C# Script를 생성합니다
- 생성 후 VideoPlayerScript으로 이름을 변경 후 Script를 열고 아래와 같이 코드를 작성합니다.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.Video;  // Video Class를 사용하기 위해 namespace 선언
Using UnityEngine.UI;  // UI Class를 사용하기 위해 namespace 선언

public class VideoPlayerScript : MonoBehaviour
{
    public RawImage rawImage;    // RawImage Type rawImage 변수 선언
    public VideoPlayer video;       // VideoPlayer Type video 변수 선언
    public RenderTexture render; // RenderTexture Type render 변수 선언
```

☺ **과정 목표**

Unity에서 VideoPlayer의 기능과 사용법을 탐구합니다.

```
void Update()
{
    if(Input.GetKeyDown(KeyCode.Alpha1))
    {
        rawImage.texture = render; // rendertexture를 rawImage의 texture에 할당
        video.Play();  //비디오 플레이
    }
    if (Input.GetKeyDown(KeyCode.Alpha2))
    {
        rawImage.texture = null;  // rawImage의 texture를 Null으로 초기화
        video.Stop();  // 비디오 중지
    }
}
```

☺ **과정 목표**

Unity에서 VideoPlayer의 기능과 사용법을 탐구합니다.

3. Object에서 Video Play하는 방법

- 먼저 재생시킬 영상 파일을 준비합니다. (mp4 또는 mov파일 권장)
- Hierarchy창에서 마우스 우 클릭 후 3D Object - Quad를 생성합니다.
- 생성 된 Quad를 적절한 위치에 배치시킵니다.
- Hierarchy창에서 마우스 우 클릭 후 Video - VideoPlayer를 생성합니다.
- Video Player Object의 Inspector 창에서 Video Clip 부분에 준비한 영상 파일을 할당합니다.
- Project창에서 마우스 우 클릭 후 Create – Render Texture를 생성합니다.
- 해상도를 1920 1080 Size로 맞춰줍니다. (기본 영상 Size와 동일하게)
- Video Player Object의 Inspector 창에서 Target Texture부분에 생성한 Render Texture를 할당합니다.
- Project 창에서 Create – Material 항목을 선택하여 material을 생성합니다.
- 생성 된 Material 의 Albedo 좌측 네모 칸에 Render Texture를 할당합니다.
- Quad Object의 Material을 Render Texture가 적용 된 Material로 변경합니다.
- Play하여 정상적으로 영상이 출력되는지 확인합니다.

☺ 과정 목표
Unity에서 VideoPlayer의 기능과 사용법을 탐구합니다.

4. 코드로 제어하여 동작하는 방법

- Video Player Component에서 Play On Awake 항목을 체크 해제합니다.
- Project창에 우 클릭 후 Create – C# Script를 생성합니다
- 생성 후 VideoPlayerScript으로 이름을 변경 후 Script를 열고 아래와 같이 코드를 작성합니다.

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

public class VideoPlayerScript : MonoBehaviour
{
    public GameObject quad;
    public VideoPlayer video;
    public Material renderMT;  // render Texture가 적용 된 Material
    public Material material;    // render Texture가 적용되기 전 기존 Material
```

☺ **과정 목표**

Unity에서 VideoPlayer의 기능과 사용법을 탐구합니다.

```
    private void Start()
    {
        material = quad.GetComponent<MeshRenderer>().material;  // material 변수에 기존 material 할당
    }
    void Update()
    {
        if(Input.GetKeyDown(KeyCode.Alpha1)){
            quad.GetComponent<MeshRenderer>().material = renderMT;  // renderTexture 적용 material로 변경
            video.Play();
        }
        if (Input.GetKeyDown(KeyCode.Alpha2)){
            quad.GetComponent<MeshRenderer>().material = material;  // 기존 material로 변경
            video.Stop();
        }
    }
}
```

M9. Unity AI System

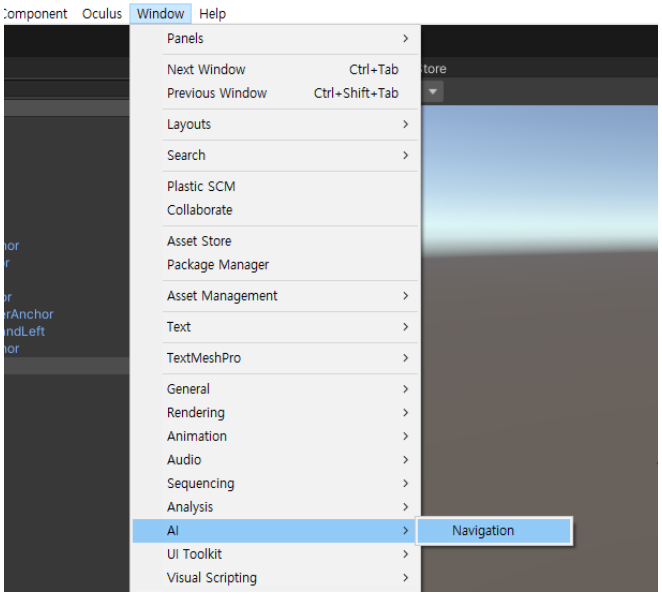
☺ **과정 목표**

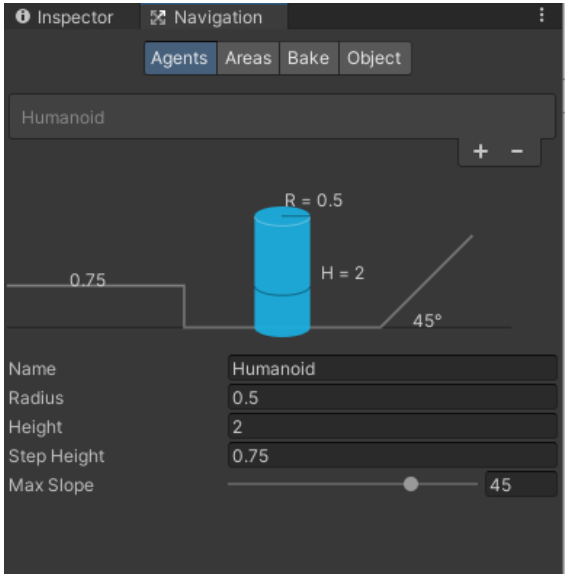
Unity Navigation 기능에 대해 알아봅니다.

✓ Window – AI – Navigation 항목을 선택합니다.

✓선택하면 Inspector 창 옆에 탭이 하나 추가로 생성되며 해당 탭에서 Navigation 옵션을 볼 수 있습니다.

✓해당 탭에서 지형에 대한 Agent의 의 키 넓이 경사각 등을 설정 할 수 있습니다.



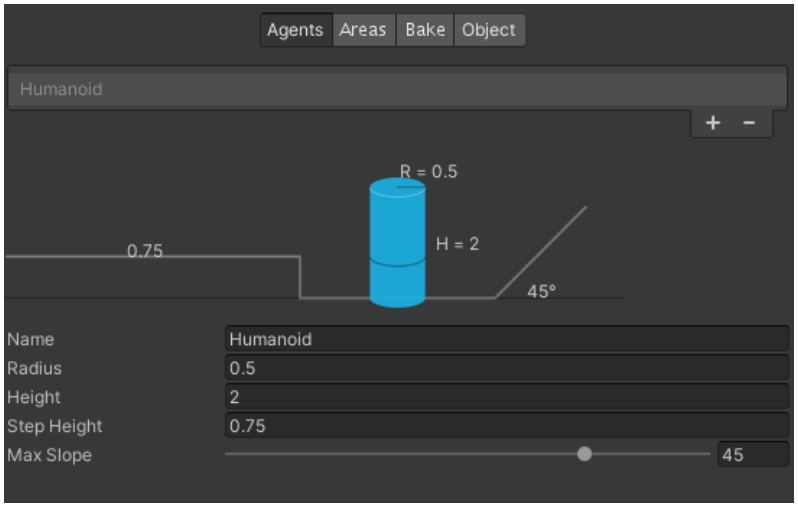


☺ **과정 목표**

Unity Navigation 기능에 대해 알아봅니다.

1. Navigation Agents탭의 기능 설명입니다.

- Name : 에이전트의 이름을 설정합니다
- Radius : 에이전트의 반지름을 설정합니다
- Height : 에이전트의 높이를 설정합니다.
- Step Height : 에이전트가 오를 수 있는 높이(계단)를 설정합니다.
- Max Slope : 에이전트가 올라갈 수 있는 경사도를 설정합니다.



☺ 과정 목표

Unity Navigation 기능에 대해 알아봅니다.

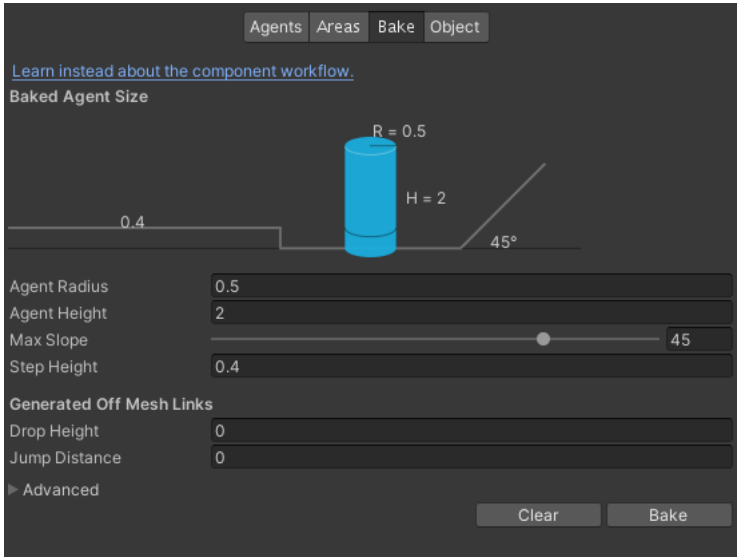
1. Navigation Baked탭의 기능 설명입니다.

Baked Agent Size

- Agent Radius : 에이전트가 지나갈 수 있는 반지름을 설정 합니다.
- Agent Height : 에이전트가 아래로 지나갈 수 있는 높이를 설정합니다.
- Max Slope : 에이전트가 올라갈 수 있는 경사 각도를 설정합니다.
- Step Height : 에이전트가 오르거나 내려갈 수 있는 높이(계단)를 설정합니다.

Generated Off Mesh Links : 언덕이나 사다리 절벽 등을 이동하게 함

- Drop Height : 에이전트가 이동 할 수 있는 절벽 아래의 높이
- Jump Distance : 에이전트가 뛰어 넘을 수 있는 거리
- Clear : 초기화
- Baked : 위 정보를 바탕으로 Navigation 정보를 Bake함.
- Bake 시 Project창에 NavMesh Scene이 생성됨.



☺ 과정 목표

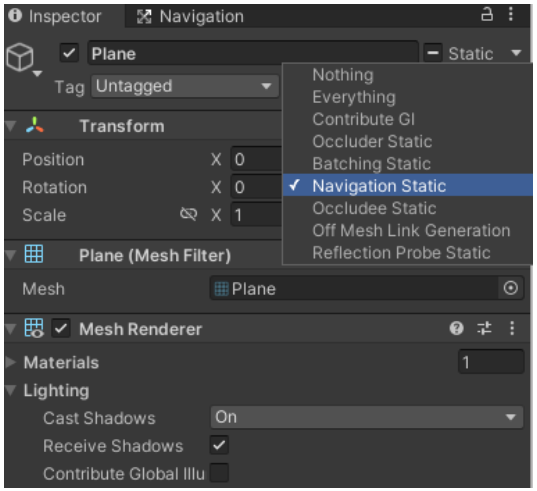
Unity Navigation 기능에 대해 알아봅니다.

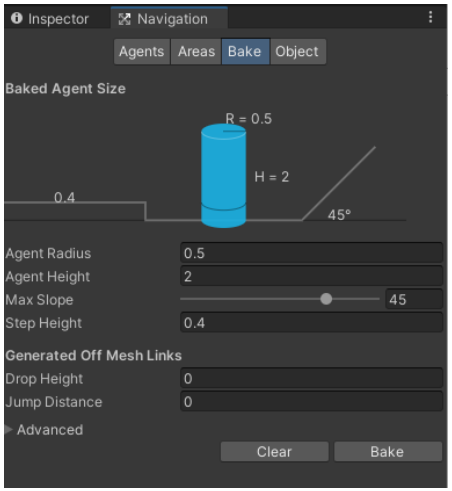
✓ Navigation은 실제 AI가 실제 이동할 지형을 설정하는 항목입니다.

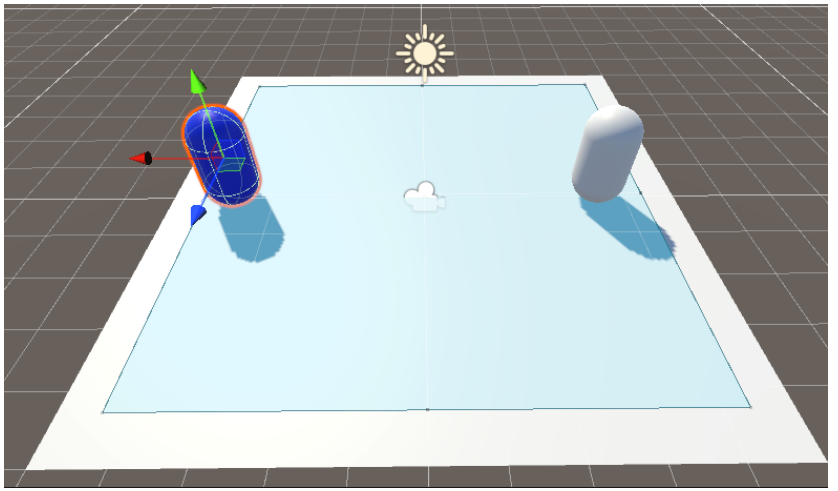
✓ AI가 지나갈 수 있는 지형Object를 선택하여 Navigation Static 으로 변경합니다.

✓설정 후 Navigation 탭으로 다시 돌아와 Bake 탭으로 들어간 후 아래 Bake 버튼을 클릭합니다.

✓해당 오브젝트는 이제 AI가 이동할 수 있는 지형 Object로 인식하여 아래 하늘색으로 활성화 된 지역 내에서 자유롭게 이동이 가능합니다.







☺ **과정 목표**

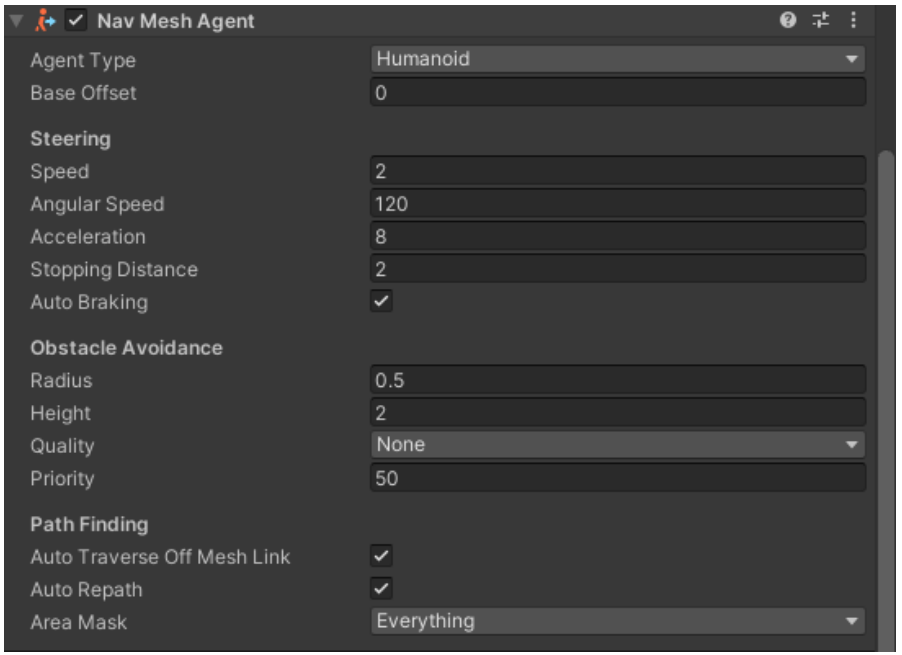
Unity Nav Mesh Agent 기능에 대해 알아봅니다.

1. Nav Mesh Agent Component에 대한 기능 설명입니다.

A. **Steering (Agent Move)**

- Component 생성 후 Inspector 창에서 Agent의 값을 Setting합니다.
  - Speed : Agent의 이동 속도입니다.
  - Angular Speed : Agent의 방향을 바꿀 때 회전 속도입니다.
  - Acceleration : Agent의 가속도입니다.
  - Stopping Distance : Agent가 목적지의 Distance 값에 가까워 졌을 때 멈추게 하는 기능입니다.
- Auto Braking : Agent가 목적지에 가까워 졌을 때 멈출 수 있습니다.

(목적지에 도착해도 에이전트를 멈추지 않을 때 사용)



☺ **과정 목표**

Unity Nav Mesh Agent 기능에 대해 알아봅니다.

1. Nav Mesh Agent Component에 대한 기능 설명입니다.

B. Obstacle Avoidance (Agent 장애물 회피)

- Radius : 장애물을 회피 시 Agent의 반지름
- Height : Agent의 높이
- Quality : Agent가 장애물과 충돌 시 처리 수준

(None이면 충돌 X 값이 올라갈수록 충돌 Check)

- Priority : 장애물과 충돌했을 때 우선순위

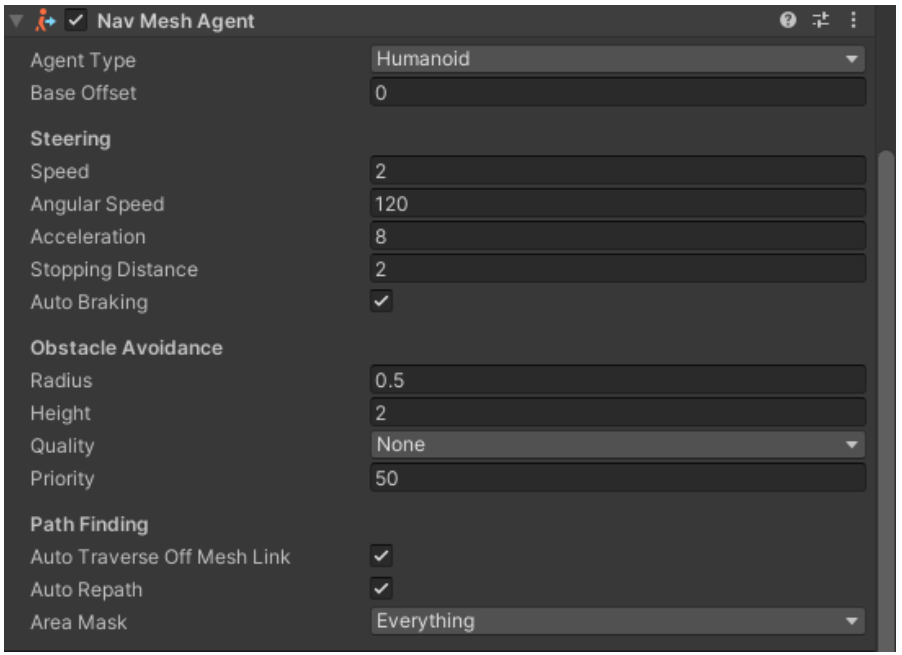
(이동 상태 시 해당 값에 따라 경로탐색의 우선순위)

C. Path Finding (경로탐색)

- Auto Traverse Off Mesh Link : Off Mesh Link가 존재 할 경우 자동으로 탐색하여 이동할지 여부
- Auto Repath : 이동 중 경로를 자동으로 탐색할지 여부

(장애물로 Agent가 지나갈 수 없을 때 경로 재 탐색)

- Area Mask : Agent가 이동 가능한 구역 지정

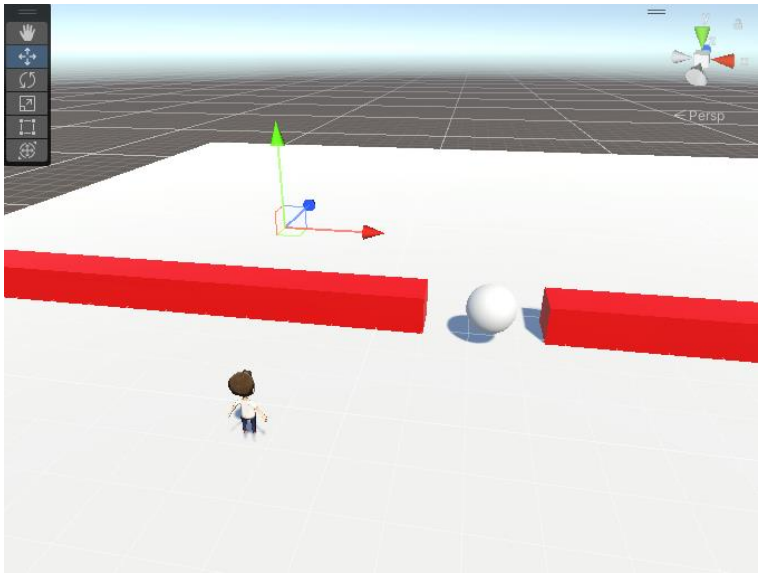


☺ **과정 목표**

Unity Navigation 기능에 대해 알아봅니다.

✓NavMeshAgent 기능을 활용한 간단한 예제.

1. NavMeshAgent를 이용하여 캐릭터를 이동해보기
   - 간단한 지형지물을 생성하여 Scene에 배치시킵니다.
   - Hierarchy 창에서 Empty Object를 생성 후 이동 할 방향에 배치시킵니다.
   - Character에 Nav Mesh Agent Component를 추가합니다.
   - Radius 항목과 Height 항목을 캐릭터에 맞게 설정합니다.
   - 장애물 Object의 Inspector창에서 Navigation Static 항목을 선택하여 Static Object로 변경합니다.
   - Window – AI – Navigation 항목을 선택합니다.
   - Bake 탭으로 가 지형을 Bake합니다.
   - Character Script로 돌아가 아래와 같이 수정합니다.



☺ **과정 목표**

Unity Navigation 기능에 대해 알아봅니다.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using TMPro;
using UnityEngine.AI;

public class Character : MonoBehaviour
{
    public TextMeshProUGUI scoreText;
    float totalScore;
    Animator animator;
    NavMeshAgent agent;  // Nav Agent 기능을 사용하기 위해 NavMeshAgent 변수 선언
    public Transform target;  // Agent가 목표 위치로 가기 위해 target 목표지점 선언
    bool isMove;  // 현재 이동 가능인지 Bool Type 변수 선언
```

☺ **과정 목표**

Unity Navigation 기능에 대해 알아봅니다.

```
// Start is called before the first frame update
void Start()
{
    agent = GetComponent<NavMeshAgent>();  //NavMeshAgent 기능을 사용하기 위해 agent 변수에 할당
    Score(totalScore);
    animator = GetComponent<Animator>();
}

void Update()
{
    if(Input.GetKeyDown(KeyCode.Z))  // Z키를 눌렀을 때 이동기능 활성화
    {
        isMove = true;
    }
```

☺ **과정 목표**

Unity Navigation 기능에 대해 알아봅니다.

```
        if(isMove)  // 이동 가능 상태 일 경우 실행
        {
            if(Vector3.Distance(transform.position , target.position) > 0.1f)
              // 현재 내 위치와 타겟의 위치가 0.1차이 이상 날 경우에 아래 코드 실행
            {
                agent.SetDestination(target.position); // Agent가 목표 위치로 이동
                animator.SetBool("FrontMove", true);  // 이동 시 Walk Animation 실행
            }
            else // 목표위치와의 거리 차이가 0.1 이하 일 경우 실행
            {
                isMove = false;  // 이동이 끝났음으로 false 초기화
                animator.SetBool("FrontMove", false); // 이동 Animation Stop
            }
        }
    }
```

☺ **과정 목표**

Unity Navigation 기능에 대해 알아봅니다.

```
    public void Score(float score)
    {
        totalScore += score;
        scoreText.text = string.Format("Score : {0}", totalScore);
    }
}
```

M10. Unity Raycast

☺ **과정 목표**

Ray, Raycast, RaycastHit의 기능과 사용법을 탐구합니다.

1. Unity Raycast는 간단하게 설명해서 현재 오브젝트에서 임의의 광선(Ray)를 쏴서 광선에 부딪힌 오브젝트를 감지하기 위한 메소드입니다.
2. Raycast에 감지 된 오브젝트의 정보를 RaycastHit 구조체 변수에 담아서 해당 변수에 접근하여 사용 할 수 있습니다.
3. Raycast는 Physics Class 안에 함수로 존재하며 RayCast의 메소드를 활용 해 오버로딩 되 있는 여러 함수들을 사용 할 수 있습니다.
4. RayCast의 반환 타입은 Bool 타입입니다. 지정된 위치에서 광선을 일정 거리만큼 쐈을 경우 true 또는 false값이 출력됩니다.
5. 값이 true일 경우 RaycastHit을 통해 충돌 된 객체의 정보를 out hitInfo 변수에 저장하여 사용 할 수 있습니다.
6. RayCast 사용 시 Ray를 쐈을 때 맞은 대상이 Collider가 있어야 체크가 가능합니다. 또한 Layer를 이용하여 특정 Layer가 포함 된 객체만 충돌 체크가 가능합니다.

☺ **과정 목표**

Ray, Raycast, RaycastHit의 기능과 사용법을 탐구합니다.

✓ Unity Raycast Parameter 소개

✓RayCast는 여러 함수를 제공합니다. 해당 함수의 Parameter가 하는 역할에 대해

✓설명합니다.

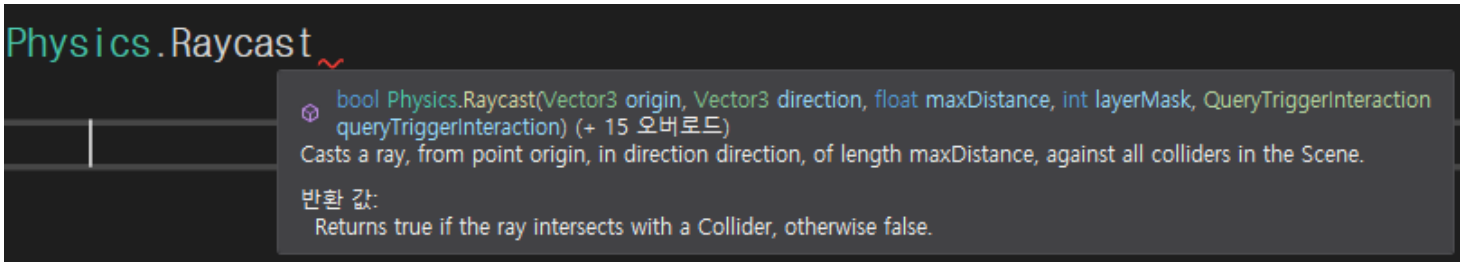
✓(Vector3) Origin : RayCast의 원점입니다. (광선의 시작 위치)

✓(Vector3) direction : 광선의 방향을 나타냅니다.

✓(Float) maxDistance : 광선이 나가는 거리를 나타냅니다.

✓(LayerMask) layermask : 광선이 오브젝트에 닿았을 때 선택적으로 필터링 할 때 사용합니다.

✓QuertTriggerInteraction : 광선이 Trigger 충돌을 감지 할 것인지 설정합니다.



☺ **과정 목표**

Ray, Raycast, RaycastHit의 기능과 사용법을 탐구합니다.

✓Ray에 대해 알아보기

1. Ray는 일반적으로 광선의 표현입니다. 실제 Ray를 원점에서 시작하여 어떤 방향으로 쏠 지에 대한 정보를 인자 값으로 받습니다.
   - Ray가 포함하는 정보 ray = direction (방향) , origin (원점)

설명

Representation of rays.

A ray is an infinite line starting at origin and going in some direction.

변수

direction	The direction of the ray.
origin	The origin point of the ray.

생성자

Ray	Creates a ray starting at origin along direction.

Public 함수

GetPoint	Returns a point at distance units along the ray.
ToString	Returns a nicely formatted string for this ray.

☺ **과정 목표**

Ray, Raycast, RaycastHit의 기능과 사용법을 탐구합니다.

✓RaycastHit에 대해 알아보기

1. RaycastHit은 Ray의 충돌 된 Object의 정보 값을 저장하는 역할을 합니다.
   일반적으로 충돌 될 시 대상에 Collider가 존재해야 충돌이 진행되어 값이 저장됩니다.
   - RaycastHit이 포함하는 정보

barycentricCoordinate	우리가 부딪힌 삼각형의 중심 좌표.
collider	충돌한 Collider입니다.
distance	광선의 원점에서 충돌 지점까지의 거리입니다.
lightmapCoord	충돌 지점의 uv 라이트맵 좌표입니다.
normal	광선이 닿은 표면의 법선.
point	광선이 충돌체에 충돌하는 세계 공간의 충돌 지점입니다.
rigidbody	충돌한 충돌체의 Rigidbody. 콜라이더가 강체에 부착되어 있지 않으면 null입니다.
textureCoord	충격 지점에서의 uv 텍스처 좌표입니다.
textureCoord2	충격 지점에서의 보조 uv 텍스처 좌표입니다.
transform	충돌한 rigidbody 또는 collider의 Transform입니다.
triangleIndex	적중된 삼각형의 인덱스입니다.

☺ **과정 목표**

Ray, Raycast, RaycastHit의 기능과 사용법을 탐구합니다.

✓ 실제 Raycast 사용 예제

1. 앞에 있는 오브젝트의 이름 출력하기 위해 RaycastTest라는 Script를 생성 후 위해 아래 코드를 작성합니다.

```
public class RaycastTest : MonoBehaviour
{
    RaycastHit hitInfo;  // Ray에 맞은 Object의 정보를 저장하기 위한 변수 선언
void Update()
    {
        if(Input.GetKeyDown(KeyCode.F)) {  // F 키를 눌렀을 때
            if (Physics.Raycast(transform.position, Vector3.forward, out hitInfo)) {
        //내 위치에서 정면에 Raycast를 쏴서 맞은 Object를 hitInfo 변수에 저장
              Debug.Log(hitInfo.transform.name); // hitInfo에 맞은 Object이름을 출력
            }
        }
    }
}
```

M11. Mouse Move System

☺ **과정 목표**

Mouse를 통해 Character를 이동하는 방법에 대해 알아봅니다.

1. 위에서 배운 Raycast 기능과 Agent 기능을 활용하여 Mouse로 해당 위치를 클릭 할 경우 목표지점으로 캐릭터를 이동시키게 하는 방법에 대해 알아봅니다.
   A. 마우스 클릭 좌표(2D)를 3D좌표로 변환하여 해당 좌표로 변환 해야함 (Raycast)
   B. RaycastHit 구조체를 이용하여 클릭 위치의 좌표 값을 저장.
   C. 저장 된 좌표로 Agent Component 기능을 통해 캐릭터 이동.
   D. LayerMask 기능을 통해 바닥면만 이동 범위로 이동.
2. 마우스 클릭 좌표를 3D 좌표로 변환하는 방법.
   - 유니티에서 기본 제공하는 ScreenPointToRay() 메소드를 활용하여 변환이 가능합니다.
   - 변환 코드

```
Ray ray; // ray 변수 선언
  ray = Camera.main.ScreenPointToRay(Input.mousePosition);
// MainCamera에서 Input메소드의 MousePosition (마우스의 현재 좌표) 를 받아 원점으로부터 마우스가
   위치 된 곳의 방향을 나타냅니다.
```

☺ 과정 목표

Mouse를 통해 Character를 이동하는 방법에 대해 알아봅니다.

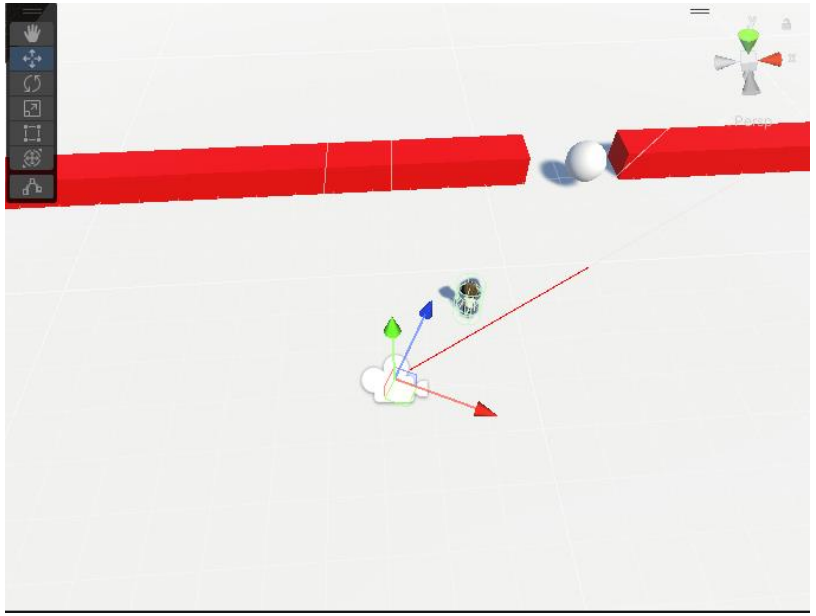
3. 실제 Ray가 가리키는 방향을 확인하기.

A. 실제로 Ray가 원점에서 어느 곳을 확인하는지 알고 싶을 경우가 있습니다.

유니티에서는 눈에 직접 확인 할 수 있도록 DrawRay()메소드를 지원합니다.

- DrawRay메소드 사용 방법.
- Debug.DrawRay(ray.origin , ray.direction * (거리) , Color.red (Ray색상))
- 해당 메소드를 통해 작성하게 되면 현재 내가

마우스로 가리키는 위치를 Ray로 눈에 나타나게 됩니다.



☺ 과정 목표

Mouse를 통해 Character를 이동하는 방법에 대해 알아봅니다.

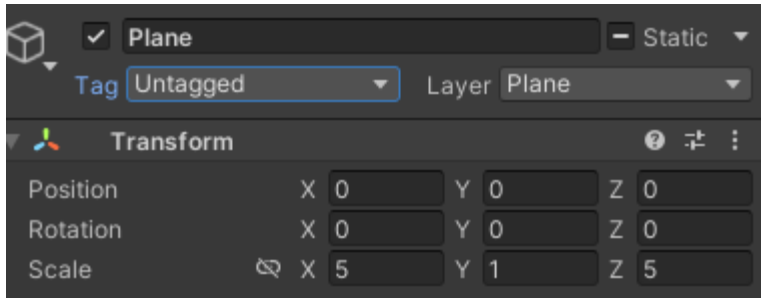
4. 실제 Agent Setting과 코드를 작성하여 Character 움직여보기

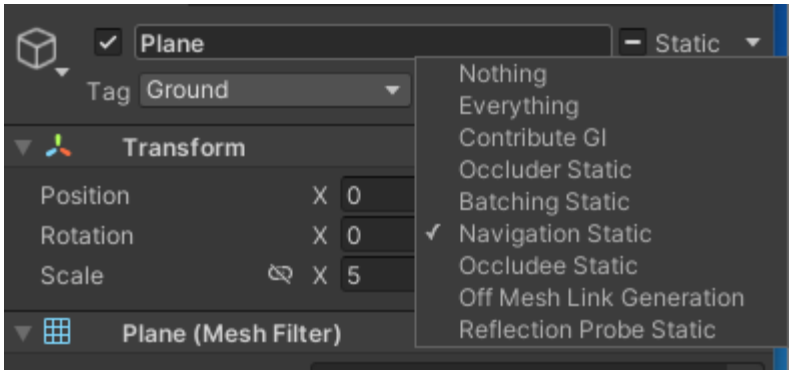
A. Navigation Setting

- Plane(바닥) Object를 클릭합니다.
- Object의 Layer항목을 선택 후 Add Layer를 엽니다.
- 빈 항목에 Plane 으로 Layer를 추가 한 후 Layer를 Plane으로 변경합니다.

// Layer를 변경하는 이유는 해당 Layer에만 Ray에 반응하도록 하기 위함 입니다.

- Plane Inspector의 Static항목을 Navigation Static으로 변경합니다.
- 추가적인 Object가 배치되어 있을 경우 해당 Object도 Navigation Static 항목을 Check해야 합니다.



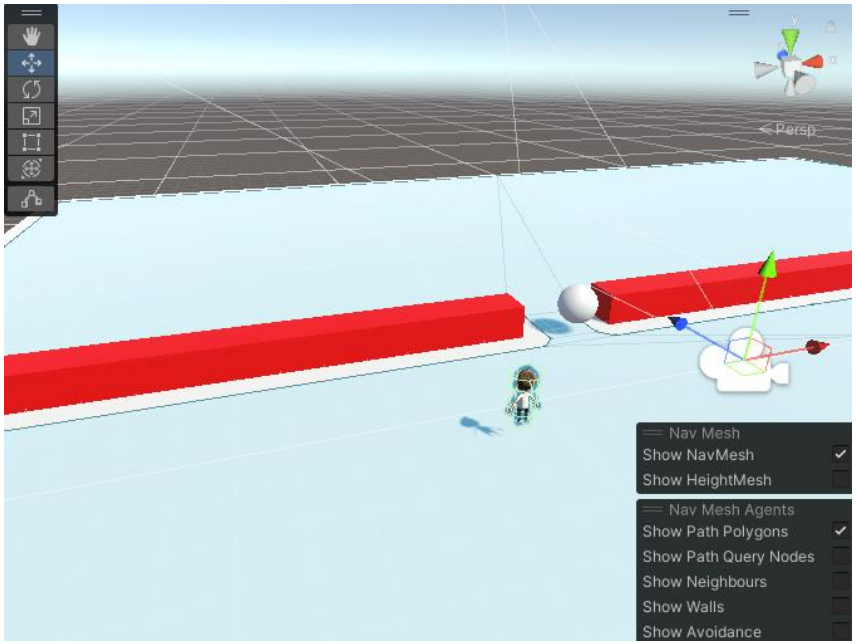


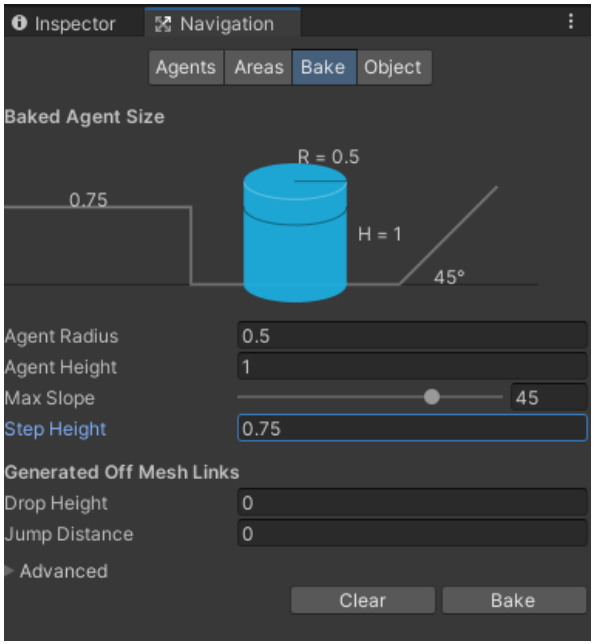
☺ 과정 목표

Mouse를 통해 Character를 이동하는 방법에 대해 알아봅니다.

- Window – AI – Navigation 항목을 선택합니다.
- Agent에서 실제 캐릭터가 이동 할 수 있는 Radius와 Height를 설정합니다.

  // Character의 Radius = 0.3 / Height = 1
- Bake 탭으로 넘어와서 Bake 항목을 선택하여 지형을 구워서 Agent가 이동할 범위를 지정해 줍니다.
- Bake 탭의 Step Height를 통하여 최대 넘을 수 있는 지형지물의 높이를 설정 할 수도 있습니다.



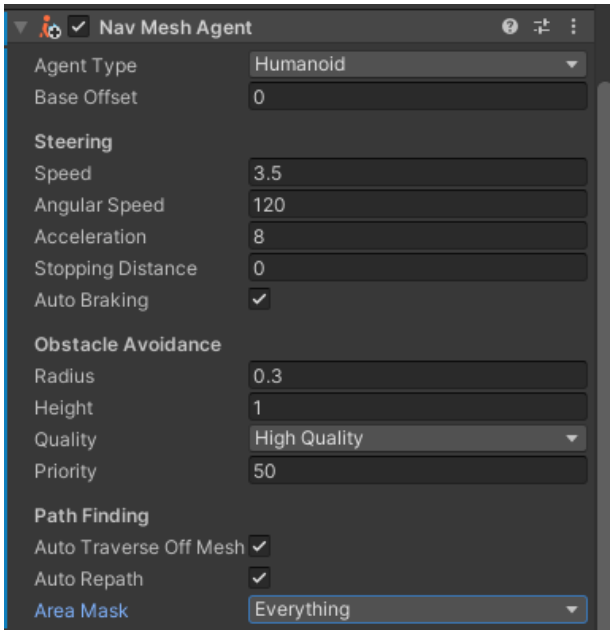


☺ **과정 목표**

Mouse를 통해 Character를 이동하는 방법에 대해 알아봅니다.

B. Nav Mesh Agent Setting

- Character에 Nav Mesh Agent Component 항목을 추가합니다.
- Steering 항목을 통해 Character의 속도 회전 가속도 등을 설정 할 수 있습니다.
- Obstacle Avoidance (장애물 회피) 항목의 Radius 항목을 0.3 Height 1 으로 변경합니다.
- 빈 항목에 Plane 으로 Layer를 추가 한 후 Layer를 Plane으로 변경합니다.



☺ **과정 목표**

Mouse를 통해 Character를 이동하는 방법에 대해 알아봅니다.

C. 총 종합한 코드 구현

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using TMPro;
using UnityEngine.AI;
```

☺ **과정 목표**

Mouse를 통해 Character를 이동하는 방법에 대해 알아봅니다.

```
public class Character : MonoBehaviour
{
    public Camera cam;
    public TextMeshProUGUI scoreText;
    //public Text scoreText;
    public LayerMask layer;
    float totalScore;
    bool isMove;
    Ray ray;
    RaycastHit hit;
    Vector3 moveVector;
    Animator animator;
    NavMeshAgent agent;
```

☺ 과정 목표

Mouse를 통해 Character를 이동하는 방법에 대해 알아봅니다.

```
void Start()
{
    agent = GetComponent<NavMeshAgent>();
    Score(totalScore);
    //speed = 5;
    animator = GetComponent<Animator>();
}

// Update is called once per frame
void Update()
{
    ray = cam.ScreenPointToRay(Input.mousePosition);
    Debug.DrawRay(ray.origin , ray.direction *100 , Color.red);
```

☺ 과정 목표

Mouse를 통해 Character를 이동하는 방법에 대해 알아봅니다.

```
if (Input.GetMouseButtonDown(0))
{
    ray = cam.ScreenPointToRay(Input.mousePosition);

    if (Physics.Raycast(ray, out hit,float.MaxValue , layer))
    {
        moveVector = hit.point;

        isMove = true;

        Debug.Log(moveVector);
    }
}
```

☺ 과정 목표

Mouse를 통해 Character를 이동하는 방법에 대해 알아봅니다.

```
if (isMove)
{
    if (Vector3.Distance(transform.position, moveVector) > 0.1f)
    {
        agent.SetDestination(moveVector);
        animator.SetBool("FrontMove", true);
    }
    else
    {
        isMove = false;
        animator.SetBool("FrontMove", false);
    }
}
```

☺ 과정 목표

Mouse를 통해 Character를 이동하는 방법에 대해 알아봅니다.

```
        if (Input.GetKeyDown(KeyCode.Space))
        {
            animator.SetTrigger("Wave");
        }
    }

    public void Score(float score)
    {
            totalScore += score;
            scoreText.text = string.Format("Score : {0}", totalScore);
    }
}
```

M12. Unity Webcam

☺ **과정 목표**

Unity에서 Webcam을 연결하여 사용하는 방법에 대해 탐구합니다.

✓Unity상에서 Webcam을 연결하여 캐릭터에 애니메이션을 추가하여 움직임을 주는 방법에 대해 알아봅시다.

1. PC에 기본적으로 Webcam Device가 연결되어 있어야 하며, 유니티에서 제공하는 Webcam Device와 Webcam Texture를 통해 간단하게 구현이 가능하기에 해당 방법에 대해 알아봅니다.
2. Webcam 실행 코드

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

public class Webcam : MonoBehaviour
{
    public RawImage display;  // 실제 Cam이 보여 질 이미지 Inspector에서 RawImage UI 할당
    WebCamTexture camTexture; // 실제 Cam 입력이 보여지는 텍스처
    private int currentIndex = 0; // 총 연결 된 Device 숫자
```

☺ **과정 목표**

Unity에서 Webcam을 연결하여 사용하는 방법에 대해 탐구합니다.

```
private void Start()
    {
        WebCamDevice[] devices = WebCamTexture.devices;  // 현재 인식 된 Device를 새 배열에 넣음
        for (int i = 0; i < devices.Length; i++)
        {
            Debug.Log(devices[i].name); // 총 연결 된 Cam Device의 이름을 출력
        }

        if (camTexture != null)  // 현재 Cam 입력이 있을 경우 (현재 사용 중일 경우)
        {
            display.texture = null; // 값 Null 초기화
            camTexture.Stop(); // 현재 장치를 중지
            camTexture = null; // 값 Null 초기화
        }
```

☺ **과정 목표**

Unity에서 Webcam을 연결하여 사용하는 방법에 대해 탐구합니다.

```
        WebCamDevice device = WebCamTexture.devices[currentIndex]; // 연결 된 장치를 device 변수에 넣음
        camTexture = new WebCamTexture(device.name);  // 연결 된 장치에 대한 새로운 Texture 생성
        display.texture = camTexture;  // 생성 된 Texture를 직접 보기 위해 RawImage 초기화
    }

    private void Update()
    {
        if(Input.GetKeyDown(KeyCode.Space))
        {
            camTexture.Play(); // Cam Texture Play. RawImage를 통하여 출력
        }
    }
}
```

End of Document