두더지 게임 만들기

- 새 프로젝트 생성
  - 스마트 캠퍼스에서 Texture, Sound 패키지 파일을 다운로드 하여, Import 합니다.
- 두더지 구멍 스크립트 C# 생성하기(hole)

public enum MoleState {

None, Open, Idle, Close, Catch

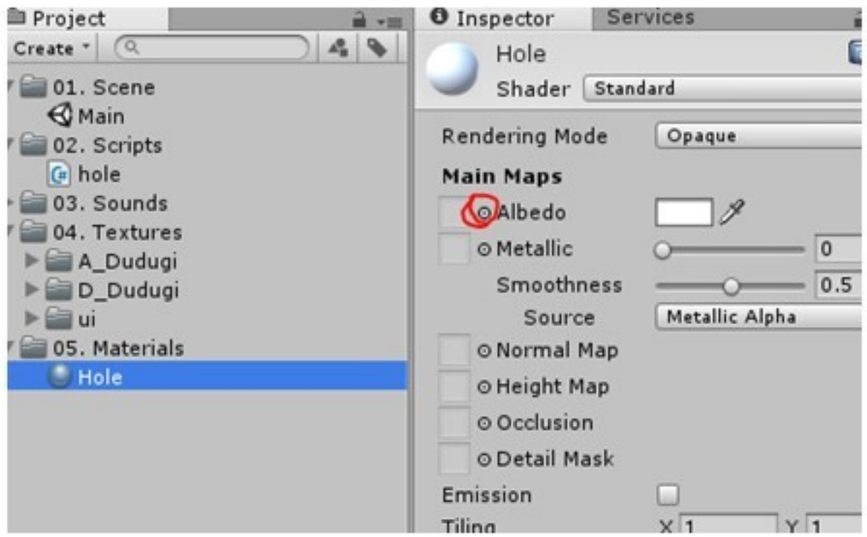
}

public class hole : MonoBehaviour {

public MoleState MS;

두더지 게임 만들기

- GameObject > Create Other > Quad
  - Quad의 이름을 Hole로 변경하고 Project의 hole스크립트를 연결한다.
- Hole 매터리얼 만들기
  - Hole 매터리얼에 Texture 폴더 안의 두더지 이미지 중 한 장을 선택한다.
  - Hole 매터리얼을 선택하고 오른쪽 그림의 아이콘을 선택한다.
  - Hole 매터리얼을 Hierarchy에 있는 Hole 오브젝트에 연결한다.
  - Hierarchy에 있는 Hole을 선택하고 Inspector에서 매터리얼 컴포넌트를 선택하여 Shader를 Unlit/Transparent 로 선택한다.

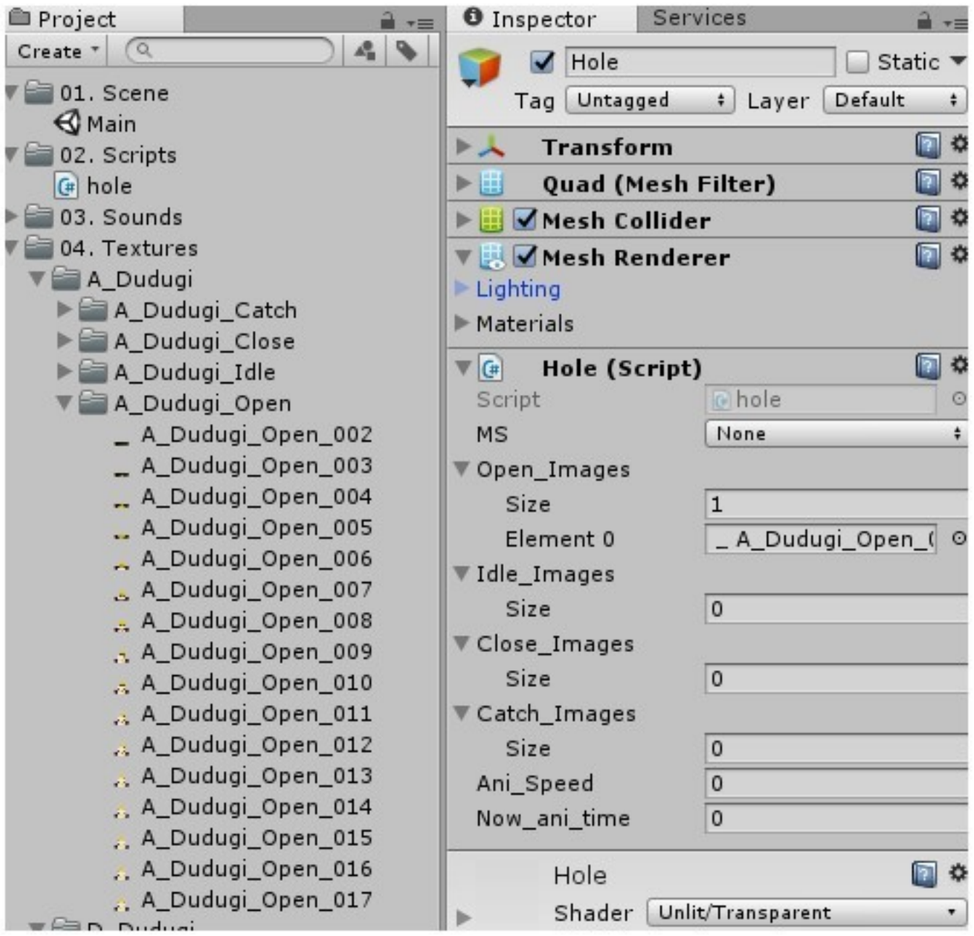


두더지 게임 만들기

- Main Camera 설정하기
  - Position : 0, 0, -10 / Projection : Orthographic /
    Size : 1.5
- hole 스크립트에 다음의 문구 추가하기

```
public Texture[ ] Open_Images;
public Texture[ ] Idle_Images;
public Texture[ ] Close_Images;
public Texture[ ] Catch_Images;
public float Ani_Speed;
public float _now_ani_time;
int Ani_count;
```

두더지 게임 만들기



- Open, Idle, Close, Catch_Images에 변수를 설정한다.
  - Project에 Import 했던 Image를 연결한다.
  - 연결하기 전에 Inspector에 자물쇠 잠금표시를 설정한다.
  - 자물쇠 잠금표시 설정 후 Hole스크립트에 연결하고자 하는 이미지를 Shift를 이용해 전체 선택하여 연결한다.
  - Catch, Close, Idle, Open 에 각각에 맞는 이미지를 연결한다.

두더지 게임 만들기

- hole 스크립트에 다음의 문구 추가하기

```
void Update (){
    if (MS == MoleState.Open) {
        Open_Ing ();
}
if (MS == MoleState.Idle) {
    Idle_Ing ();
}
if (MS == MoleState.Close) {
    Close_Ing ();
}
if (MS == MoleState.Catch) {
    Catch_Ing ();   }    }
```

```
public void Open_On(){
        MS = MoleState.Open;
        Ani_count = 0;
    }
    public void Open_Ing(){
      GetComponent<Renderer>().material.mainTex
ture = Open_Images [Ani_count];
        Ani_count += 1;
        if (Ani_count >= Open_Images.Length) {
            Idle_On ();
        }   }
```

두더지 게임 만들기

```
public void Idle_On(){
        MS = MoleState.Idle;
        Ani_count = 0;
    }

    public void Idle_Ing(){
        GetComponent<Renderer> ().material.mainTexture
= Idle_Images [Ani_count];
        Ani_count += 1;
        if (Ani_count >= Idle_Images.Length) {
            Close_On ();
        }
    }
```

```
public void Close_On(){
        MS = MoleState.Close;
        Ani_count = 0;
    }

    public void Close_Ing(){
        GetComponent<Renderer> ().material.mainTexture
= Close_Images [Ani_count];
        Ani_count += 1;
        if (Ani_count >= Close_Images.Length) {
            MS = MoleState.None;
            Ani_count = 0;
        }
    }
```

두더지 게임 만들기

```
public void Catch_On(){
        MS = MoleState.Catch;
        Ani_count = 0;
    }
    public void Catch_Ing(){
      GetComponent<Renderer> ().material.mainTexture
= Catch_Images [Ani_count];
        Ani_count += 1;
        if (Ani_count >= Catch_Images.Length) {
            MS = MoleState.None;
            Ani_count = 0;
        }
    }
```

두더지 게임 만들기

- hole 스크립트에 문구 수정 및 추가하기

```
void Update (){
 if (_now_ani_time>=Ani_Speed) {
    if (MS == MoleState.Open) {
        Open_Ing ();        }
    if (MS == MoleState.Idle) {
        Idle_Ing ();        }
    if (MS == MoleState.Close) {
        Close_Ing ();       }
    if (MS == MoleState.Catch) {
        Catch_Ing ();           }
     _now_ani_time=0;
} else { _now_ani_time += Time.deltaTime; }
```

```
public void Close_Ing(){
    GetComponent<Renderer> ().material.mainTexture =
Close_Images [Ani_count];
     Ani_count += 1;
     if (Ani_count >= Close_Images.Length) {
        StartCoroutine("Wait");      }  }


 public void Catch_Ing(){
    GetComponent<Renderer> ().material.mainTexture =
Catch_Images [Ani_count];
     Ani_count += 1;
     if (Ani_count >= Catch_Images.Length) {
        StartCoroutine("Wait");      }   }
```

두더지 게임 만들기

- hole 스크립트에 문구 수정 및 추가하기

```
public IEnumerator Wait () {
    MS = MoleState.None;
    Ani_count = 0;
    float wait_Time = Random.Range(0.5f, 4.5f);
    yield return new WaitForSeconds (wait_Time);
    Open_On ();
}
```

두더지 잡기와 효과음 넣기

- hole 스크립트에 다음의 문구를 추가하기

```
public void OnMouseDown () {
    if(MS == MoleState.Idle || MS == MoleState.Open){
        Catch_On();
} }
```

- 게임을 플레이하여 두더지를 터치해 터치가 잘 이루어지는지 확인하기

두더지 잡기와 효과음 넣기

- Quad에 AudioSource 컴포넌트 추가하기
- AudioSource를 추가한 후 Hole 스크립트 수정하기

int Ani_count; ← *이 스크립트 아래에 추가하기*

public AudioClip Open_Sound;

public AudioClip Catch_Sound;

---

public void Open_On(){

MS = MoleState.Open;

Ani_count = 0; ← *이 스크립트 아래에 추가하기*

GetComponent<AudioSource>().clip = Open__Sound;

GetComponent<AudioSource>().Play();

}

---

public void Catch Open_On(){

MS = MoleState.Catch;

Ani_count = 0; ← *이 스크립트 아래에 추가하기*

GetComponent<AudioSource>().clip = Catch_Sound;

GetComponent<AudioSource>().Play();

}

착한두더지와 나쁜두더지 만들기

이번 챕터 주요점

1. 두더지가 Open 되는 순간 랜덤함수를 이용해서 착한두더지가 나올지 나쁜두더지가 나올지 결정한다.
2. 지금까지 만들었던 애니메이션 함수 내부에 각 두더지의 종류에 따라서 다른 종류의 애니메이션을 재생하도록 변경한다.
3. Catch_On 상황에서 미리 나쁜두더지인지 착한두더지인지에 따라서 다른 명령을 전달할 수 있도록 구분한다.

착한두더지와 나쁜두더지 만들기

public Texture[] Catch_Images; ← 이 스크립트 아래에 작성하기

public bool GoodMole;

public int PerGood = 15;

public Texture[ ] Open_Images_2;

public Texture[ ] Idle_Images_2;

public Texture[ ] Close_Images_2;

public Texture[ ] Catch_Images_2;

착한두더지와 나쁜두더지 만들기

변수가 선언되었으니 Open_On() 함수로 가서 스크립트를 변경한다.

```
public void Open_On(){
    MS = MoleState.Open;
    Ani_count = 0;
GetComponent<AudioSource>().clip = Open__Sound;
GetComponent<AudioSource>().Play(); ← 이 스크립트 아래에 추가하기
int a = Random.Range(0, 100);
if (a <=PerGood){
            GoodMole = true;
} else {
            GoodMole = false; }   }
```

착한두더지와 나쁜두더지 만들기

두더지의 종류에 따라 애니메이션이 다르게 재생되는 함수 추가하기

public void Open_Ing(){ ← 이 함수를 아래와 같이 변경한다.

GetComponent<Renderer> ().material.mainTexture = Open_Images [Ani_count];

```
public void Open_Ing(){
    if(GoodMole == false){  GetComponent<Renderer>  ().material.mainTexture
                                = Open_Images [Ani_count];
} else {
   GetComponent<Renderer>  ().material.mainTexture = Open_Images_2 [Ani_count]; }
```

착한두더지와 나쁜두더지 만들기

두더지의 종류에 따라 애니메이션이 다르게 재생되는 함수 추가하기

public void Idle_Ing(){ ← 이 함수를 아래와 같이 변경한다.

GetComponent<Renderer> ().material.mainTexture = Idle_Images [Ani_count];

```
public void Idle_Ing(){
    if(GoodMole == false){  GetComponent<Renderer> ().material.mainTexture
                                = Idle_Images [Ani_count];
} else {
   GetComponent<Renderer> ().material.mainTexture = Idle_Images_2 [Ani_count]; }
```

착한두더지와 나쁜두더지 만들기

두더지의 종류에 따라 애니메이션이 다르게 재생되는 함수 추가하기

public void Close_Ing(){ ← *이 함수를 아래와 같이 변경한다.*

GetComponent<Renderer> ().material.mainTexture = Close_Images [Ani_count];

```
public void Close_Ing(){
    if(GoodMole == false){  GetComponent<Renderer> ().material.mainTexture
                                = Close_Images [Ani_count];
} else {
   GetComponent<Renderer> ().material.mainTexture = Close_Images_2 [Ani_count]; }
```

착한두더지와 나쁜두더지 만들기

두더지의 종류에 따라 애니메이션이 다르게 재생되는 함수 추가하기

public void Catch_Ing(){ ← *이 함수를 아래와 같이 변경한다.*

GetComponent<Renderer> ().material.mainTexture = Catch_Images [Ani_count];

```
public void Catch_Ing(){
    if(GoodMole == false){  GetComponent<Renderer> ().material.mainTexture
                                = Catch_Images [Ani_count];
} else {
   GetComponent<Renderer> ().material.mainTexture = Catch_Images_2 [Ani_count]; }
```

게임 매니저 만들기

하이어라키에 빈오브젝트를 생성하고 이름을 GameManager로 변경 후 프로젝트 뷰에서 아래의 GameManager 스크립트를 만들고 빈오브젝트로 생성된 GameManager에 연결한다.

```
using UnityEngine.UI;
public enum GameState { Ready, Play, End } ← 클래스명 위에 작성한다.
public class GameManager : MonoBehaviour { ← 클래스명이니 작성하지 않는다.
    public GameState GS;
    public float LimitTime;
    public GUIText _TimeText;

    public int iCount_Bad;
    public int iCount_Good;
    public int iScore = 0;

    public GameObject PlayUI;
    public GameObject FinalUI;
    public GUIText _ScoreText;
    public GUIText _FinalScoretxt;
    public GUIText _BadMoletxt;
    public GUIText _GoodMoletxt;
    public AudioClip sndWin;
```

게임 매니저 만들기

```
void Update () {
        if (GS == GameState.Play) {
                LimitTime -= Time.deltaTime;
                if (LimitTime <= 0) {
                LimitTime = 0;
                End ();
                }
        }
        _TimeText.text = "Time: " +string.Format ("{0:N1}", LimitTime);
        iScore = (iCount_Bad * 100) - (iCount_Good * 1000 * 1);
        _ScoreText.text = "Score: " + iScore.ToString ();
}
```

게임 매니저 만들기

```
public void GO() {
        GS = GameState.Play;
}
public void End() {
        GS = GameState.End;
        PlayUI.SetActive (false);
        FinalUI.SetActive (true);
        AudioSource.PlayClipAtPoint (sndWin, transform.position);
        _FinalScoretxt.text = iScore.ToString ();
        _BadMoletxt.text = iCount_Bad.ToString ();
        _GoodMoletxt.text = iCount_Good.ToString ();
}
```

Hole 스크립트 수정하기

- Hole 스크립트에 다음의 구문을 추가한다.

```
public GameManager GM;
void Start ( ) {
        GM.GO ( );
        if (GM.GS == GameState.Play) {
                Open_On ( );
        }
}
```

화면 GUI 구성하기

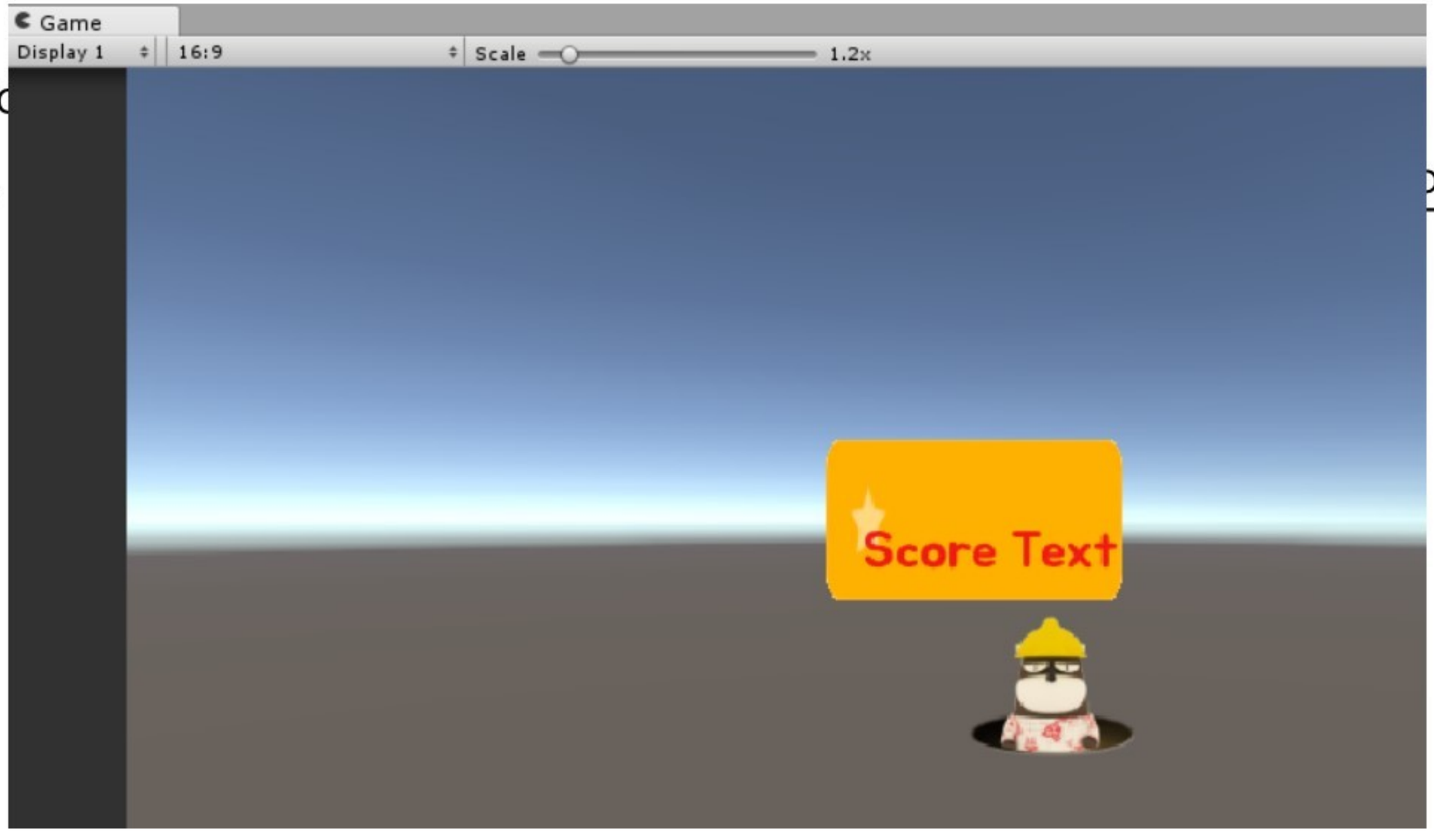
- Scoretxt와 Timetxt 배치하기
  - 빈 오브젝트를 생성하고, 빈 오브젝트의 이름을 Scoretxt로 변경한다.
  - Scoretxt 오브젝트에 GUI Text 컴포넌트를 추가하고 Text에 Score Text 라고 작성한다.
  - Text의 크기나 정렬, 폰트 등은 자신의 게임 해상도에 맞게 설정한다.
  - Scoretxt가 표시되는 위치도 자신의 게임 해상도에 맞게 설정한다.
  - 위와 같은 방법으로 Timetxt 오브젝트도 만들어 둔다.

화면 GUI 구성하기

- Score_img 배치하기
  - Cube를 추가하여 이름을 Score_img로 변경하고, Score 이미지를 연결한다. 그리고 이전에 만들어 둔 Scoretxt 오브젝트 아래로 배치하여 Score Text가 자신의 게임 해상도에 맞게끔 위치 조정을 한다.

화면 GUI 구성하기

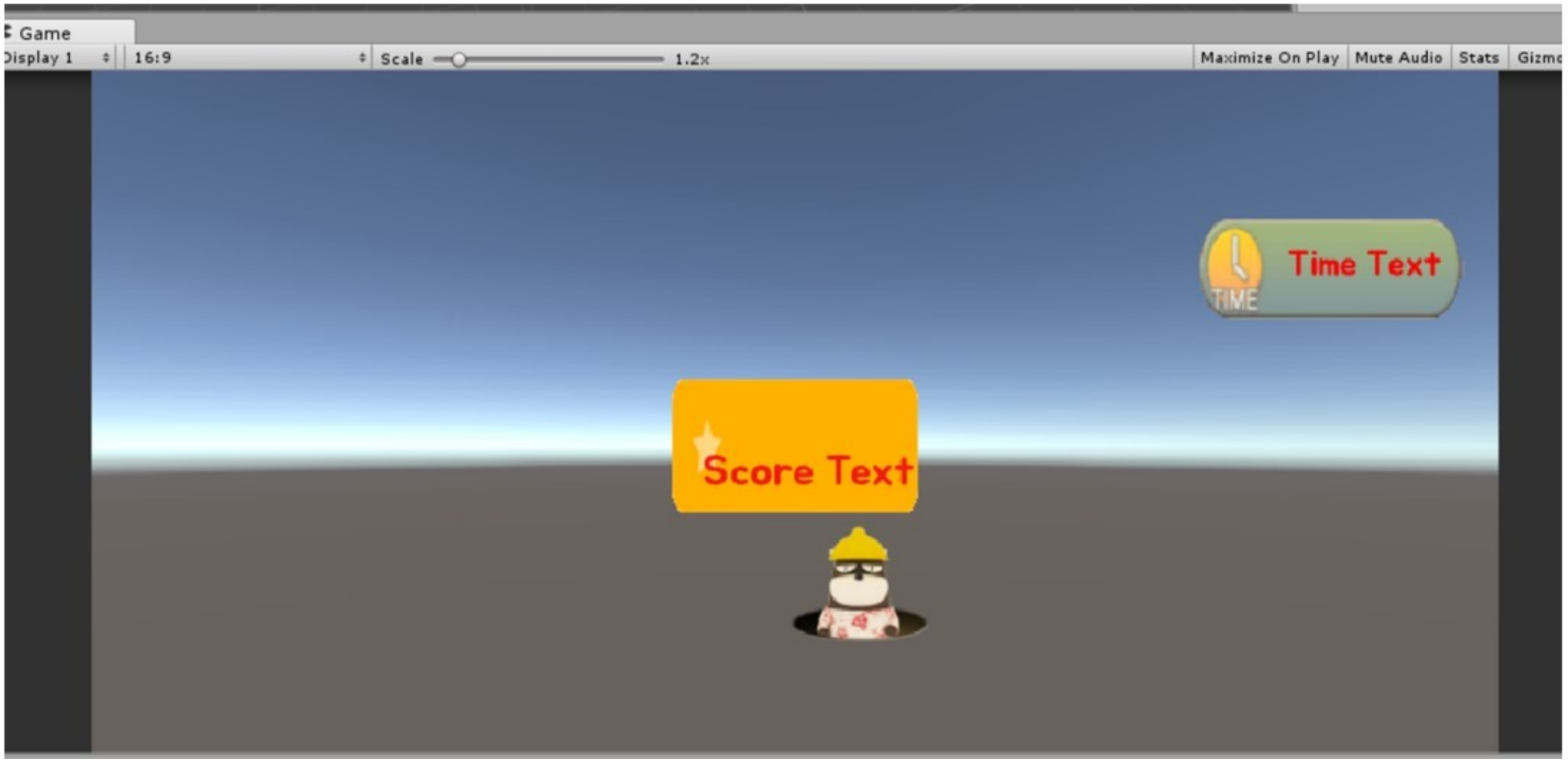
- Sc
  - 연결



화면 GUI 구성하기

- Time_img 배치하기
  - 같은 방법으로 이번에도 Cube를 추가하고 이름을 Time_img로 변경하여, 이전에 만든 Timetxt 오브젝트 아래로 배치하고 Time Text가 자신의 해상도에 맞게끔 위치 조정을 한다.

화면 GUI 구성하기


Game
Display 1
16:9
Scale
1.2x
Maximize On Play
Mute Audio
Stats
TIME
Time Text
Score Text

화면 GUI 구성하기

- 완성된 Timetxt와 Scoretxt를 GameManager 오브젝트에 설정한 Time Text, Score Text 변수 창에 연결한다.

- 빈 오브젝트를 하나 생성하고 이름을 PlayUI로 지정한 뒤, PlayUI 오브젝트의 하위 오브젝트로 Hole, Scoretxt, Timetxt, Score_img, Time_img를 설정한다.

화면 END GUI 구성하기

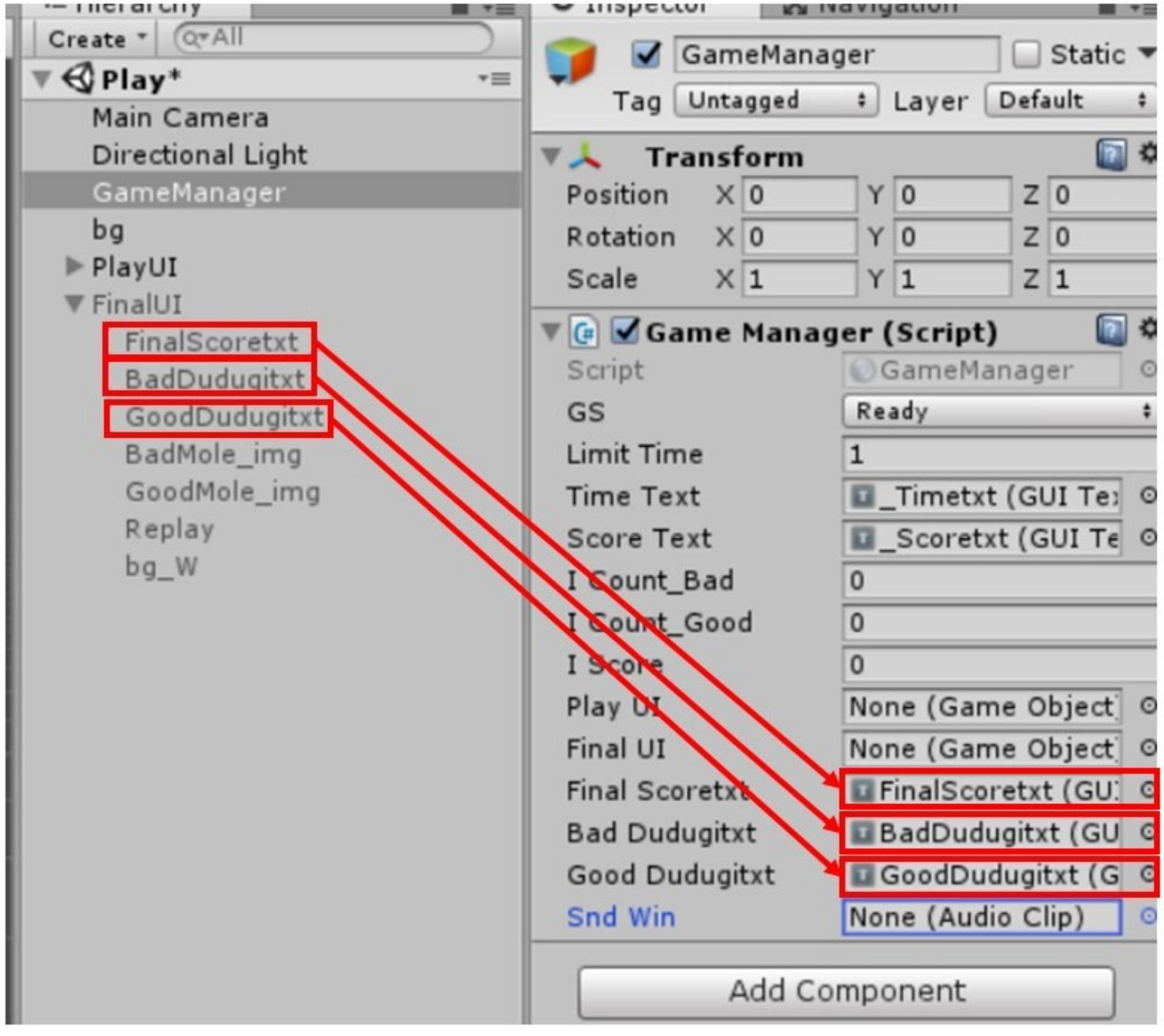
- 앞서 설명한 Scoretxt, Timetxt를 만드는 방법과 동일하게 BadDudugitxt와 GoodDudugitxt, FinalScoretxt를 만든다.

- 다음으로 앞서 Score_img, Time_img 오브젝트를 만든 것과 동일하게 Bad_Dudugi_BG, Good_Dudugi_BG 이미지를 이용하여 Bad_Dudugi_img, Good_Dudugi_img 오브젝트를 만든다.

화면 END GUI 구성하기

- 5개의 오브젝트를 모두 완성했다면 빈오브젝트를 하나 생성하고 이름을 FinalUI로 변경한 뒤, FinalUI 오브젝트의 하위 오브젝트로 BadDudugitxt, GoodDudugitxt, Bad_Dudugi_img, Good_Dudugi_img, FinalScoretxt 오브젝트를 배치한다.

- 이제 GameManager 오브젝트를 선택하고 방금 완성한 GUIText(BadDudugitxt, GoodDudugitxt, FinalScoretxt)를 변수 설정창에 연결한다.


Create
All
Play*
Main Camera
Directional Light
GameManager
bg
PlayUI
FinalUI
FinalScoretxt
BadDudugitxt
GoodDudugitxt
BadMole_img
GoodMole_img
Replay
bg_W
GameManager
Static
Tag
Untagged
Layer
Default
Transform
Position
Rotation
Scale
Game Manager (Script)
Script
GameManager
GS
Ready
Limit Time
Time Text
_Timetxt (GUI Te
Score Text
_Scoretxt (GUI Te
I Count_Bad
I Count_Good
I Score
Play UI
None (Game Object
Final UI
None (Game Object
Final Scoretxt
FinalScoretxt (GU
Bad Dudugitxt
BadDudugitxt (GU
Good Dudugitxt
GoodDudugitxt (G
Snd Win
None (Audio Clip)
Add Component

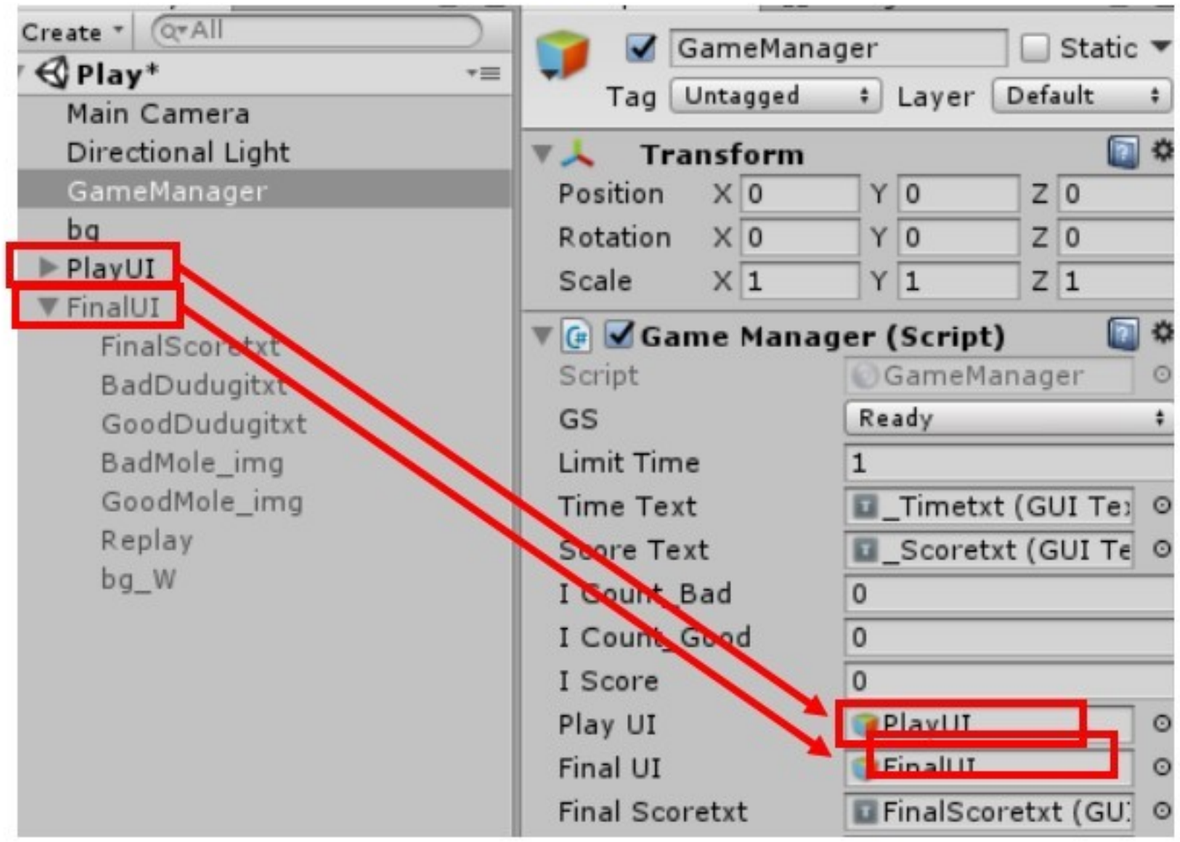
화면 END GUI 구성하기

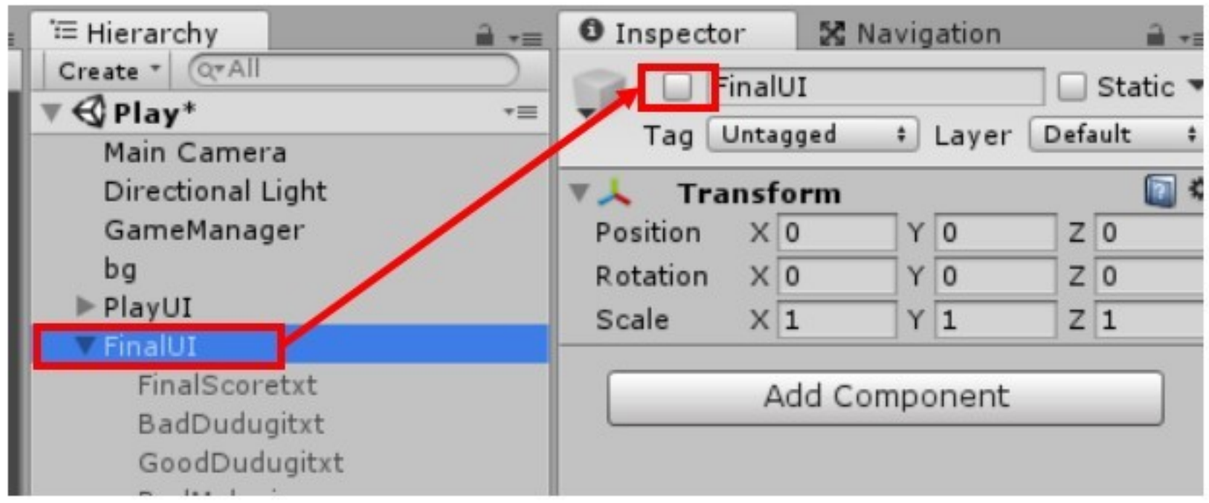
- 다음으로 GameManager 오브젝트를 선택하고 Inspector를 확인한다.

- 이번에는 하이어라키에 만들어 놓은 PlayUI와 FinalUI를 Play UI와 Final UI 설정창에 연결하고, 하이어라키에 있는 FinalUI를 선택하여 Inspector에서 활성화 모드를 체크해제한다.

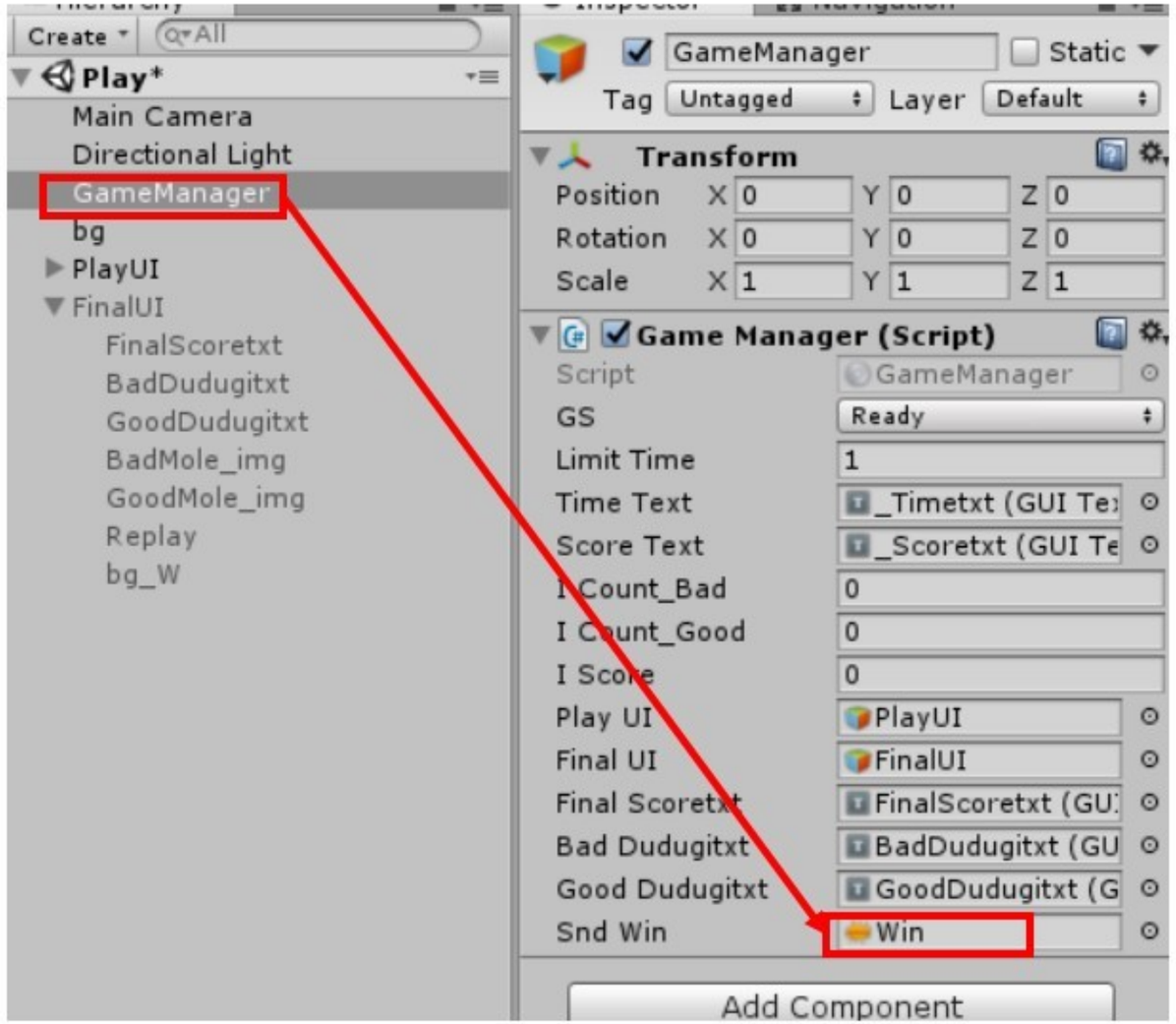
- 마지막으로 GameManager를 선택하고 Snd Win 항목에 Win 사운드를 연결한다.

화면 END GUI 구성하기





화면 END GUI 구성하기



게임 시작버튼 만들기

- 게임을 시작하기 위한 버튼을 만드는 스크립트를 작성한다.
- 먼저 새로운 Scene을 하나 생성하고 해당 Scene의 이름을 Title로 설정한다.
- 다음으로 GameStart 스크립트를 만들고 다음의 코드를 작성한다.

```
using UnityEngine.SceneManagement;

public class GameStart : MonoBehaviour {

public AudioClip sndReady;
public AudioClip sndGo;
```

게임 시작버튼 만들기

```
void Start (){
        AudioSource.PlayClipAtPoint (sndReady, transform.position);
}

public void OnMouseDown(){
        SceneManager.LoadScene ("Play");

        AudioSource.PlayClipAtPoint (sndGo, transform.position);
        }
```

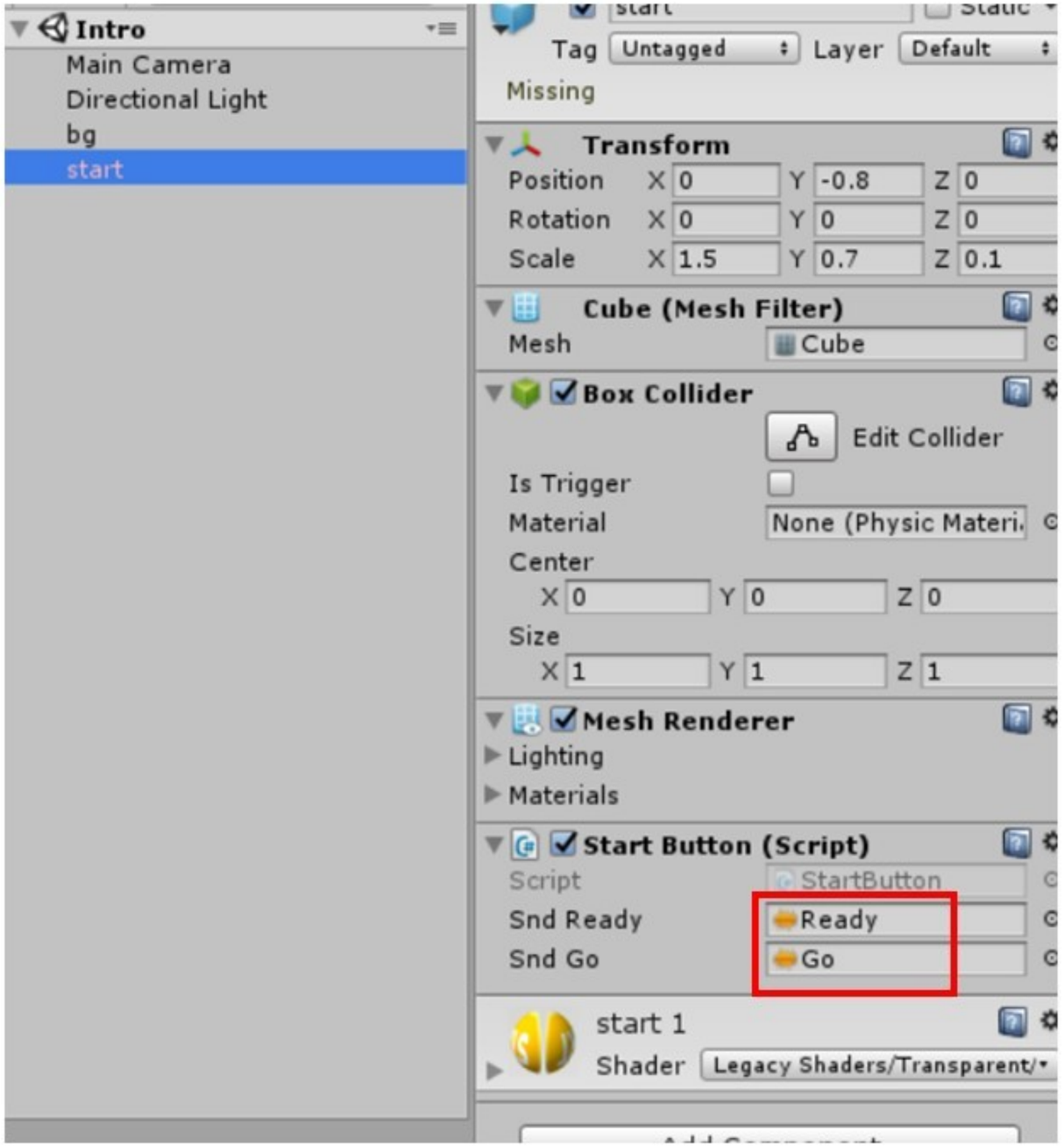
"이곳에 작성될 Scene의 이름은 자신의 게임 메인 화면의 Scene 이름을 작성해 넣는다."

게임 시작버튼 만들기

- Cube나 자신이 원하는 모양의 오브젝트를 만들어 GameStart 스크립트를 연결하고 Snd Ready, Snd GO에 사운드 파일을 연결한다.

- 버튼으로 생성한 오브젝트에 이미지를 추가하여 버튼의 느낌을 살린다.

- 게임 Title 화면에 적절히 버튼을 배치한다.


Intro
Main Camera
Directional Light
bg
start
Tag Untagged
Layer Default
Missing
Transform
Position X 0 Y -0.8 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1.5 Y 0.7 Z 0.1
Cube (Mesh Filter)
Mesh Cube
Box Collider
Edit Collider
Is Trigger
Material None (Physic Materi
Center
X 0 Y 0 Z 0
Size
X 1 Y 1 Z 1
Mesh Renderer
Lighting
Materials
Start Button (Script)
Script StartButton
Snd Ready Ready
Snd Go Go
start 1
Shader Legacy Shaders/Transparent/


자신이 원하는 오브젝트에 이미지를
추가하여 아래 그림처럼 버튼과 같은
느낌을 주도록 한다.
START

게임 재시작버튼 만들기

- 게임을 재시작하기 위한 버튼을 만드는 스크립트를 작성한다.
- 자신의 게임 메인 Scene으로 돌아가서 GameReplay 스크립트를 만든다.

```
using UnityEngine.SceneManagement;

public class GameReplay : MonoBehaviour {
public AudioClip sndReplay;
```

게임 재시작버튼 만들기

```
public void OnMouseDown(){
        SceneManager.LoadScene ("Title");
        AudioSource.PlayClipAtPoint (sndReplay, transform.position);
}
```

"이곳에 작성될 Scene의 이름은 자신의 타이틀 Scene의 이름을 넣는다.

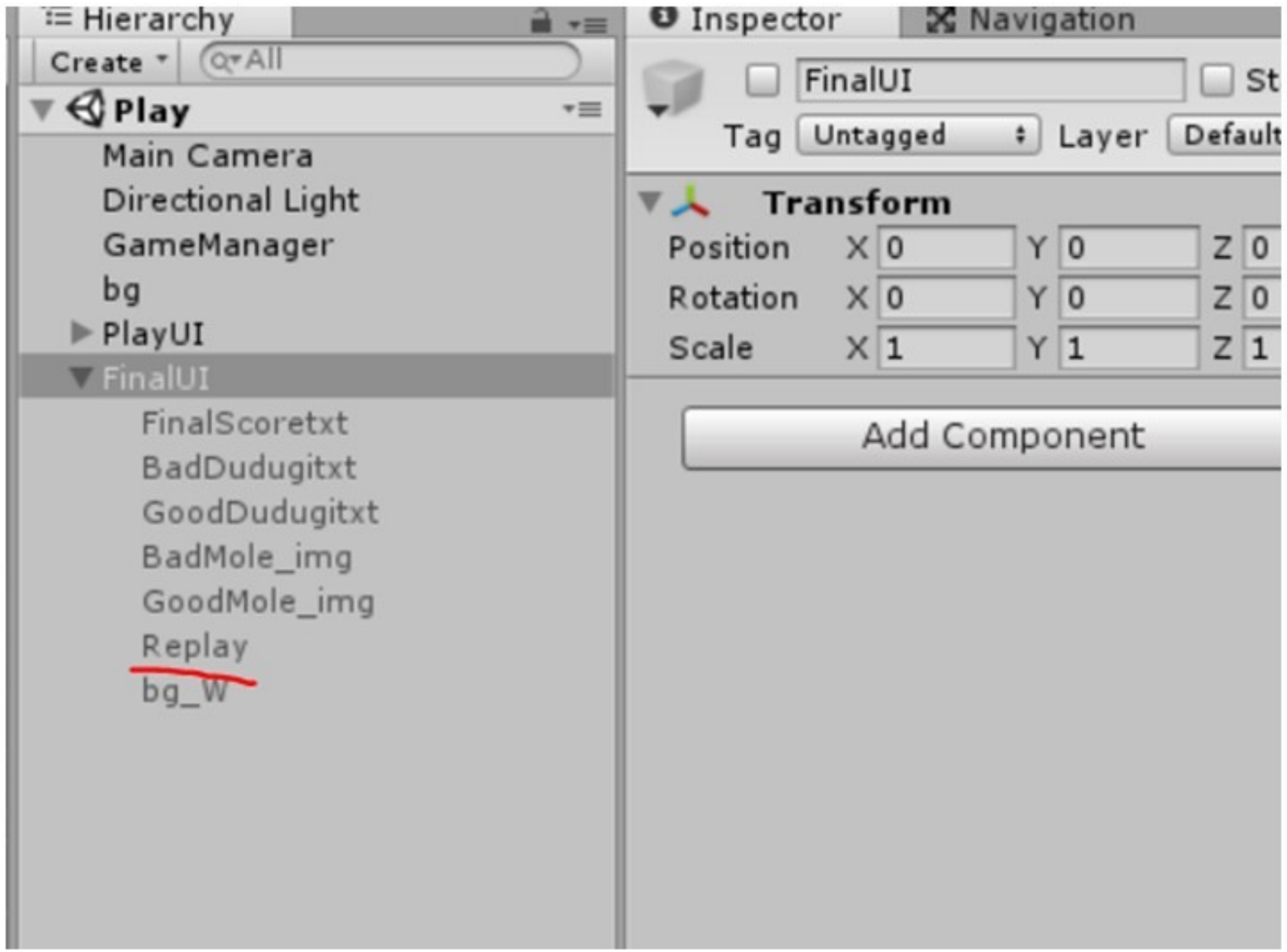
게임 재시작버튼 만들기

- Cube나 자신이 원하는 모양의 오브젝트를 만들어 오브젝트의 이름을 Replay로 설정하고 GameReplay 스크립트를 연결한다.

- Replay 오브젝트를 선택하여 Snd Replay 항목에 사운드 파일을 연결한다.

- Replay 오브젝트를 FinalUI 오브젝트의 하위 오브젝트로 배치한다.

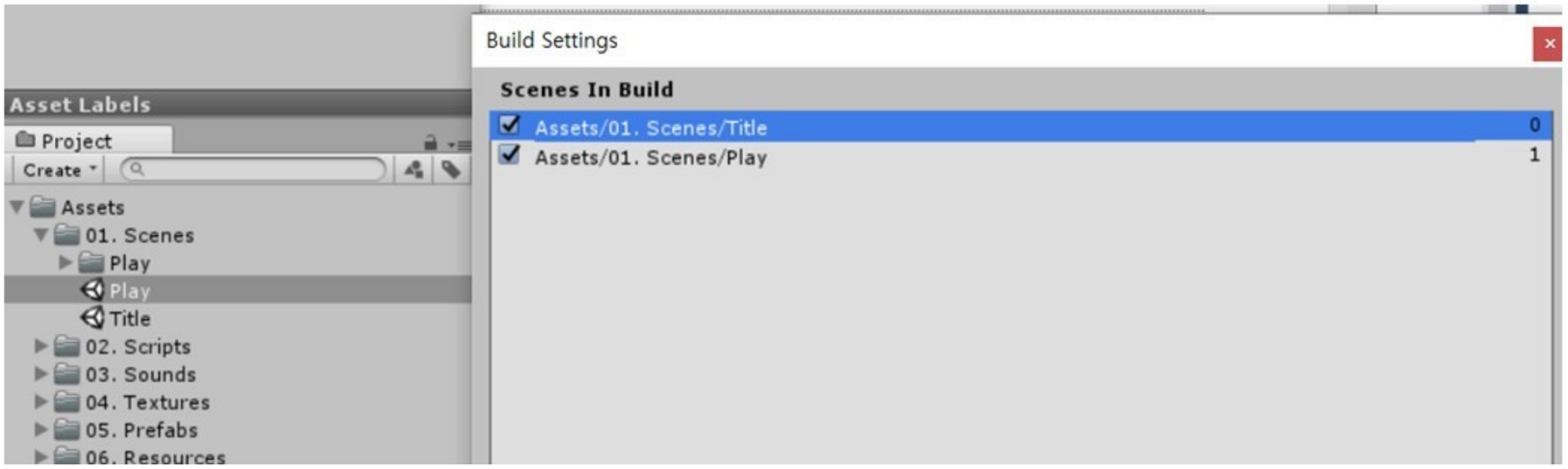
게임 재시작버튼 만들기


Hierarchy
Create
All
Play
Main Camera
Directional Light
GameManager
bg
PlayUI
FinalUI
FinalScoretxt
BadDudugitxt
GoodDudugitxt
BadMole_img
GoodMole_img
Replay
bg_W
Inspector
Navigation
FinalUI
Tag
Untagged
Layer
Transform
Position
Rotation
Scale
Add Component

게임 빌드처리하기

- 완성된 2개의 Scene(Title, Play)을 빌드처리 한다.
- Title을 먼저 배치하고 다음으로 Play Scene을 배치한다.

게임 빌드처리 하기


Build Settings
Scenes In Build
Assets/01. Scenes/Title
0
Assets/01. Scenes/Play
1
Asset Labels
Project
Create
Assets
01. Scenes
Play
Play
Title
02. Scripts
03. Sounds
04. Textures
05. Prefabs
06. Resources