[2019년 1학기 실전프로젝트2 설계 보고서]

Unity를 이용한 스토리 기반 모바일 게임 개발

제출자: 곽영혜

제출날짜: 2019/5/25

지도교수: 조성배

목차

1. 문제 정의 ----------(p.03)

2. 배경이론 및 기존 방법들 ----------(p.05)

3. 제안된 문제접근방법 ----------(p.07)

4. 실험결과 ----------(p.09)

5. 결론 ----------(p.10)

* 플레이어와 Item이 부딪쳤을 때, Item이 사라지도록 설정하는 방법 ----------(p.11)

** 플레이어가 획득한 아이템의 개수를 Slider로 나타내는 방법 ----------(p.13)

6. 참고문헌 ----------(p.01)

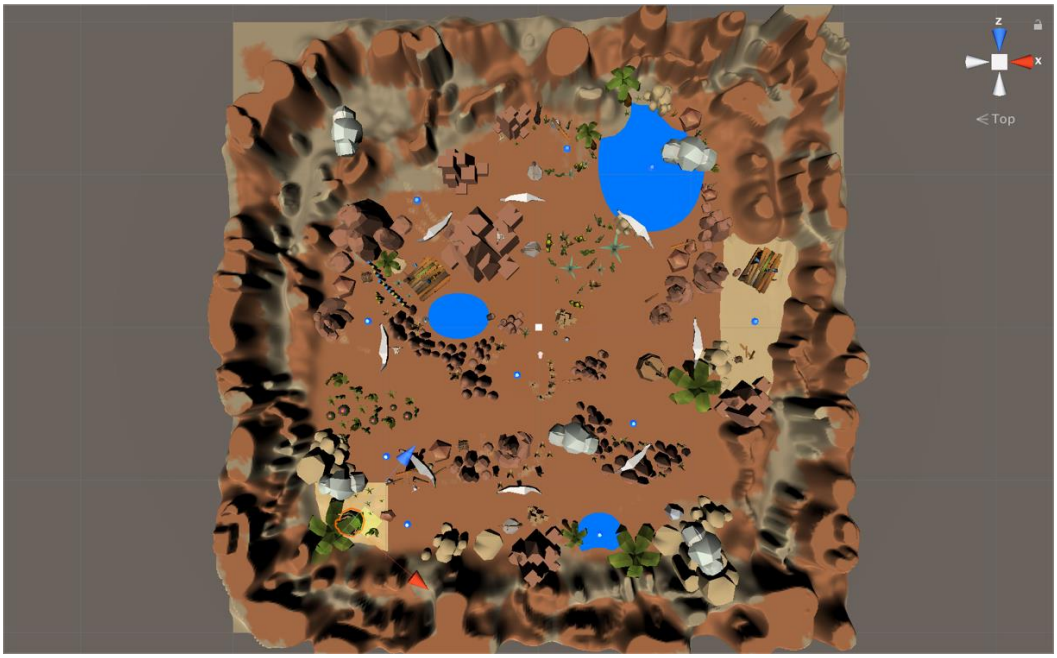
1. 문제 정의

게임의 개발 환경을 3D로 선정하고 이를 구현해 나가던 중, 맵 구성에서 수많은 asset들의 오브젝트들이 사용됨에 따라 유저와 물리적 충돌이 일어나게 될 경우, 이를 나타내도록 하는 component의 설정을 각 오브젝트들마다 다르게 설정 값을 주어야 하는 문제가 발생하게 되었습니다.

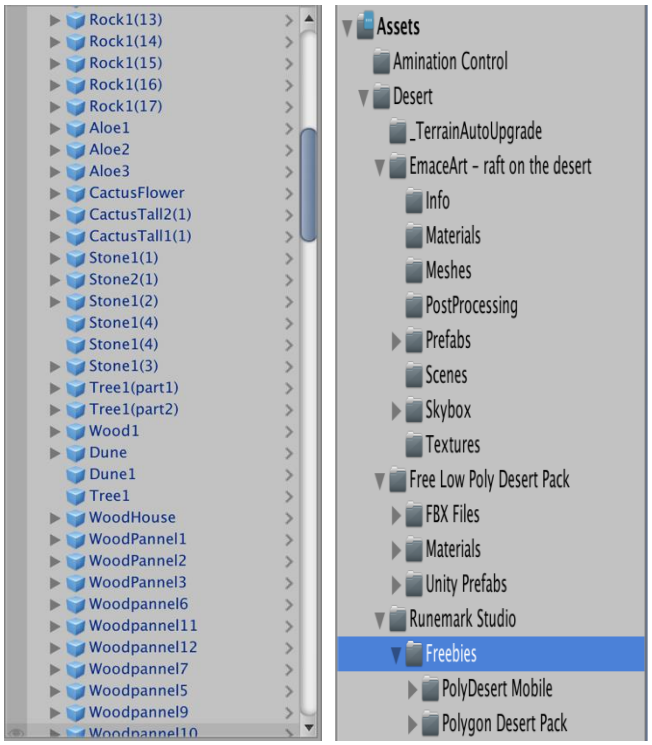
<Cat이라는 오브젝트에 대한 사진과 component 설정>

위의 사진을 보시게 되면 게임 맵 환경 위에 회색 박스의 게임 플레이어와 NPC 캐릭터인 고양이, 그리고 Item인 파란색 보석이 있음을 확인 할 수 있습니다. 이들 중에서 플레이어는 Item인 보석을 제외하고는 맵 환경, NPC 캐릭터와 충돌할 경우에는 물리력을 가져야 하고 이를 나타내는 component가 바로 Collider입니다.

Collider은 오브젝트의 형태에 따라 직육면체, 캡슐, 구 등으로 적용 범위를 조절할 수 있습니다.



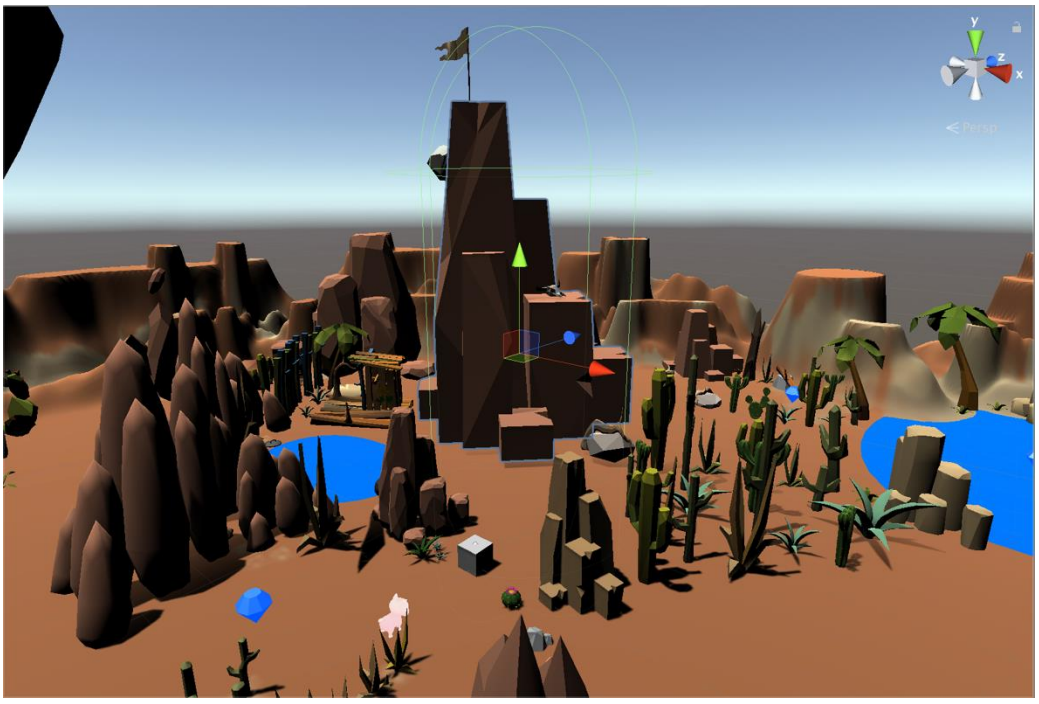
<게임 맵의 전체 크기(수 많은 오브젝트들이 포함되어 있음)>



그러나 맵을 구성하기 위해 많은 Asset들과 object들이 사용된 것을 확인 할 수 있습니다. 또한 동일한 object를 사용하여도 맵에서 구현할 때는 디자인적 구현을 위해 크기와 각도를 다르게 설정하였기에 그에 따른 Collider 범위 설정도 달라져야 했습니다.

2. 배경이론 및 기존 방법들

- 이미 게임 맵에 구현되어 있는 각 오브젝트의 설정마다 Collider를 적용하는 방법

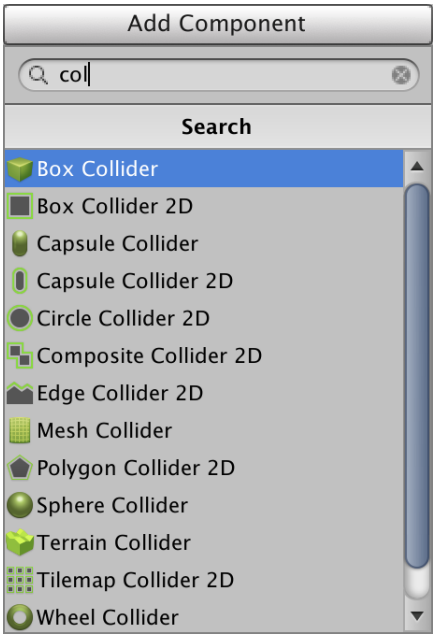


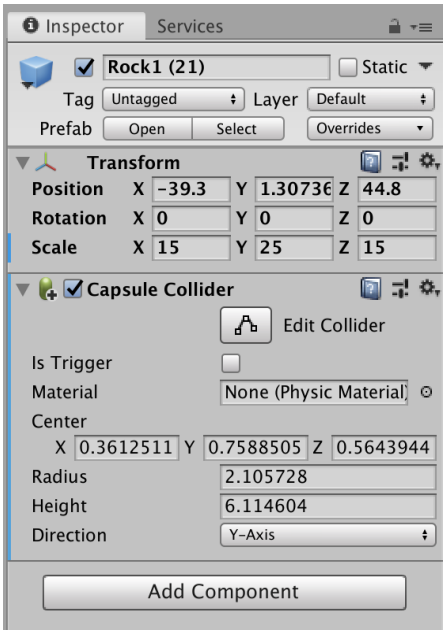
<게임 맵 상에 구현되어 있는 오브젝트들>

- 먼저 게임 맵 상에 오브젝트들을 적절하게 배치 후, 크기, 각도, 색상 등을 조절하여 디자인을 한 오브젝트들의 component 설정들을 수정해 나아가는 방식입니다.

장점 : 각 오브젝트 마다 물리적 충돌을 감지하는 Collider 적용 범위를 세밀하게 설정할 수 있습니다.

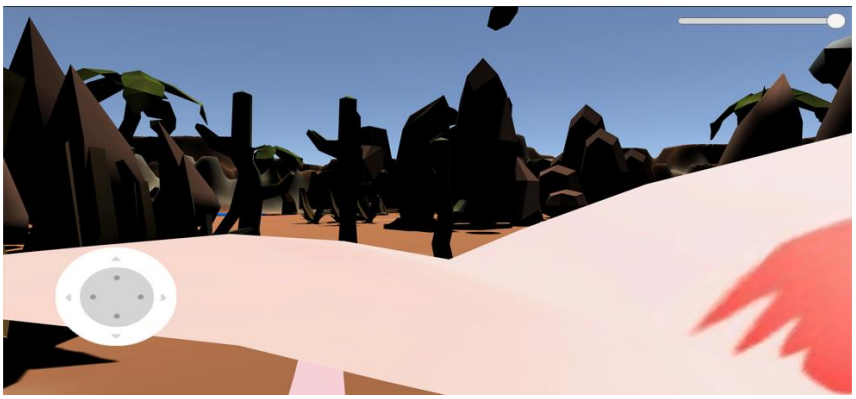
단점 : 게임 맵에 대량의 오브젝트들이 존재하기 때문에, 각 오브젝트를 수정하는데 많은 시간이 소요됩니다.





<오브젝트들 중 하나인 Rock에 대한 기본 설정 Component>

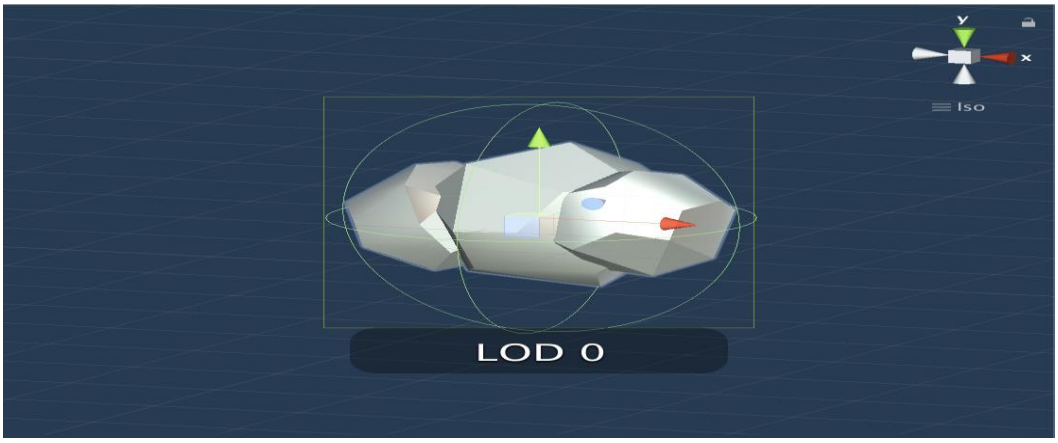
- 각 오브젝트에 대한 component 설정에서 "Add Component" 버튼을 클릭하여 오브젝트 모양에 맞는 Collider를 선택합니다.
- 선택된 Collider의 범위를 오브젝트에 맞게 편집하게 되면 플레이어와의 충돌을 감지하게 됩니다.
  - Collider 설정을 수정할 때, Is Trigger가 체크되지 않은 상태여야 됩니다. 만일 이것이 체크된 상태인 경우, 오브젝트와 플레이어와의 물리적 충돌이 일어나지 않고 플레이어가 오브젝트를 통과하게 됩니다.

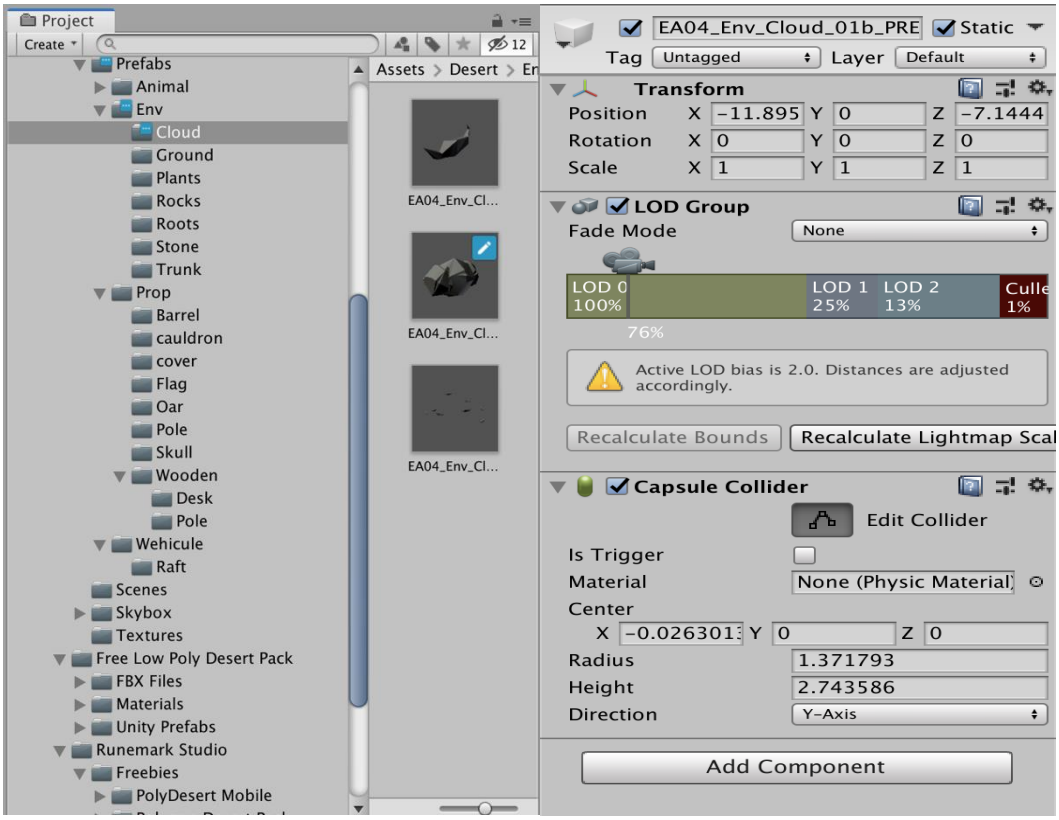


<플레이어가 Is Trigger가 체크되지 않은 오브젝트를 통과하는 장면>

3. 제안된 문제접근방법

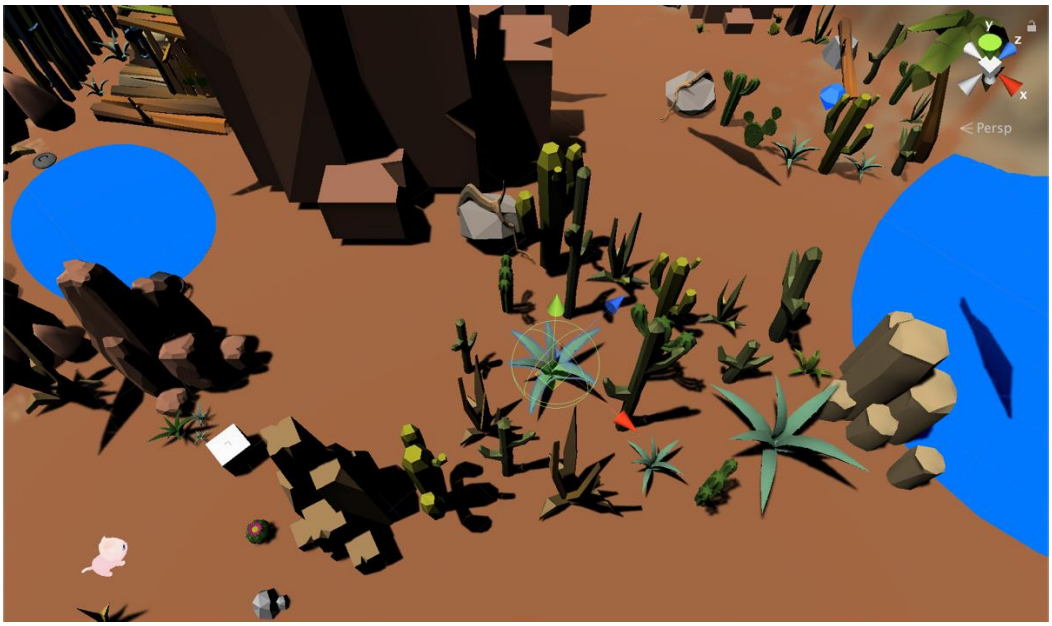
- Prefabs의 오브젝트마다 기본 설정에 Collider를 적용하고 이를 맵에 직접적으로 사용하거나 기존에 맵에 있는 오브젝트를 복사시켜서 수를 늘리는 방법





<오브젝트들 중 하나인 Cloud에 대한 기본 설정 Component>

- 위의 사진처럼 Asset 폴더에 저장된 prefabs에 있는 오브젝트에 대한 설정 창으로 이동합니다.
- component 설정에서 "Add Component" 버튼을 클릭하여 오브젝트 모양에 맞는 Collider를 선택합니다.
- 선택된 Collider의 범위를 오브젝트에 맞게 편집하게 되면 플레이어와의 충돌을 감지하게 됩니다.



<실제 게임 맵 상에 적용된 오브젝트(알로에)>

4. 실험결과

- 위의 방법을 통해 다수의 오브젝트들을 생성해야 하는 맵 디자인의 효율성을 높이면서 추후 플레이어가 물체에 물리적으로 충돌할 경우에 적용되는 코드를 쉽게 설정할 수 있었습니다.

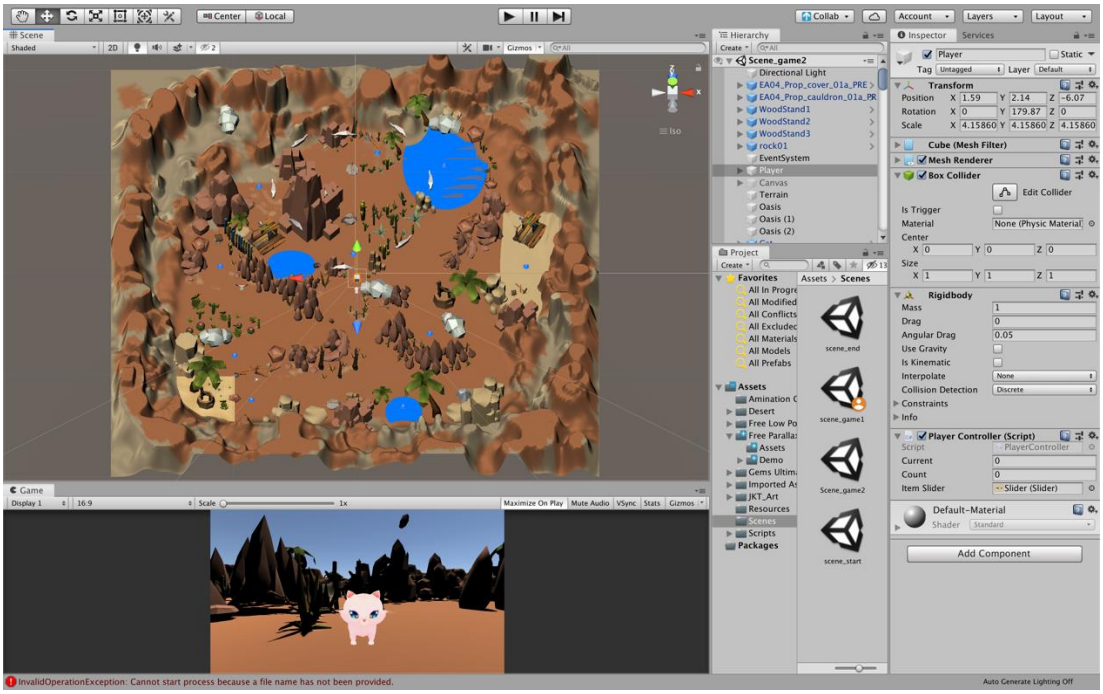
- 그러나 맵 디자인 적으로 오브젝트들의 외관 수정(크기, 각도, 색상, 반사 등)은 사용자가 각 오브젝트마다 따로 설정해 주어야 하는 필연적인 어려움은 극복할 수 없었습니다.

<동일한 오브젝트들(선인장)이 중복적으로 사용되어도 각 크기, 위치, 각도가 다름을 확인 할 수 있음>

5. 결론

- Asset들의 prefabs에 Collider를 적용하여 이를 게임 맵 상에서 구현하는 방법은 맵 디자인의 효율성을 높이지만, 추후 디자이너가 각 오브젝트의 외관을 세밀하게 수정하는 경우에는 큰 차이가 없음을 알 수 있었습니다.

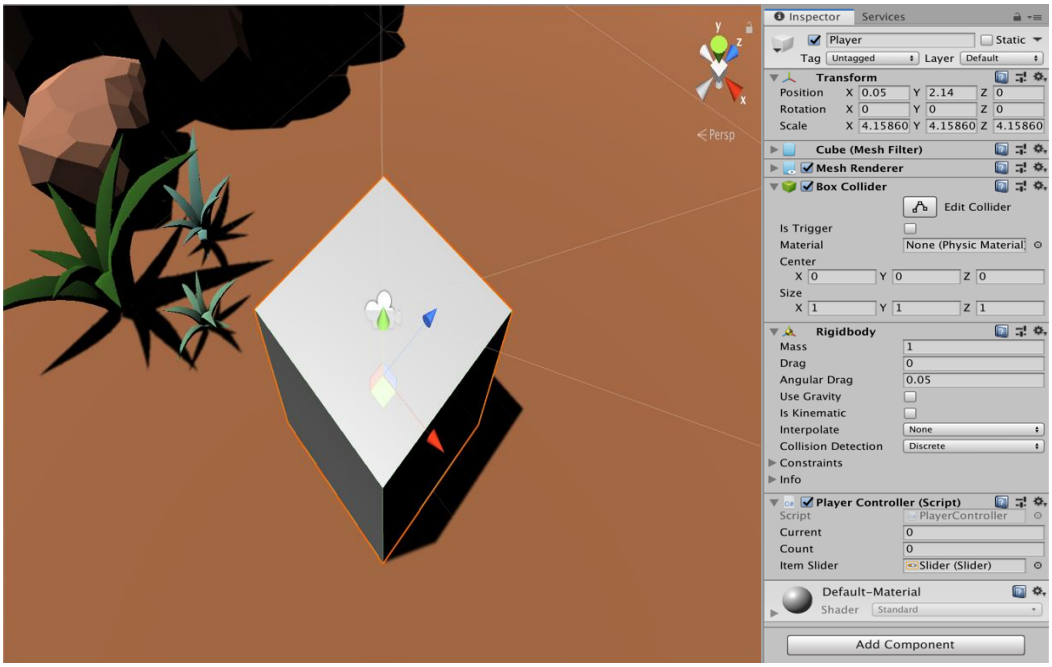
- 이번 프로젝트를 진행하면서, 향후 프로젝트를 진행하게 되면 많은 Scene들과 각 Scene을 구성하게 될 많은 오브젝트들을 디자인하는 데 있어 시간을 효율적으로 사용하여 프로젝트를 진행하여야 되는 중요성을 깨닫게 되었습니다.



<프로젝트 Scene들 중 하나, 많은 asset과 오브젝트들이 사용된 것을 확인할 수 있음>ㄴ

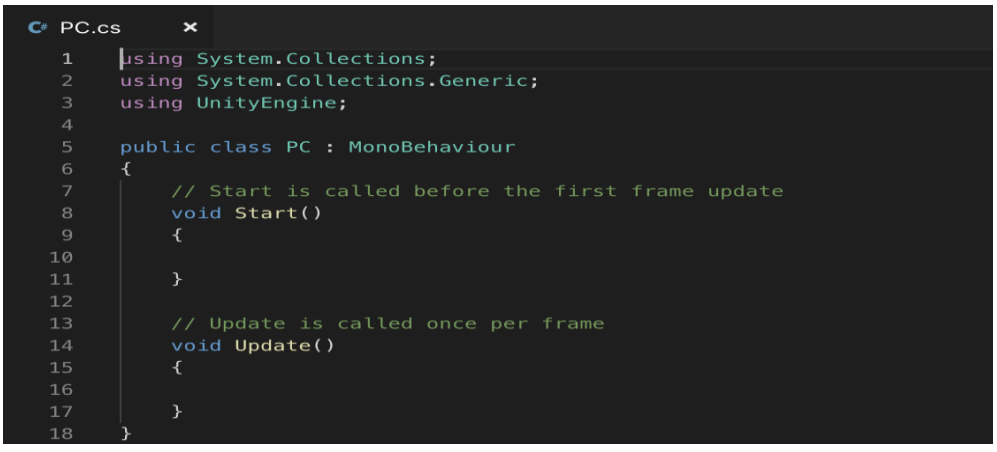
* 플레이어와 Item이 부딪쳤을 때, Item이 사라지도록 설정하는 방법

· C# Script로 플레이어가 각 상황에 따른 액션을 취하도록 설정

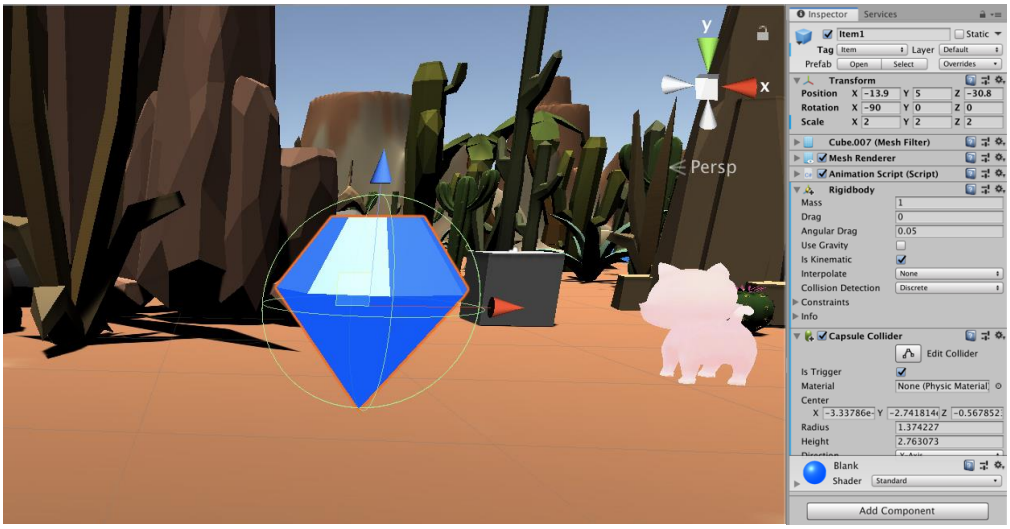


<게임의 플레이어와 오브젝트에 대한 Component 설정들>

- "Add Component"에서 "script"를 검색 후, "New Script"를 선택하여 플레이어 오브젝트에 적용할 새 스크립트를 작성합니다.



<C# Script를 생성하고 수정할 경우 나타나는 초기 설정들>



<Item과 오브젝트에 대한 Component 설정들>

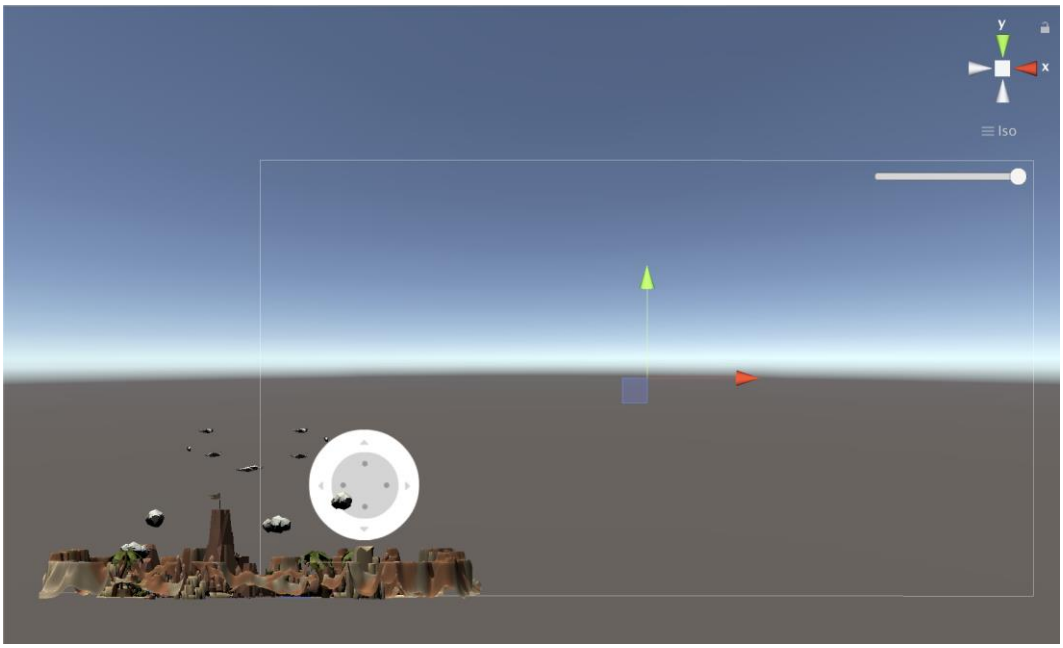
- Item 설정 창의 상단에서 Tag를 선택하여 "Add Tag"로 새로운 Tag "Item"을 생성 후, 적용시킵니다. 이를 모든 Item 오브젝트에 적용합니다.

```
20      void OnTriggerEnter(Collider other) //Player, Item Tag
21      {
22          if (other.gameObject.CompareTag("Item")) //Item과 부딪쳤을 경우
23          {
24              other.gameObject.SetActive(false); //부딪친 물체는 사라짐
```

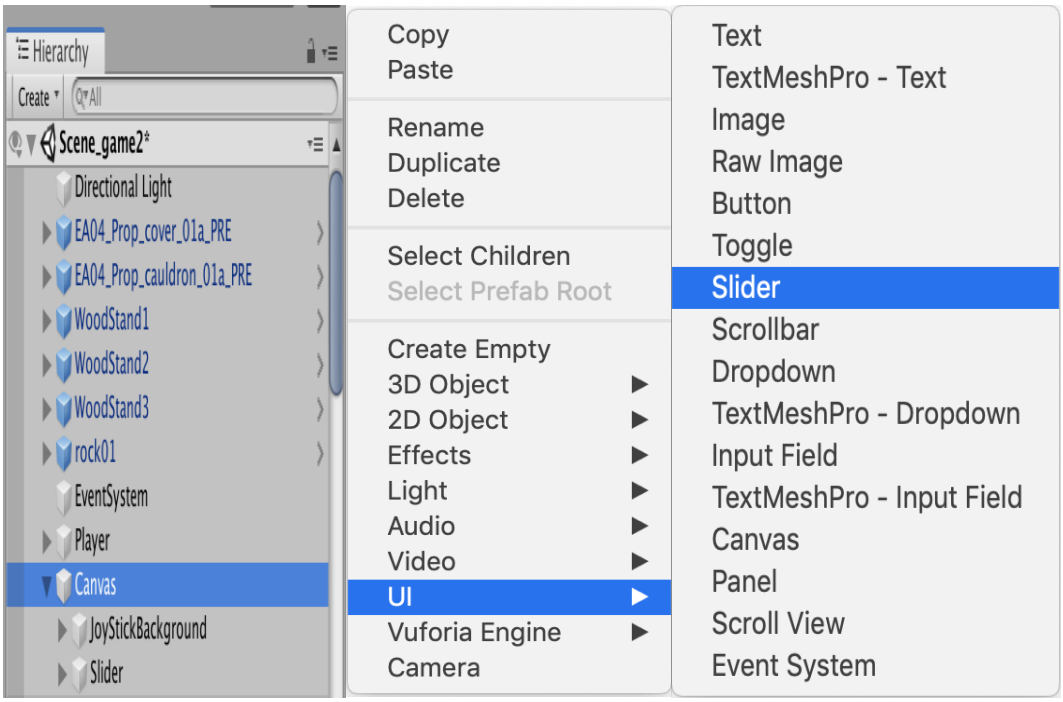
<"Item"이라는 Tag를 가진 오브젝트와 충돌하는 경우의 method>

- 플레이어와 부딪친 오브젝트들이 가진 Tag 이름이 Item들에 설정해 두었던 Tag 이름인 "Item"과 일치하는 경우, 부딪친 오브젝트들, 즉 Item들은 사라지도록 설정하여 플레이어가 Item을 획득한 것 처럼의 효과를 나타냅니다.

** 플레이어가 획득한 아이템의 개수를 Slider 로 나타내는 방법

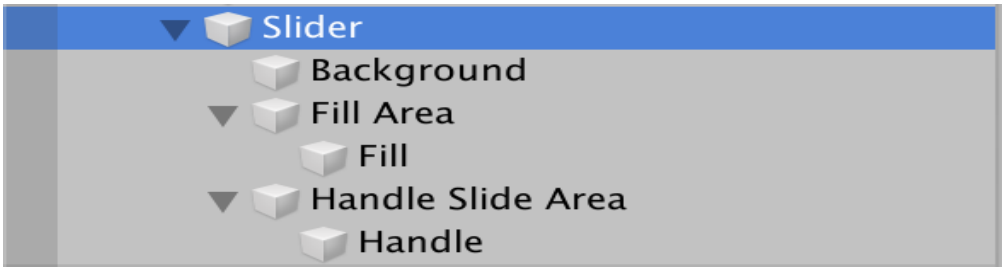


<게임 화면 창을 나타내는 Canvas>

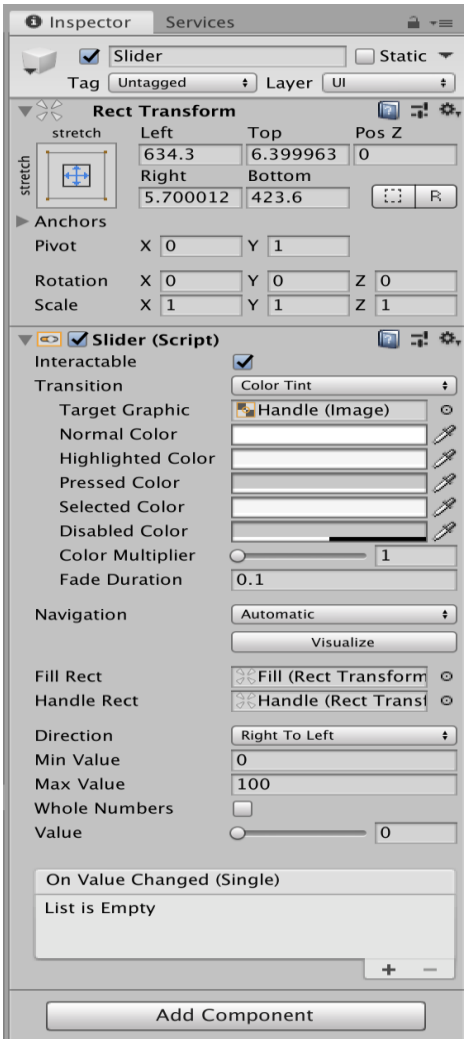


<Canvas 에 Slider UI 를 추가하는 과정>

- 위의 사진과 같이 게임 화면 창에 Item 획득 진척 상태를 확인하는 Slider 를 추가하기 위해서는 Canvas 에서 Slider UI 를 추가합니다.



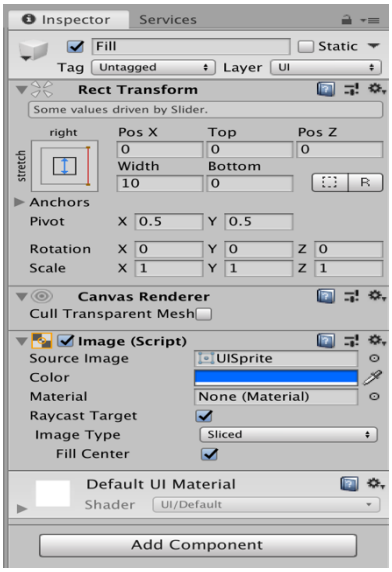
<Slider UI 의 기본 구성요소>



- Slider 의 기본 Component 요소들 중, Rect Transform 을 통해 게임 화면 상에서 Slider 가 어디에 나타날 것인지 위치를 설정합니다.

- 저희 팀은 우측 상단에 Slider 를 놓았기 때문에 Direction 을 설정하여 Slider 가 채워지는 방향을 오른쪽에서 왼쪽 방향으로 채워지도록 하였습니다.

- Slider 가 0 에서 끝까지 다 차는 것을 나타내기 위해 Min, Max Value 를 0 과 100 으로 설정합니다.



- Fill 오브젝트 Component 에서 Item 을 획득할 때마다 Slider 에 채워지는 색상을 파란색으로 설정합니다.

```
6   public class PlayerController : MonoBehaviour
7   {
8       public int current;
9       public int count = 0;
10      public Slider ItemSlider;
```

<플레이어에 대한 C# 스크립트에 Slider 에 나타낼 변수들 설정>

- 플레이어의 C# 스크립트를 수정하는데, Slider 의 현 상태를 나타내는 변수 current 와 초기 상태 값을 나타내는 변수인 count = 0(초기에는 Item 을 획득하지 않은 상태이기 때문), 그리고 이를 적용하는 곳인 Slider 를 생성합니다.

```
14      void Start() //Activate only once
15      {
16          rb = GetComponent<Rigidbody>(); //Get Player's Rigidbody, MonoBehaviour->GetComponent
17          current = count; //초기 카운터는 0개
18      }
```

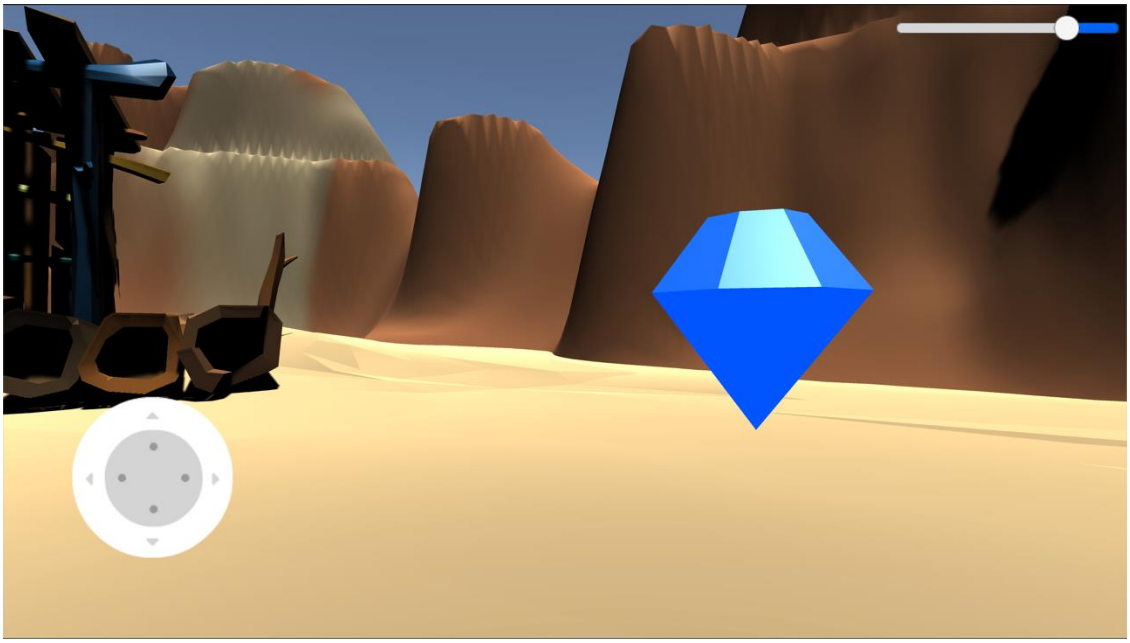
<플레이어에 대한 C# 스크립트에서 게임을 시작할 때, current 에 count 값을 대입>

- 게임이 시작되면 Slider 초기 값은 0 이므로 count 값을 current 에 저장합니다.

```
20      void OnTriggerEnter(Collider other) //Player, Item Tag
21      {
22          if (other.gameObject.CompareTag("Item")) //Item과 부딪쳤을 경우
23          {
24              other.gameObject.SetActive(false); //부딪친 물체는 사라짐
25              current += 10; //왼쪽 상단의 카운트 증가
26              ItemSlider.value = current;
27          }
28      }
```

<플레이어에 대한 C# 스크립트에 Item 과 부딪칠 경우 Slider 가 늘어나도록 설정>

- 플레이어와 Item 간의 물리적 충돌이 감지될 경우, current 의 값은 10(아이템 점수)만큼 증가하도록 하고 이 current 값을 Slider 에 나타나도록 설정합니다.



<플레이어가 Item 을 획득하여 우측 상단에 Slider 가 채워진 모습>

6. 참고문헌

- ** 플레이어와 Item이 부딪쳤을 때, Item이 사라지도록 설정하는 방법*

  https://unity3d.com/kr/learn/tutorials/projects/roll-ball-tutorial/displaying-score-and-text?playlist=17141

- *** 플레이어가 획득한 아이템의 개수를 Slider로 나타내는 방법*

  https://unity3d.com/kr/learn/tutorials/projects/survival-shooter/player-health?playlist=17144