

AR 콘텐츠 제작

(1. 지형 인식을 이용한 자동차 카탈로그)

목차

1. AR 프로젝트 셋팅하기
2. 바닥 지형 인식하기
3. 인식된 바닥에 자동차 모델링
생성하기
4. 자동차 모델링 조작하기

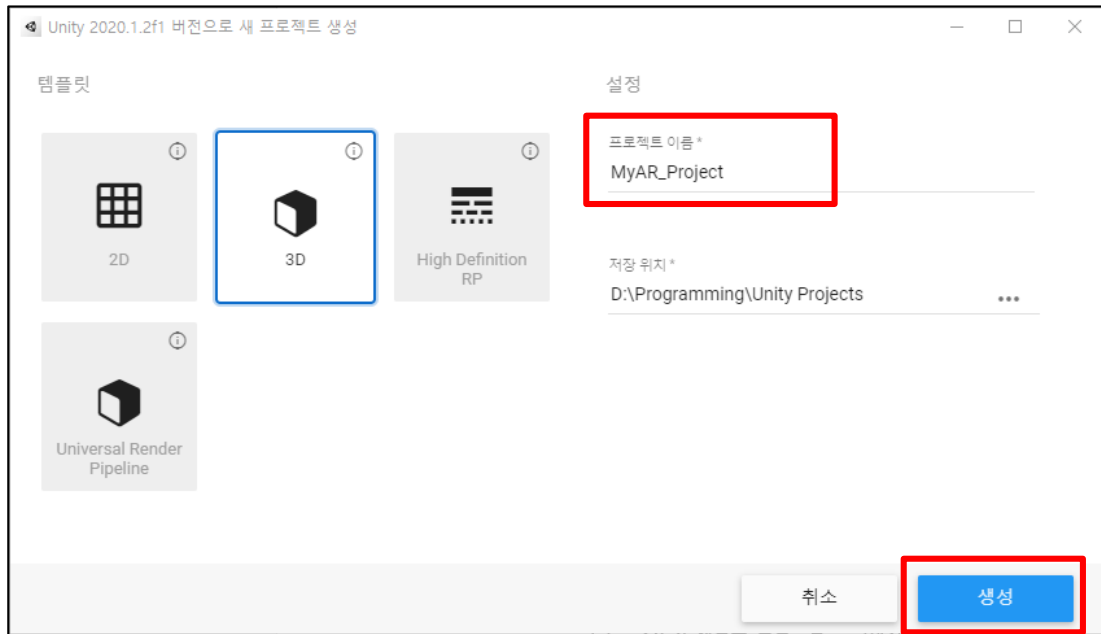
1. AR 프로젝트 셋팅하기

2.0-1 프로젝트 만들기

1. AR 프로젝트 셋팅하기

◆ 프로젝트 생성

- Unity Hub에서 [새로 생성] 버튼을 클릭
- 새로운 프로젝트 생성 창에서 “MyAR_Project”라는 이름으로 새로운 프로젝트를 생성

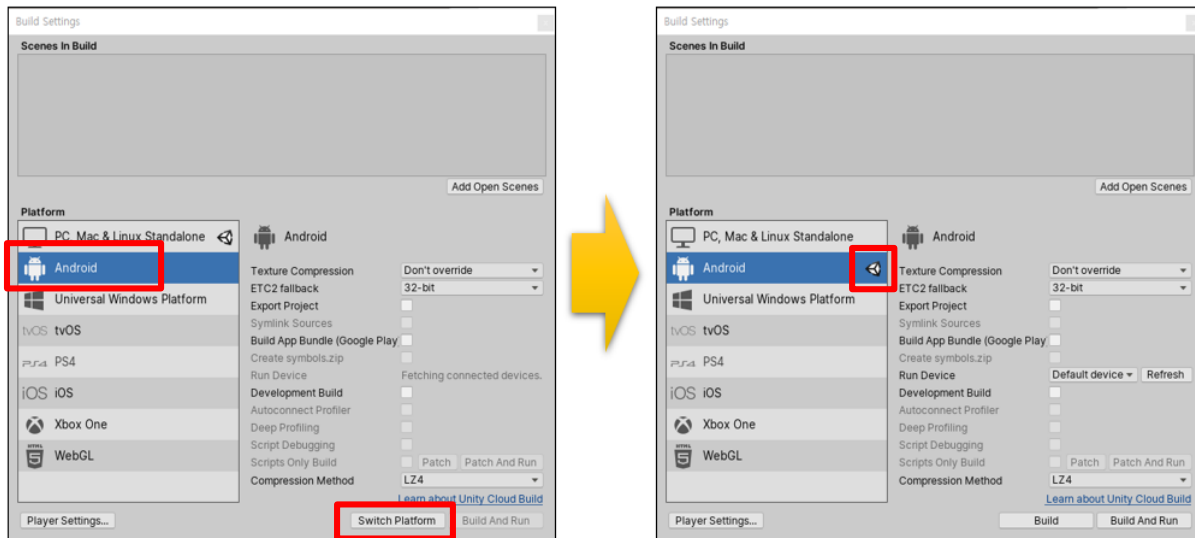


2.0-1 프로젝트 만들기

1. AR 프로젝트 셋팅하기

◆ 타겟 플랫폼의 설정

- 에디터 우측 상단의 [File – Build Settings...]를 선택하여 빌드 셋팅 창을 활성화
- 타겟 플랫폼을 [Android]로 선택하고 [Switch Platform] 버튼을 클릭
- 유니티 마크의 위치가 변경되는 것을 확인

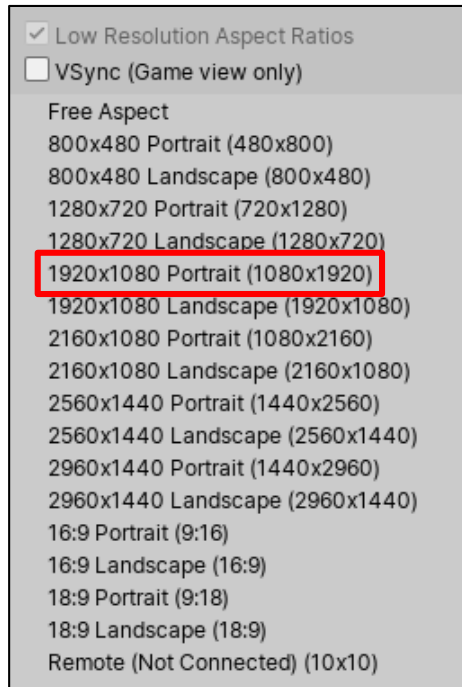


2.0-1 프로젝트 만들기

1. AR 프로젝트 셋팅하기

◆ 해상도 조정하기

- 안드로이드 수직 해상도 비율(9:16)에 맞도록 유니티 게임 뷰의 해상도를 변경
- 게임 뷰 좌측 상단에 있는 해상도 목록에서 1920x1080 Portrait (1080x1920)을 선택

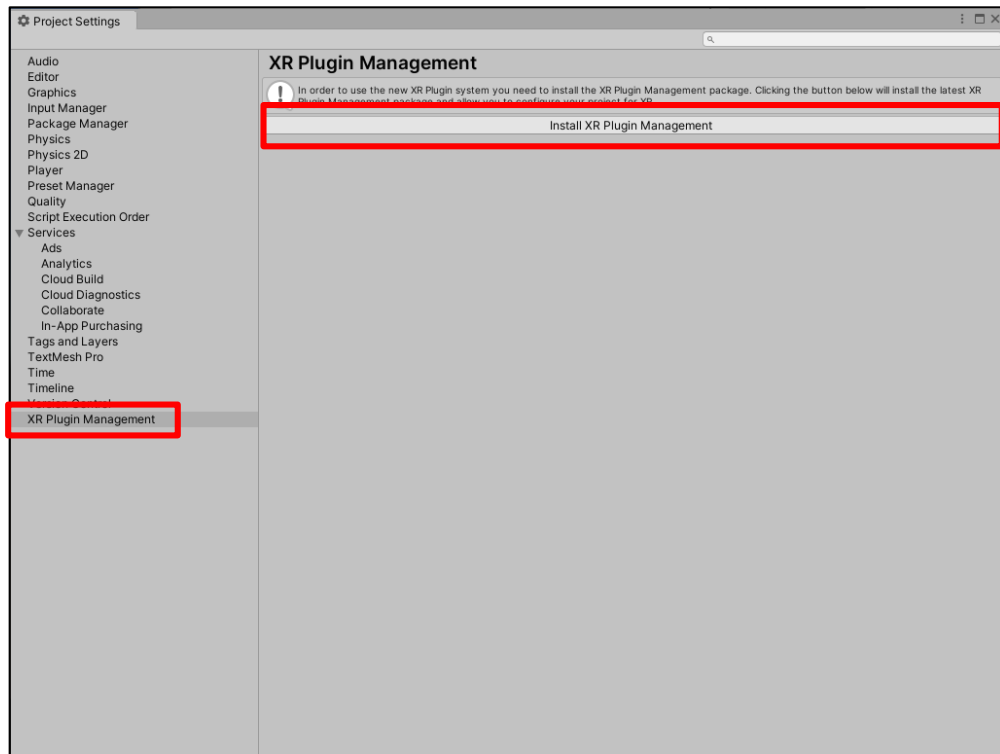


2.0-1 프로젝트 만들기

1. AR 프로젝트 셋팅하기

◆ XR Plugin Management 패키지 설치

- 에디터 상단에서 [Edit – Project Settings...]를 선택하여 프로젝트 셋팅 창을 활성화
- [XR Plugin Management] 탭을 선택하고, [Install XR Plugin Management] 버튼을 클릭하여 XR 플러그인 매니지먼트를 설치

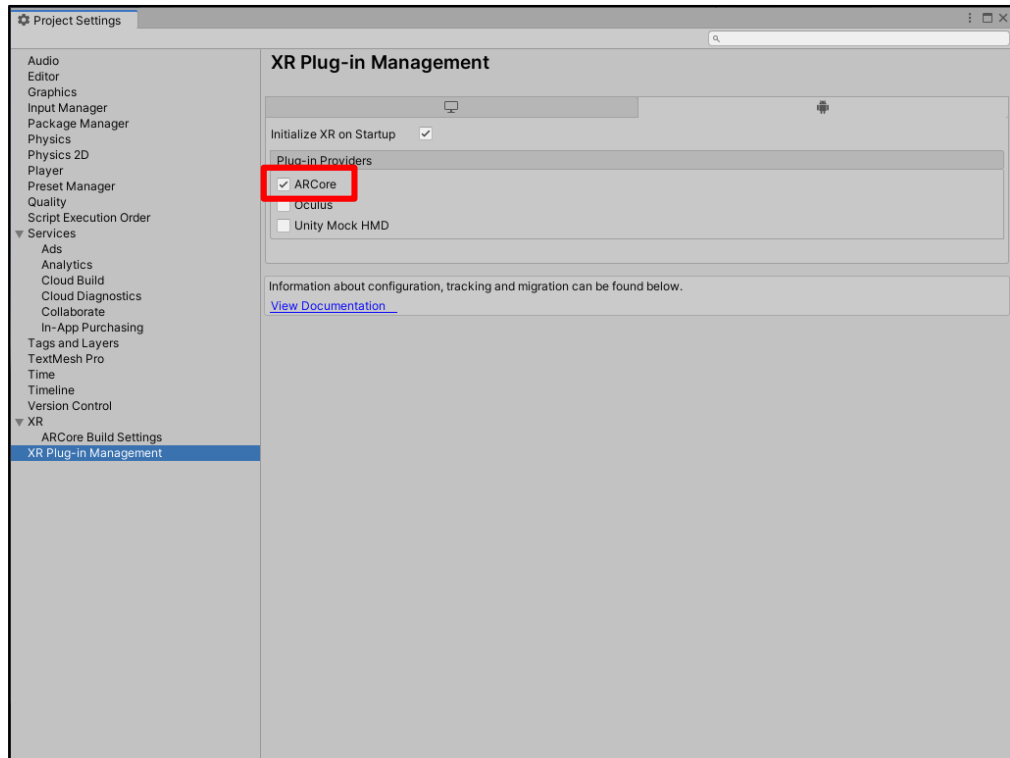


2.0-1 프로젝트 만들기

1. AR 프로젝트 셋팅하기

◆ AR Core 패키지 설치

- XR 플러그인 매니지먼트 설치가 완료되면 Plug-in Providers 아래에 있는 [ARCore] 항목 좌측의 체크 박스를 클릭
- 자동으로 ARCore XR Plugin 패키지가 프로젝트에 설치됩니다

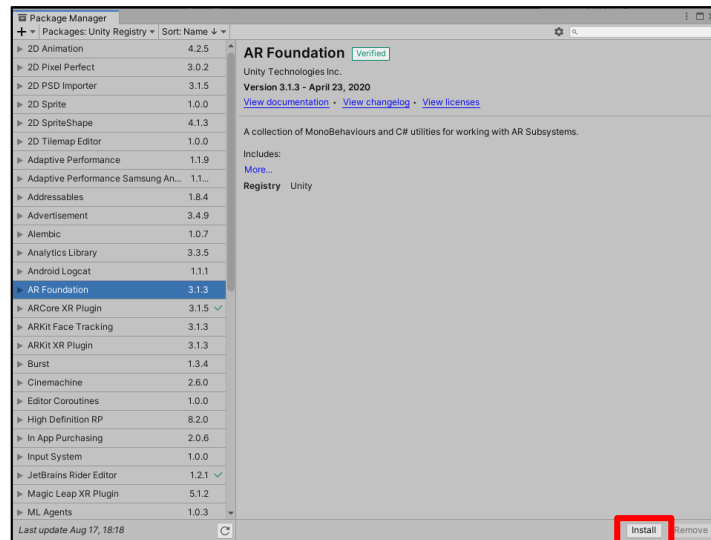
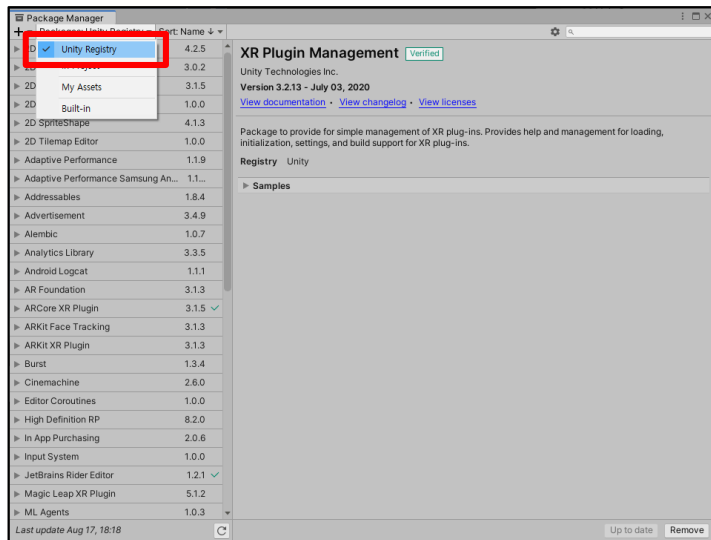


2.0-1 프로젝트 만들기

1. AR 프로젝트 셋팅하기

◆ AR Foundation 패키지 설치

- 유니티 에디터 상단의 [Window – Package Manager]를 선택하여 패키지 매니저 창을 활성화
- 좌측 상단의 [+ 버튼 - Unity Registry]를 선택
- 패키지 리스트에서 AR Foundation을 선택하고 하단의 [Install] 버튼을 클릭해서 AR 파운데이션 패키지를 설치



2. 지형 인식을 이용한 자동차 카탈로그

2.1-1 바닥 지형 인식하기

2. 지형 인식을 이용한 자동차 카탈로그

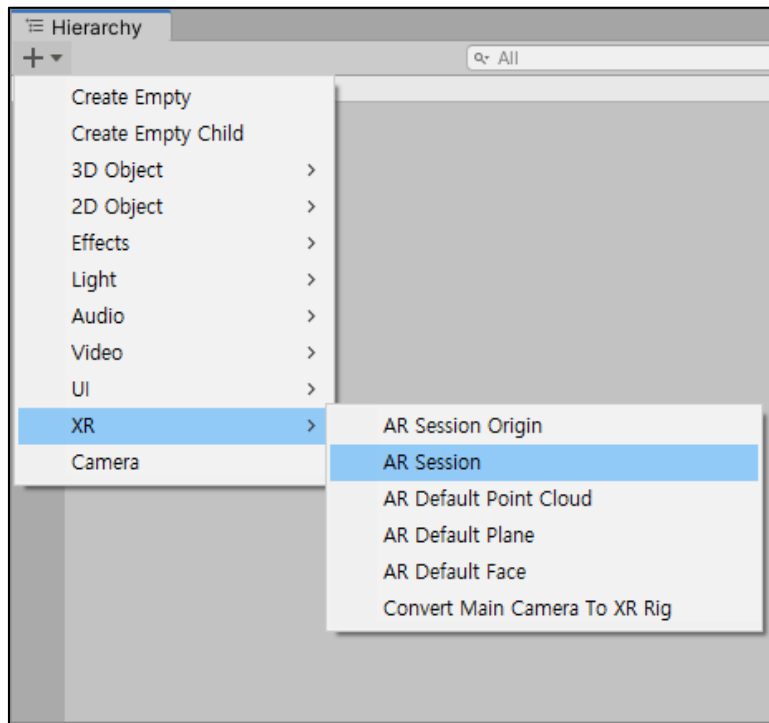
- ◆ 목표 : 스마트폰의 카메라로 찍은 화면에서 바닥을 인식하게 하고 싶다.
- ◆ 순서 :
 1. 메인 카메라로 AR 전용 카메라를 배치한다.
 2. AR 카메라가 인식한 바닥을 시각적으로 표시한다.
 3. 안드로이드 APK 파일로 빌드하여 동작 상태를 확인한다.

2.1-1 바닥 지형 인식하기

2. 지형 인식을 이용한 자동차 카탈로그

◆ 씬 저장 및 AR Session 오브젝트 생성

- 키보드의 [Ctrl] + [N]을 눌러서 새로운 씬을 생성
- [Ctrl] + [S]를 눌러서 “CarScene”이라는 이름으로 씬을 저장
- 하이어라키 뷰에서 [+ 버튼 - XR – AR Session]을 선택하여 AR Session 오브젝트를 생성

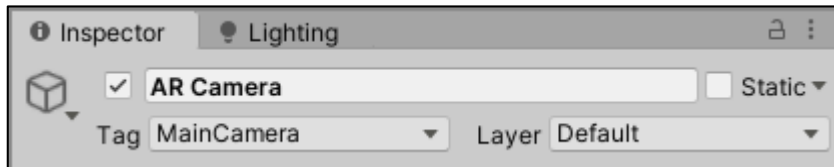
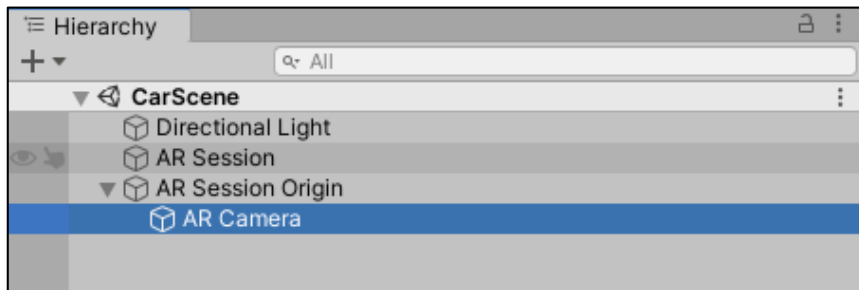


2.1-1 바닥 지형 인식하기

2. 지형 인식을 이용한 자동차 카탈로그

◆ AR 카메라 오브젝트 추가

- 키보드의 [Ctrl] + [N]을 눌러서 새로운 씬을 생성
- [Ctrl] + [S]를 눌러서 “CarScene”이라는 이름으로 씬을 저장
- 하이어라키 뷰에서 [+ 버튼 - XR – AR Session Origin]을 선택하여 AR Session Origin 오브젝트를 생성
- 기존에 씬에 배치되어 있던 Main Camera 오브젝트는 삭제
- AR Session Origin 오브젝트 하위의 AR Camera 오브젝트의 태그를 MainCamera로 변경

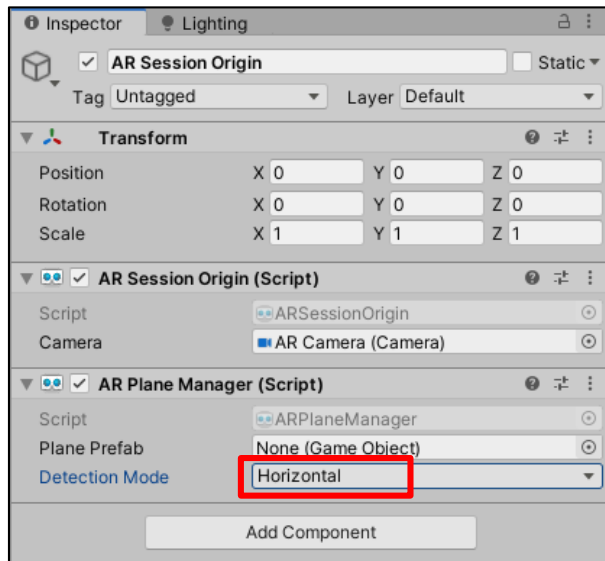
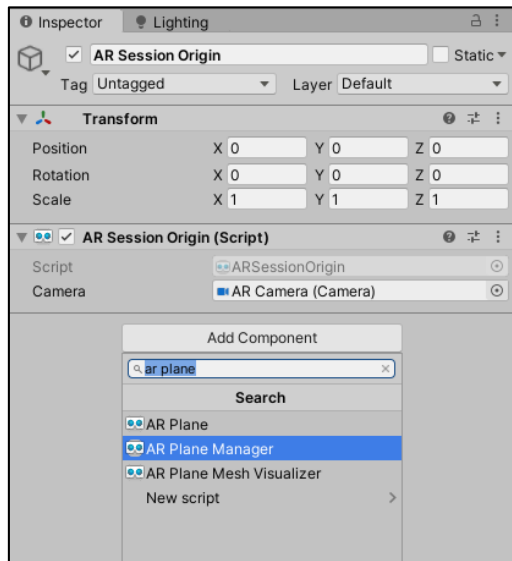


2.1-1 바닥 지형 인식하기

2. 지형 인식을 이용한 자동차 카탈로그

◆ AR Plane Manager 컴포넌트 추가

- AR Session Origin 오브젝트에 [Add Component] 버튼을 누르고 “AR Plane Manager”를 검색하여 추가
- AR Plane Manager 컴포넌트의 Detection Mode를 Everything에서 Horizontal로 변경

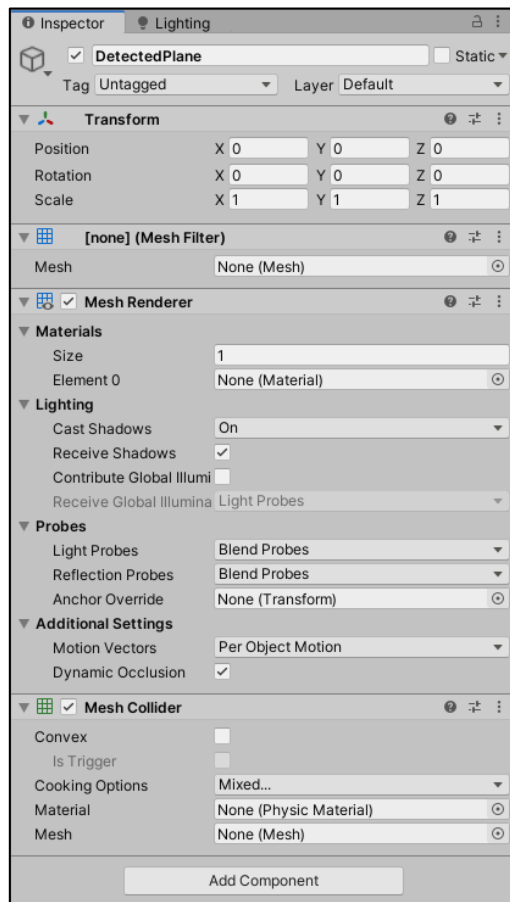


2.1-1 바닥 지형 인식하기

2. 지형 인식을 이용한 자동차 카탈로그

◆ 바닥에 생성될 오브젝트 제작

- 씰 위에 빈 게임 오브젝트를 생성하고 이름을 “DetectedPlane”으로 변경
- 하단의 [Add Component] 버튼을 클릭하고 Mesh 카테고리에 있는 Mesh Filter 컴포넌트와 Mesh Renderer 컴포넌트를 추가
- 이어서 [Add Component – Physics – Mesh Collider]를 선택해서 메쉬 콜라이더 컴포넌트를 추가

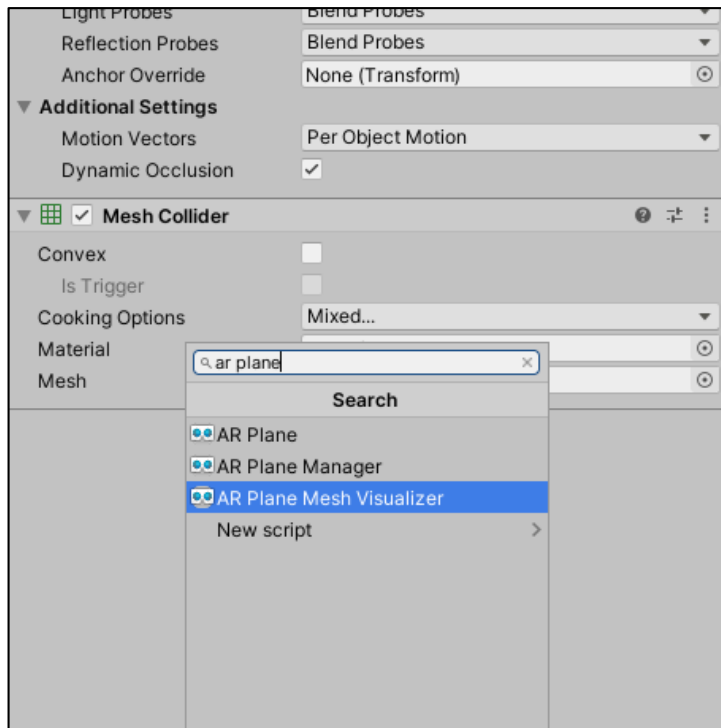


2.1-1 바닥 지형 인식하기

2. 지형 인식을 이용한 자동차 카탈로그

◆ AR 메쉬 생성 관련 컴포넌트 추가

- DetectedPlane 오브젝트에서 [Add Component] 버튼을 클릭하고, 검색 창에 “AR Plane”을 입력하여 검색된 목록에서 AR Plane Mesh Visualizer를 선택
- AR Plane Mesh Visualizer 컴포넌트가 추가되면서 AR Plane 컴포넌트도 자동으로 추가됨

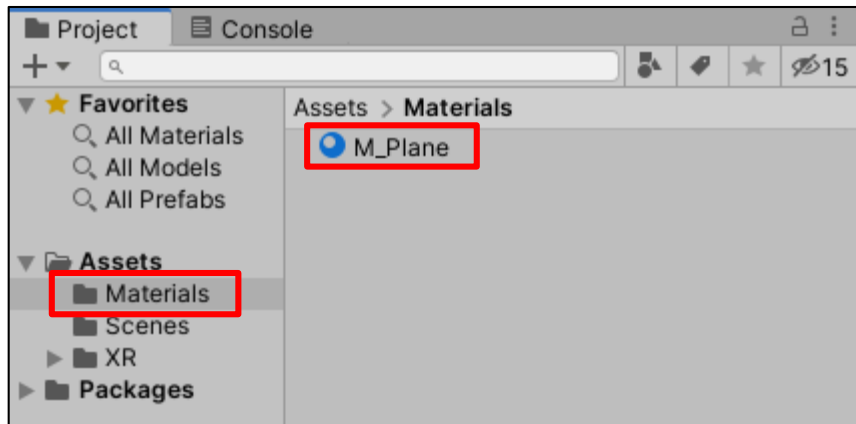


2.1-1 바닥 지형 인식하기

2. 지형 인식을 이용한 자동차 카탈로그

◆ 바닥 재질 만들기

- 프로젝트 뷰에서 [+버튼 – Folder]를 선택하여 Asset폴더 하위에 새로운 폴더를 생성하고 폴더 이름을 “Materials”로 변경
- 새로 만든 Materials 폴더 안에서 [+ 버튼 – Create – Material]을 선택하여 새로운 매터리얼 파일을 생성하고 이름을 “M_Plane”으로 변경

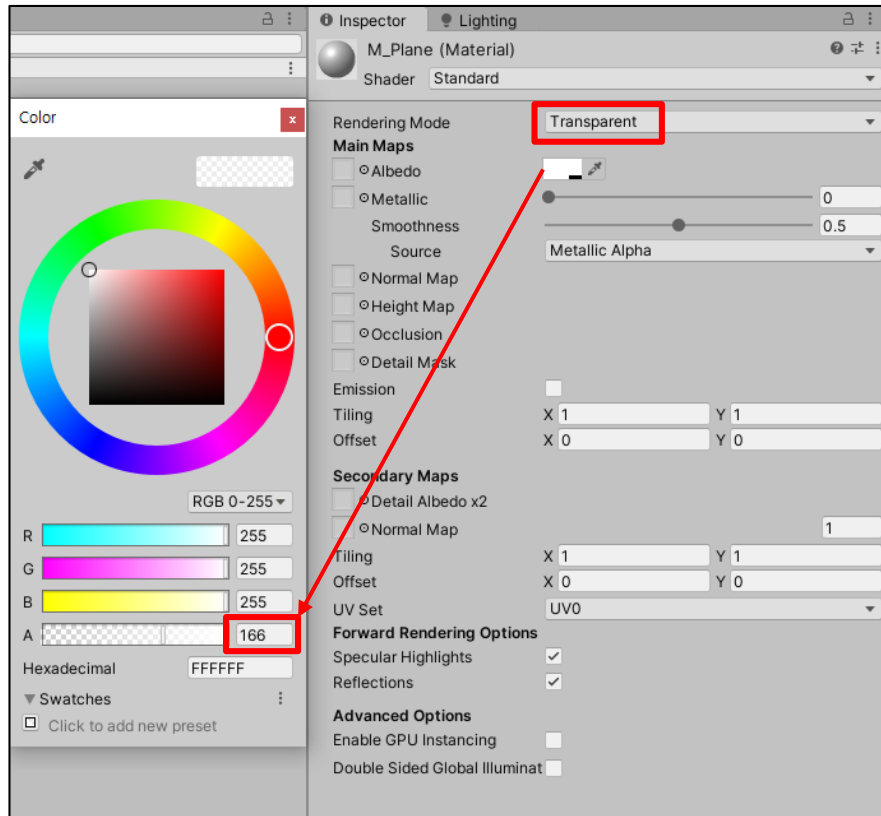


2.1-1 바닥 지형 인식하기

2. 지형 인식을 이용한 자동차 카탈로그

◆ 바닥 재질 만들기

- M_Plane 매터리얼을 반투명한 상태로 만들기 위해 인스펙터 뷰에서 렌더링 모드(Rendering Mode)를 “Transparent”로 변경
- Albedo 우측의 컬러 피커를 클릭해서 A(Alpha – 투명도)의 값을 “166”으로 조정

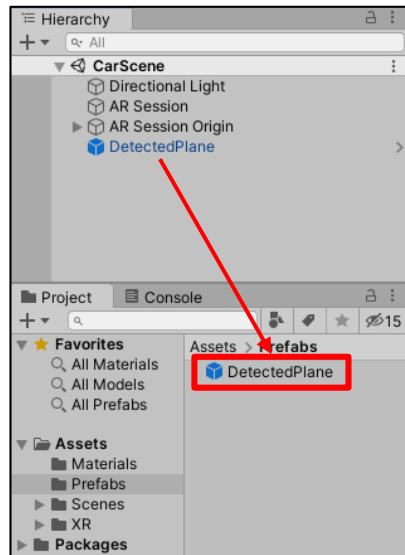
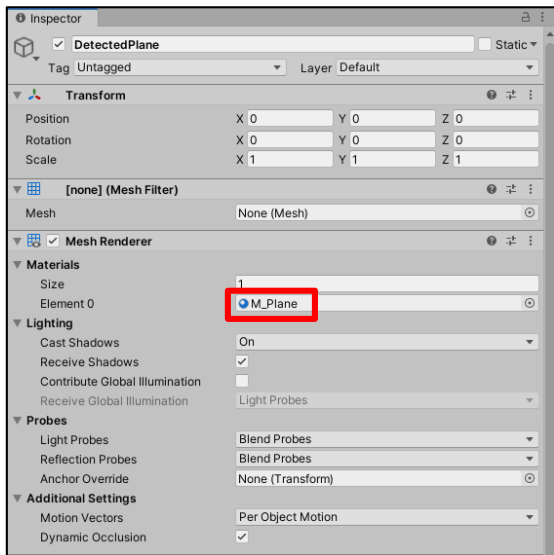


2.1-1 바닥 지형 인식하기

2. 지형 인식을 이용한 자동차 카탈로그

◆ 재질 등록 및 프리팹으로 저장하기

- DetectedPlane 오브젝트를 선택하고 Mesh Renderer 컴포넌트에 있는 Materials 항목에 M_Plane 매тери얼을 추가
- Project 뷰에 “Prefabs”라는 이름으로 새로운 폴더를 생성하고 그 안에 DetectedPlane 오브젝트를 드래그 앤 드롭하여 프리팹 파일로 생성
- 씬에 배치된 DetectedPlane 오브젝트는 삭제

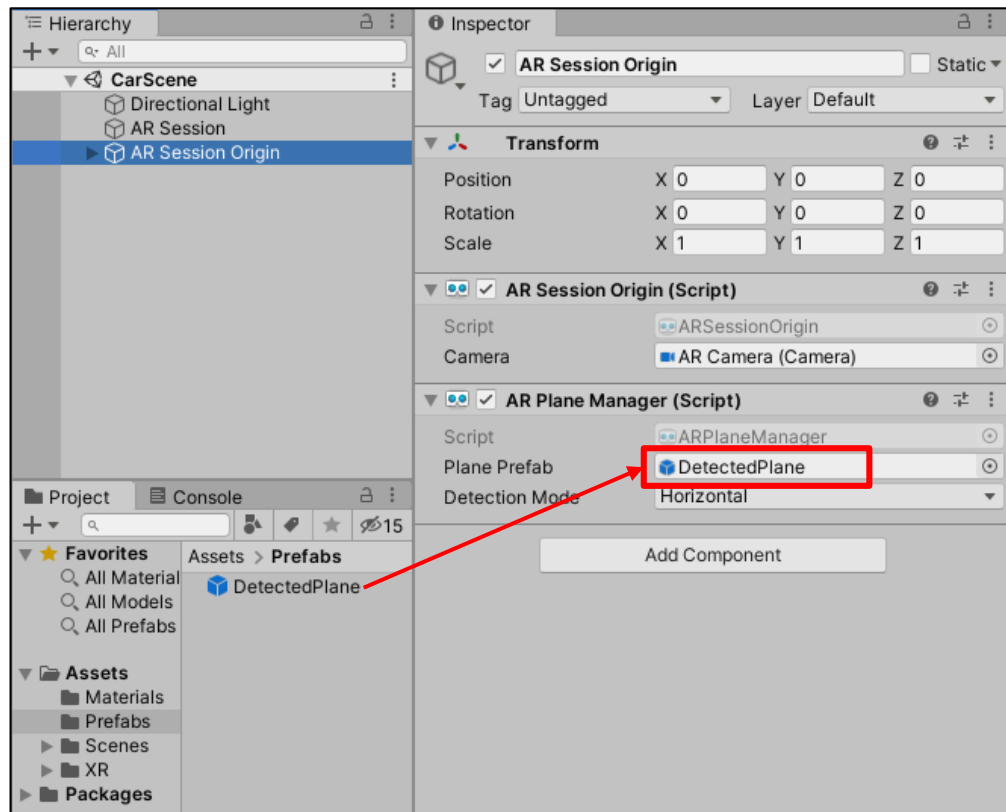


2.1-1 바닥 지형 인식하기

2. 지형 인식을 이용한 자동차 카탈로그

◆ 감지 시 생성할 프리팹 등록

- AR Session Origin 오브젝트를 선택
- AR Plane Manager 컴포넌트에 있는 Plane Prefab 항목에 DetectedPlane 프리팹을 드래그 앤 드롭하여 등록

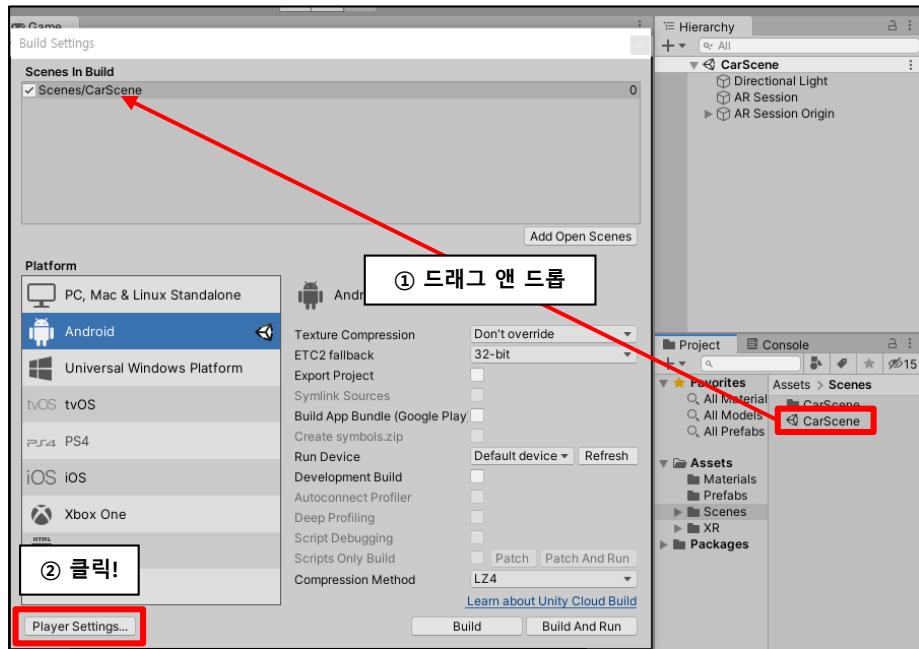


2.1-1 바닥 지형 인식하기

2. 지형 인식을 이용한 자동차 카탈로그

◆ 시작 씬 등록하기

- 키보드의 [Ctrl] + [Shift] + [B] 키를 함께 눌러서 빌드 셋팅 창을 생성
- Scene In Build 영역 박스 안쪽에 현재 씬인 CarScene을 드래그해서 0번 씬으로 추가
- 좌측 하단에 있는 [Player Settings...] 버튼을 클릭

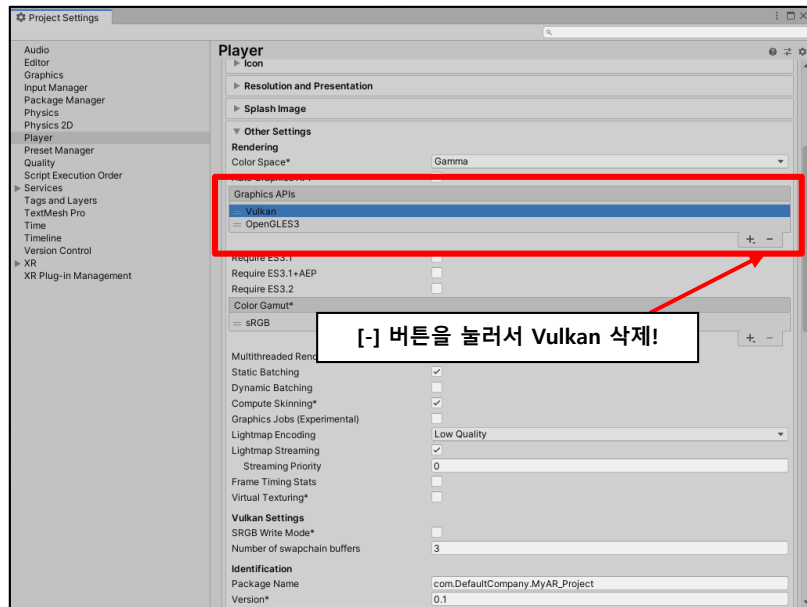
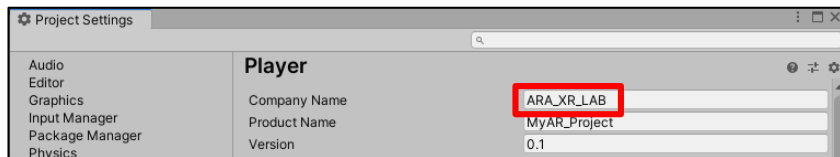


2.1-1 바닥 지형 인식하기

2. 지형 인식을 이용한 자동차 카탈로그

◆ 빌드를 위한 Player 셋팅

- Player Setting 창이 열렸으면 Company Name 항목에 기입되어 있는 “DefalutCompany” 문구를 자신만의 유니크한 회사명으로 변경
- Graphics APIs에서 Vulkan 항목을 선택하고 [-] 버튼을 눌러서 삭제

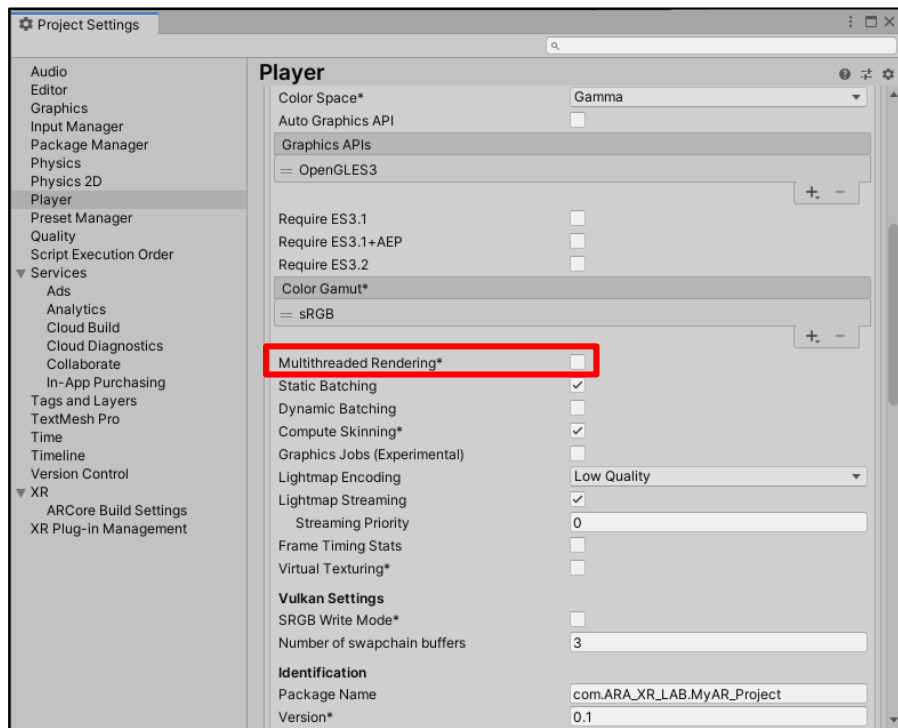


2.1-1 바닥 지형 인식하기

2. 지형 인식을 이용한 자동차 카탈로그

◆ 빌드를 위한 Player 셋팅

- Multithreaded Rendering 항목의 경우에도 멀티쓰레드 방식을 사용하지 않도록 체크를 해제
- AR Core는 싱글 쓰레드를 사용하는 OpenGL ES3만 지원하고 멀티 쓰레드 기반의 Vulkan은 지원하지 않기 때문

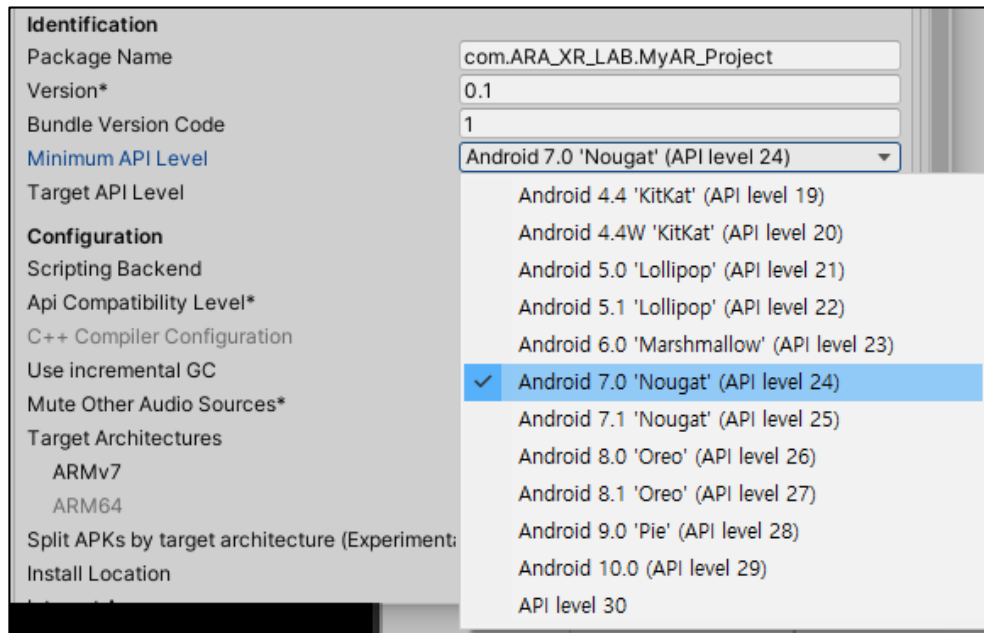


2.1-1 바닥 지형 인식하기

2. 지형 인식을 이용한 자동차 카탈로그

◆ 빌드를 위한 Player 셋팅

- AR Foundation과 AR Core의 경우 스마트폰의 안드로이드 OS 버전이 7.0 이상(API 레벨 24 이상)이어야만 사용이 가능
- Minimum API Level을 7.0(API level 24)으로 변경

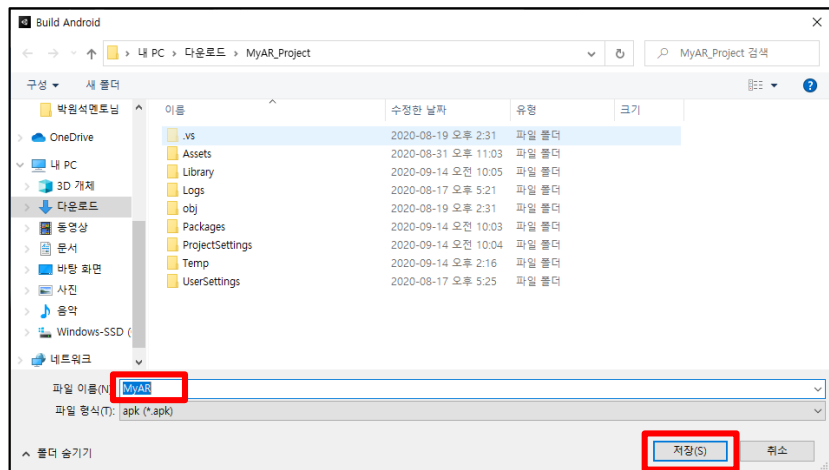
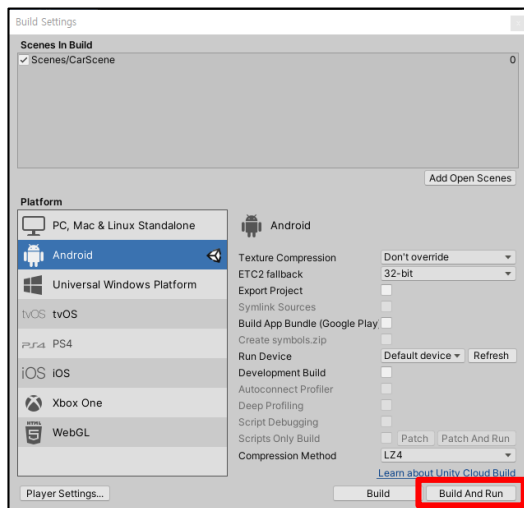


2.1-1 바닥 지형 인식하기

2. 지형 인식을 이용한 자동차 카탈로그

◆ 안드로이드 폰에 빌드하기

- 안드로이드 폰을 PC에 케이블로 연결한 상태에서 [Build And Run] 버튼을 클릭
- 생성할 빌드 파일의 이름과 저장 경로를 지정하는 창이 열리면 파일명은 “MyAR”로 하고, 저장 위치는 프로젝트 최상위 폴더로 설정
- [저장(S)] 버튼을 클릭하면 빌드가 진행됨

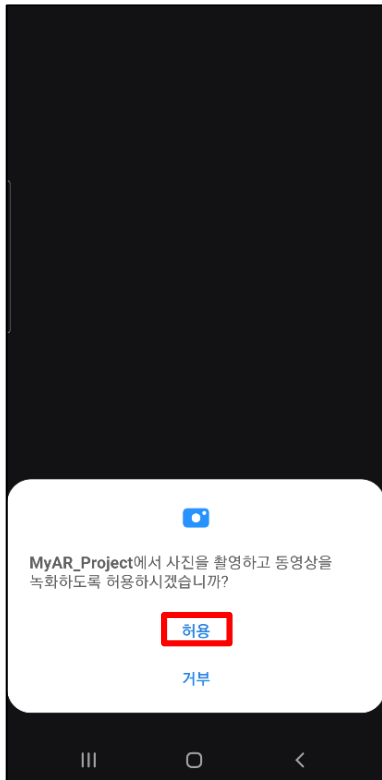


2.1-1 바닥 지형 인식하기

2. 지형 인식을 이용한 자동차 카탈로그

◆ 빌드한 앱을 테스트하기

- 빌드가 완료되면 자동으로 스마트폰에 설치까지 완료되면서 설치된 앱이 실행
- 카메라 장치 사용에 대한 허가(Permission)를 요구하는 팝업 창이 표시되는데, AR 카메라 구동에 반드시 필요하므로 [허용(Accept)] 버튼을 터치
- 바닥을 인식할 때마다 앞에서 설정했던 DetectedPlane 프리뷰가 생성되는 것을 확인
- 후면 카메라로 촬영 중인 곳의 밝기가 너무 어두우면 바닥을 잘 인식하지 못할 수 있다는 점을 주의



2-1-2. 자동차 모델링 생성하기

2. 지형 인식을 이용한 자동차 카탈로그

◆ 목표 : 카메라에 인식된 면의 특정 위치를 지정하고 터치하면 그 곳에 자동차 모델링이 생성되게 하고 싶다.

◆ 순서 :

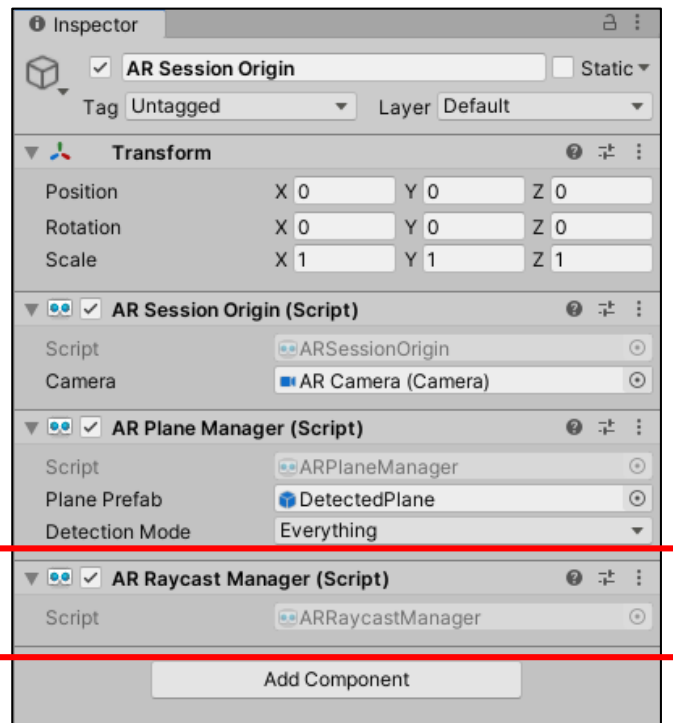
1. AR 카메라가 촬영 중인 화면 중앙을 기준으로 바닥으로 인식된 평면에 표식 이미지를 출력한다.
2. 표식이 출력 중인 상태에서 화면을 터치하면 자동차 모델링을 생성한다. 이미 자동차 모델링이 있었다면 모델링의 위치를 표식 위치로 이동한다.

2-1-2. 자동차 모델링 생성하기

2. 지형 인식을 이용한 자동차 카탈로그

◆ Ray를 이용한 바닥 감지

- AR Session Origin 오브젝트를 선택하고 [Add Component] 버튼을 클릭한 다음 검색 창에 “AR Raycast Manager”를 입력하여 컴포넌트를 추가
- 자동차 모델링을 배치할 이미지를 표시하기 위해 Materials 폴더 안에 매тери얼을 하나 생성하고, 이름을 “M_Indicator”로 변경
- 카페 자료실에서 Indicator.png 파일을 다운로드한 후 유니티 Materials 폴더에 드래그 앤 드롭하여 추가

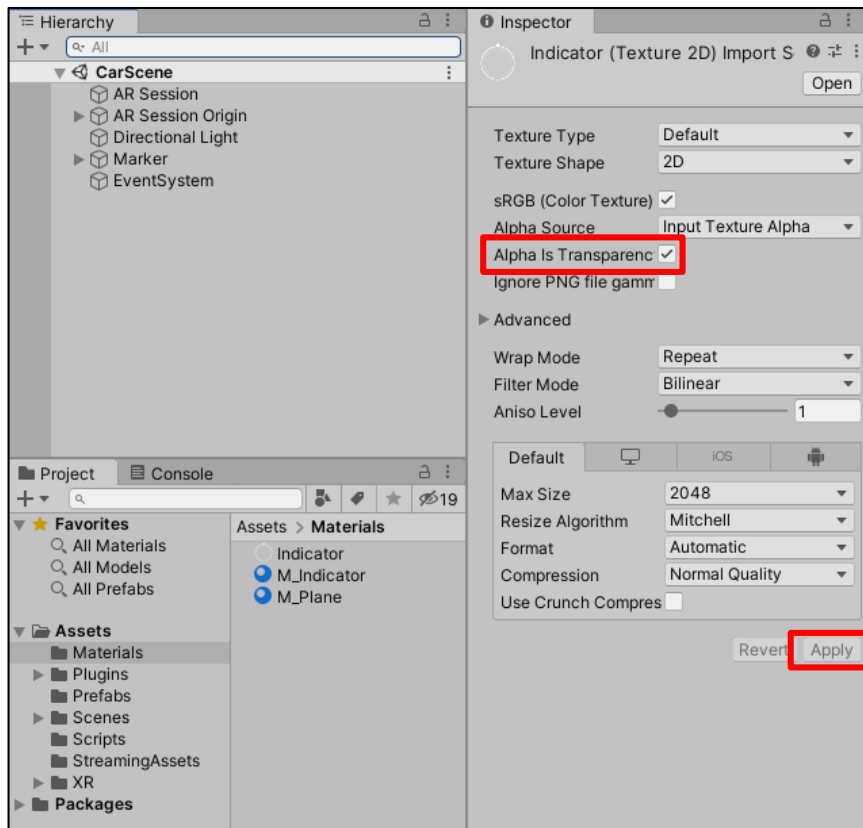


2-1-2. 자동차 모델링 생성하기

2. 지형 인식을 이용한 자동차 카탈로그

◆ 인디케이터 이미지 제작

- 이미지의 투명 값이 제대로 투명하게 표시되도록 이미지 파일을 선택하고 Hierarchy 뷰에서 [Alpha Is Transparency] 항목에 체크
- 하단의 [Apply] 버튼을 눌러서 변경된 설정을 적용
- (이미지는 타입을 Sprite로 변경하고 나서) 세팅이 완료된 이미지 파일을 M_Indicator 매터리얼에 추가

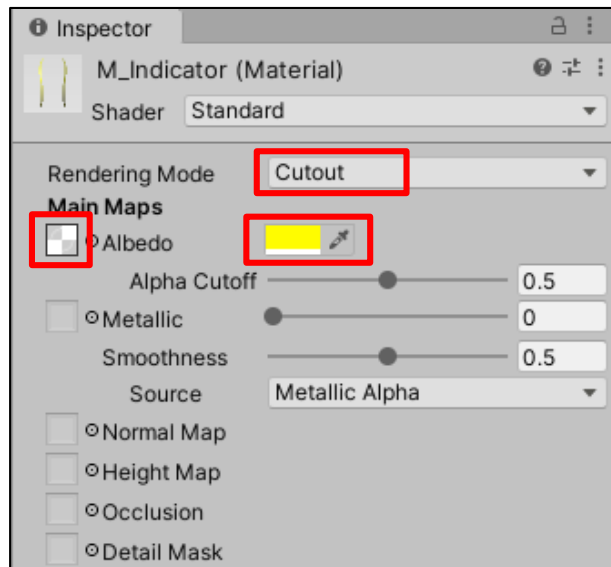


2-1-2. 자동차 모델링 생성하기

2. 지형 인식을 이용한 자동차 카탈로그

◆ 인디케이터 이미지 제작

- 세팅이 완료된 이미지 파일을 M_Indicator 매터리얼에 추가
- 이미지가 눈에 잘 띄도록 노랑색으로 Albedo를 변경
- 배경이 투명 재질로 처리될 수 있도록 Rendering Mode도 [Cutout]으로 변경

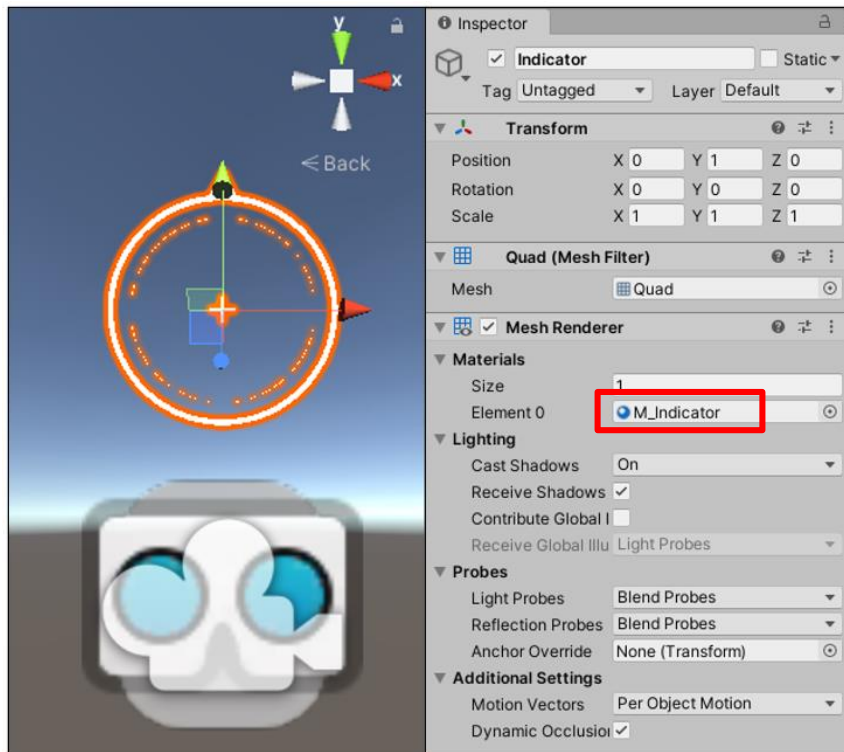


2-1-2. 자동차 모델링 생성하기

2. 지형 인식을 이용한 자동차 카탈로그

◆ 인디케이터 이미지 제작

- 하이어라키 뷰에서 [+ - 3D Object - Quad]를 선택하여 쿼드 오브젝트를 생성
- 이름을 “Indicator”로 변경
- Indicator 오브젝트의 MeshRenderer 컴포넌트에 M_Indicator 매тери얼을 추가

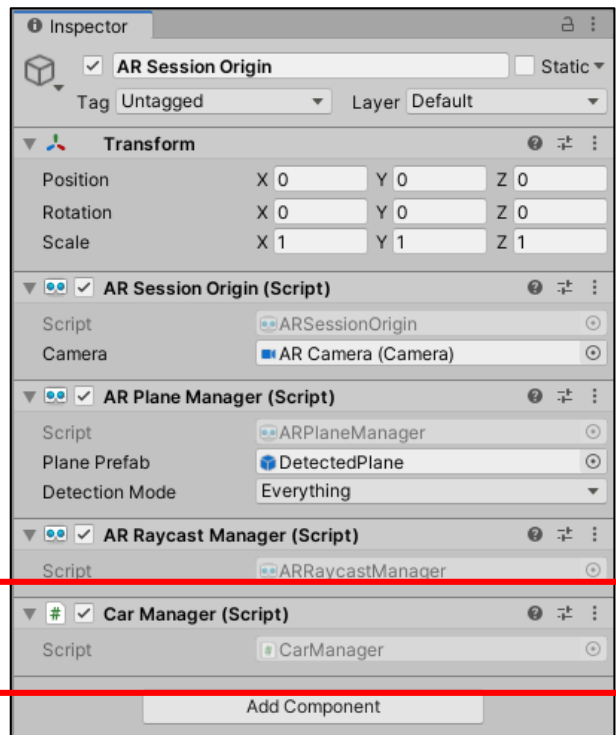


2-1-2. 자동차 모델링 생성하기

2. 지형 인식을 이용한 자동차 카탈로그

◆ AR Raycaster를 사용하기 위한 스크립트 생성

- Project 뷰에 “Scripts”라는 이름으로 폴더를 생성
- [+ - C# Script]를 선택하여 새로운 C# 스크립트를 생성하고 이름을 “CarManager”로 설정
- CarManager 스크립트를 AR Session Origin 오브젝트에 드래그 앤 드롭하여 추가



2-1-2. 자동차 모델링 생성하기

2. 지형 인식을 이용한 자동차 카탈로그

◆ AR Raycaster를 사용하기 위한 스크립트 생성

- using 문으로
UnityEngine.XR.ARFoundation 네임
스페이스 추가
- ARRaycastManager 클래스 변수 선언
- Indicator 오브젝트를 활성화하거나 위치
조정을 하기 위한 public 변수 선언

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

public class CarManager : MonoBehaviour
{
    public GameObject indicator;

    ARRaycastManager arManager;

    . . . (생략) . . .
}
```

CarManager.cs

```
ObjectManager.cs* -# x
Assembly-CSharp
ObjectManager

1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using UnityEngine.XR.ARFoundation;
5
6 Unity 스크립트 참조 0개
7 public class ObjectManager : MonoBehaviour
8 {
9     public GameObject indicator;
10    ARRaycastManager arManager;
```

2-1-2. 자동차 모델링 생성하기

2. 지형 인식을 이용한 자동차 카탈로그

◆ AR Raycaster를 사용하기 위한 스크립트

- 처음에는 인디케이터를 비활성화
- ARRaycastManager 컴포넌트 가져오기

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

using UnityEngine.XR.ARSubsystems;

public class CarManager : MonoBehaviour
{
    public GameObject indicator;

    ARRaycastManager arManager;

    void Start()
    {
        // 인디케이터를 비활성화 한다.
        indicator.SetActive(false);

        // AR Raycast Manager를 가져온다.
        arManager = GetComponent<ARRaycastManager>();
    }

    . . . (생략) . . .
}
```

CarManager.cs

2-1-2. 자동차 모델링 생성하기

2. 지형 인식을 이용한 자동차 카탈로그

◆ AR Raycaster를 사용하기 위한 스크립트

- Screen 클래스를 이용하여 정 중앙 픽셀의 좌표를 Vector2로 저장
- 레이에 부딪힌 대상들의 정보를 저장할 리스트 변수를 선언

```
public class CarManager : MonoBehaviour
{
    . . . (생략) . . .

    void Update()
    {
        // 바닥 감지 및 표식 출력용 함수
        DetectGround();
    }

    void DetectGround()
    {
        // 스크린 중앙 지점을 찾는다.
        Vector2 screenSize = new Vector2(Screen.width * 0.5f, Screen.height * 0.5f);

        // 레이에 부딪힌 대상들의 정보를 저장할 리스트 변수를 만든다.
        List<ARRaycastHit> hitInfos = new List<ARRaycastHit>();
    }
}
```

CarManager.cs

2-1-2. 자동차 모델링 생성하기

2. 지형 인식을 이용한 자동차 카탈로그

◆ AR Raycaster를 사용하기 위한 스크립트

- 레이를 이용하여 바닥을 감지
- TrackableType을 이용하여 바닥과 벽을 구분하여 감지 가능
- 화면 중앙에 바닥이 있으면 인디케이터를 활성화하고, 바닥이 없으면 인디케이터를 비활성화한다.

```
public class CarManager : MonoBehaviour
{
    . . . (생략) . . .

    void DetectGround()
    {
        // 스크린 중앙 지점을 찾는다.
        Vector2 screenSize = new Vector2(Screen.width * 0.5f, Screen.height * 0.5f);

        // 레이에 부딪힌 대상들의 정보를 저장할 리스트 변수를 만든다.
        List<ARRaycastHit> hitInfos = new List<ARRaycastHit>();

        // 만일, 스크린 중앙 지점에서 레이를 발사하였을 때 Plane 타입 추적 대상이 있다면...
        if(arManager.Raycast(screenSize, hitInfos, TrackableType.Planes))
        {
            // 표식 오브젝트를 활성화한다.
            indicator.SetActive(true);
        }
        // 그렇지 않다면 표식 오브젝트를 비활성화한다.
        else
        {
            indicator.SetActive(false);
        }
    }
}
```

CarManager.cs

2-1-2. 자동차 모델링 생성하기

2. 지형 인식을 이용한 자동차 카탈로그

◆ AR Raycaster를 사용하기 위한 스크립트

- 레이가 닿은 지점과 이미지의 방향이 일치하도록 위치 값과 회전 값을 수정

```
public class CarManager : MonoBehaviour
{
    . . . (생략) . . .

    void DetectGround()
    {
        . . . (생략) . . .

        // 만일, 스크린 중앙 지점에서 레이를 발사하였을 때 Plane 타입 추적 대상이 있다면...
        if(arManager.Raycast(screenSize, hitInfos, TrackableType.Planes))
        {
            // 표식 오브젝트를 활성화한다.
            indicator.SetActive(true);

            // 표식 오브젝트의 위치 및 회전 값을 레이가 닿은 지점에 일치시킨다.
            indicator.transform.position = hitInfos[0].pose.position;
            indicator.transform.rotation = hitInfos[0].pose.rotation;
        }

        . . . (생략) . . .
    }
}
```

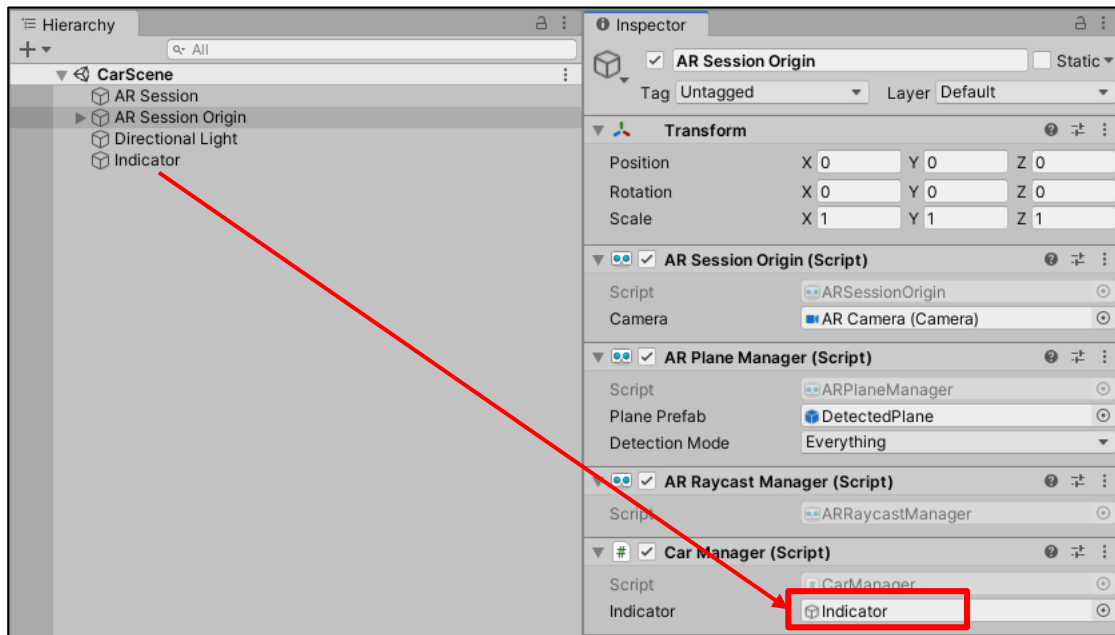
CarManager.cs

2-1-2. 자동차 모델링 생성하기

2. 지형 인식을 이용한 자동차 카탈로그

◆ 인디케이터 오브젝트 할당하기

- CarManager 컴포넌트에
인디케이터 오브젝트를 드래그 앤
드롭하여 할당
- [Ctrl] + [B] 키를 눌러서 빌드하여
테스트 실행



2-1-2. 자동차 모델링 생성하기

◆ 이미지 방향 문제

- Indicator 오브젝트가 바닥면에 수평으로 있어야 하는데 바닥에 수직으로 세워지는 문제가 발생
- 바닥은 X축과 Z축으로 펼쳐져 있는데 반해 Quad의 형태는 X축과 Y축으로 형성되어 있기 때문

2. 지형 인식을 이용한 자동차 카탈로그

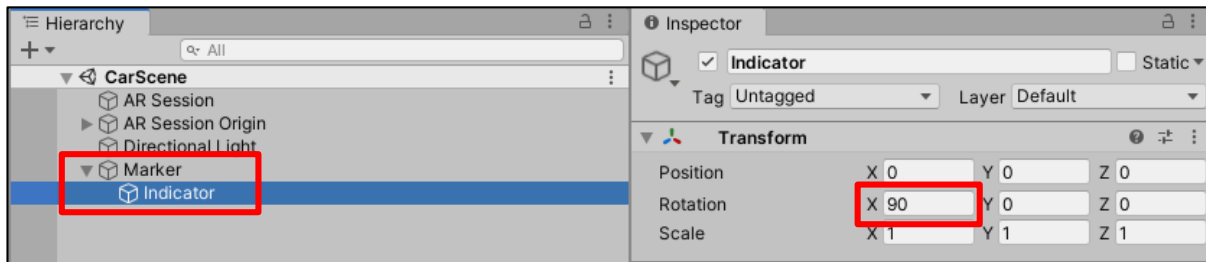


2-1-2. 자동차 모델링 생성하기

2. 지형 인식을 이용한 자동차 카탈로그

◆ 부모 오브젝트를 생성

- Hierarchy 뷰에서 빈 게임 오브젝트를 생성하고 이름을 “Marker”로 수정
- Transform 컴포넌트의 position 값과 rotation 값을 0으로 초기화하고 Indicator 오브젝트를 드래그 앤 드롭하여 자식 오브젝트로 추가
- Indicator 오브젝트의 rotation의 x 값만 90으로 변경

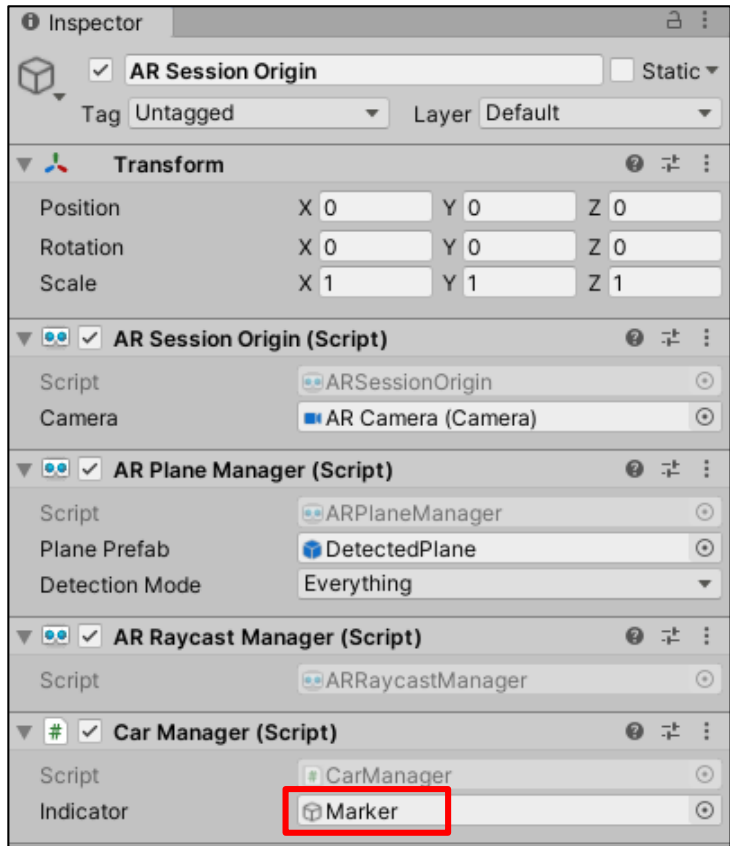
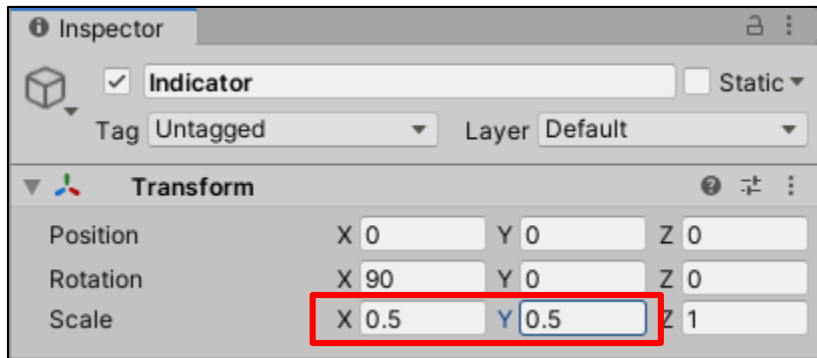


2-1-2. 자동차 모델링 생성하기

2. 지형 인식을 이용한 자동차 카탈로그

◆ 인디케이터 프리팹 교체

- AR Session Origin 오브젝트를 선택하고 CarManager 스크립트의 Indicator 항목에 Marker 오브젝트를 드래그해서 할당
- Indicator 오브젝트의 Scale의 x값과 y값을 0.5배로 축소



2-1-2. 자동차 모델링 생성하기

2. 지형 인식을 이용한 자동차 카탈로그

◆ 바닥 표시와 인디케이터가 겹치지 않도록 처리

- 바닥 면에 이미 흰색의 DetectedPlane이 깔려 있어서 Indicator 오브젝트 이미지와 겹치는 문제가 발생
- 부모인 Marker 오브젝트를 원래 위치보다 위쪽 방향으로 0.1미터 올리도록 코드를 추가
- 이후 빌드하고 다시 테스트하면서 인디케이터 이미지가 잘 표시되는 것을 확인

```
public class CarManager : MonoBehaviour
{
    ... (생략) ...

    void DetectGround()
    {
        ... (생략) ...

        // 만일, 스크린 중앙 지점에서 레이를 발사하였을 때 Plane 타입 추적 대상이 있다면...
        if(arManager.Raycast(screenSize, hitInfos, TrackableType.Planes))
        {
            // 표식 오브젝트를 활성화한다.
            indicator.SetActive(true);

            // 표식 오브젝트의 위치 및 회전 값을 레이가 닿은 지점에 일치시킨다.
            indicator.transform.position = hitInfos[0].pose.position;
            indicator.transform.rotation = hitInfos[0].pose.rotation;

            // 위치를 위쪽 방향으로 0.1미터 올린다.
            indicator.transform.position += indicator.transform.up * 0.1f;
        }
        ... (생략) ...
    }
}
```

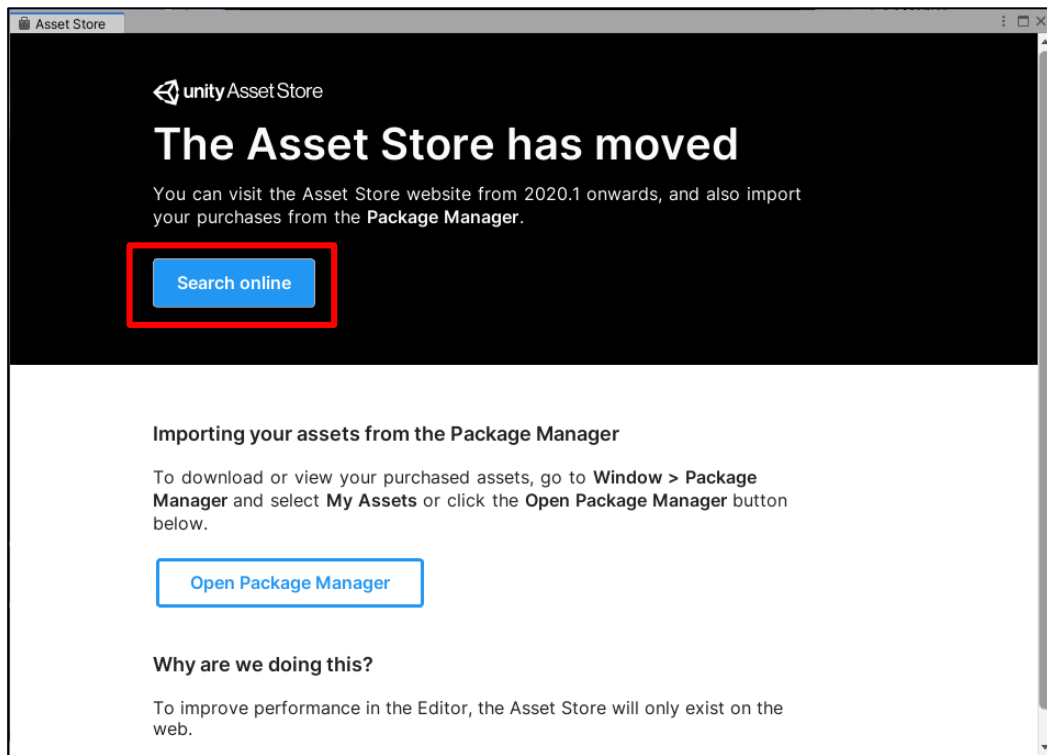
CarManager.cs

2-1-2. 자동차 모델링 생성하기

2. 지형 인식을 이용한 자동차 카탈로그

◆ 에셋 스토어에서 자동차 모델링 소스 내려받기

- 유니티 에디터 상단의 [Window – Asset Store]를 선택하여 에셋 스토어를 창을 열기
- 에셋 스토어 창이 열린 후 [Search online] 버튼을 클릭하여 유니티 에셋 스토어 웹 사이트로 접속

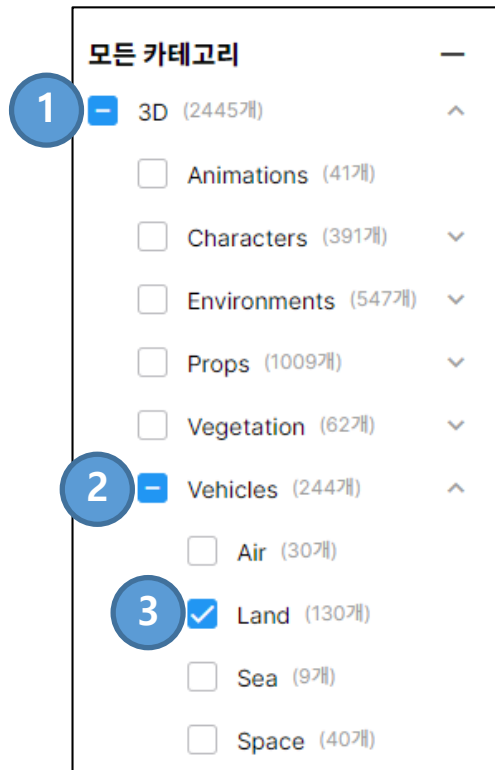


2-1-2. 자동차 모델링 생성하기

2. 지형 인식을 이용한 자동차 카탈로그

◆ 에셋 스토어에서 자동차 모델링 소스 내려받기

- 3D 모델링을 검색하기 위해서 카테고리에서 [3D] 항목을 체크하고 이어서 [Vehicles] 항목에 체크
- 자동차는 육지에서 이동하므로 [Land] 항목에도 체크

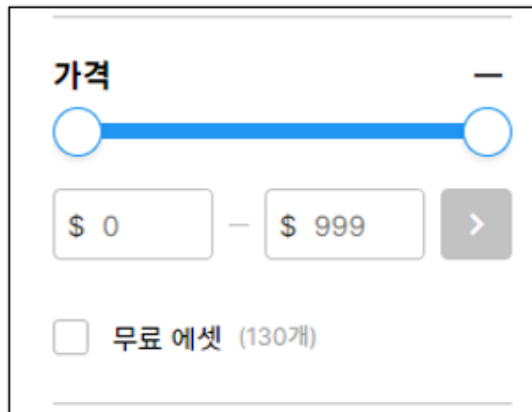


2-1-2. 자동차 모델링 생성하기

2. 지형 인식을 이용한 자동차 카탈로그

◆ 에셋 스토어에서 자동차 모델링 소스 내려받기

- 무료 에셋을 이용하기 위해 하단의 가격 탭에서도 [무료 에셋(Free Assets)] 항목에 체크



The initial state of the price filter UI shows a range slider set from \$0 to \$999. Below the slider, there are input fields for '\$ 0' and '\$ 999', a minus sign between them, and a right arrow button. At the bottom, there is an unchecked checkbox labeled '무료 에셋 (130개)'.



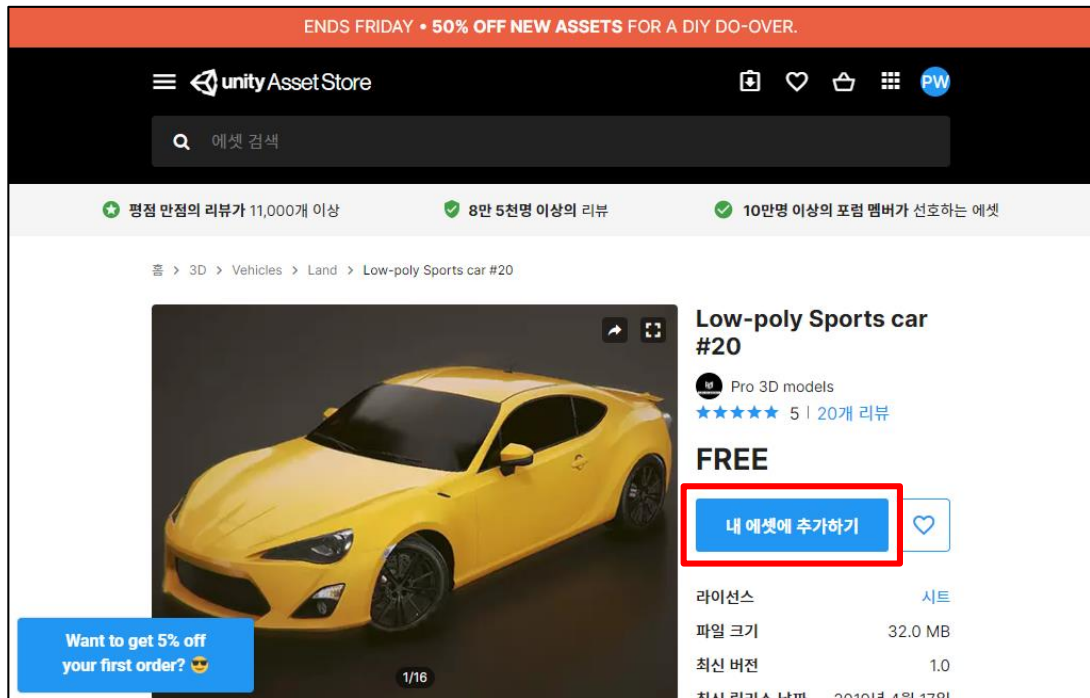
The final state of the price filter UI shows the '무료 에셋 (130개)' checkbox checked, indicating that free assets are selected for filtering.

2-1-2. 자동차 모델링 생성하기

2. 지형 인식을 이용한 자동차 카탈로그

◆ 에셋 스토어에서 자동차 모델링 소스 내려받기

- 마음에 드는 모델링을 찾았으면 우측에 [내 에셋에 추가하기] 버튼을 클릭

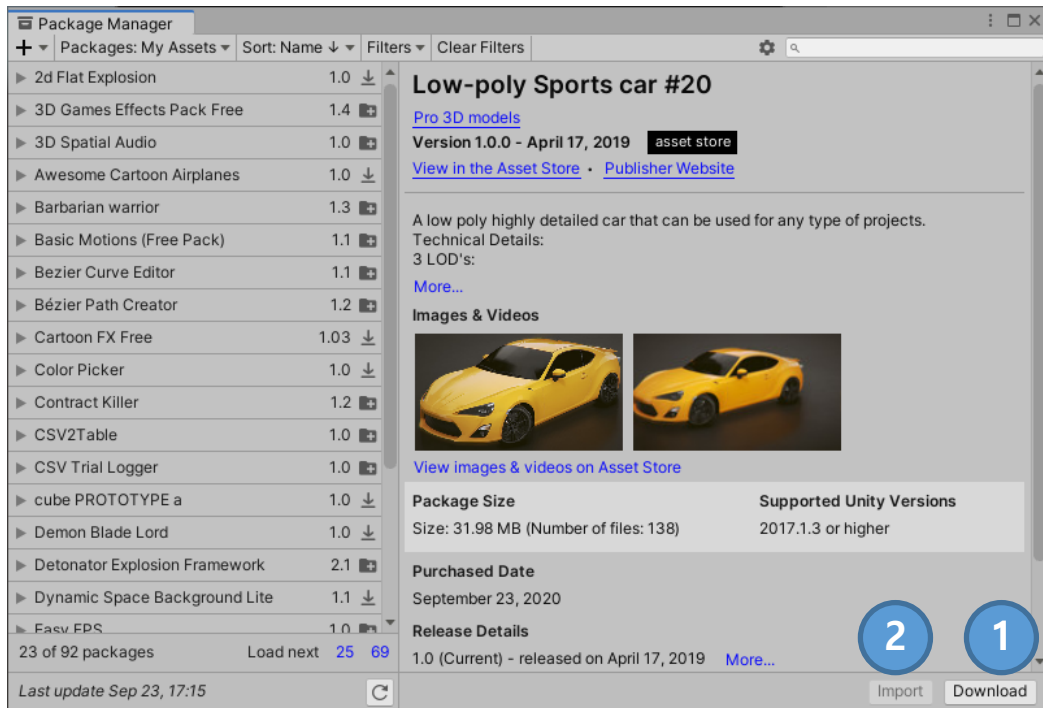


2-1-2. 자동차 모델링 생성하기

2. 지형 인식을 이용한 자동차 카탈로그

◆ 에셋 스토어에서 자동차 모델링 소스 내려받기

- 유니티 패키지 창이 열리면 우측 하단에 있는 [Download] 버튼을 클릭하여 에셋을 다운로드
- 다운로드가 완료되면 [Import] 버튼을 클릭해서 유니티 프로젝트에 추가
- 문제없이 임포트가 진행되었다면 유니티 에디터의 프로젝트 뷰에 “Best Sports CARS - Pro 3D Models”라는 이름의 폴더가 생성됨

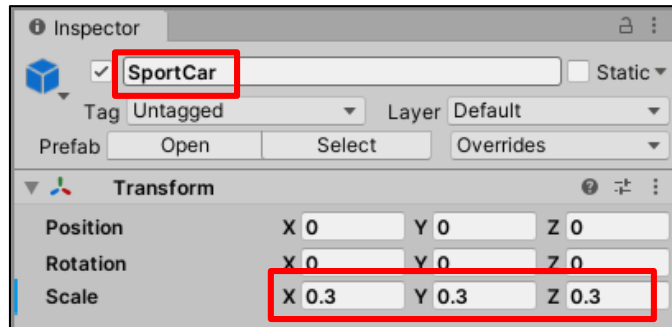
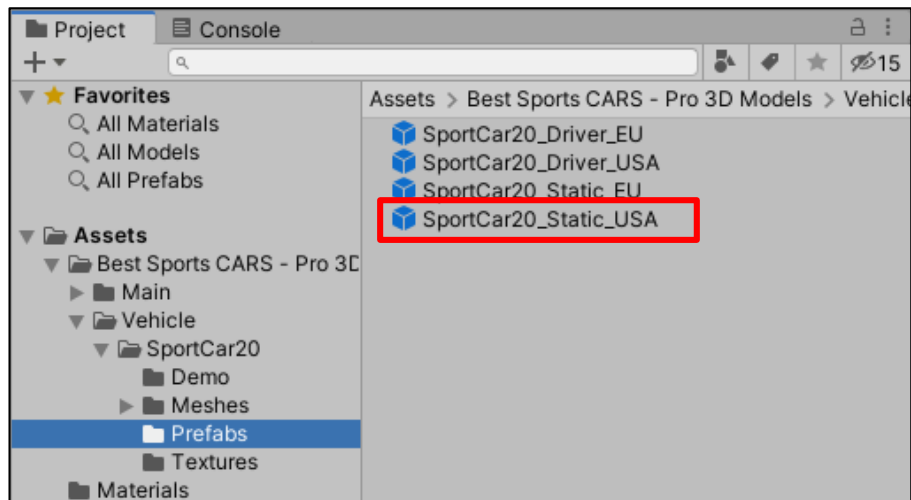


2-1-2. 자동차 모델링 생성하기

2. 지형 인식을 이용한 자동차 카탈로그

◆ 자동차 프리팹 설정

- 운전석이 좌측이고 리지드바디 컴포넌트가 없는 SportCar20_Static_USA 프리팹을 사용함
- SportCar20_Static_USA 프리팹을 씬 뷰로 드래그하여 배치해보면 거의 실제 차 만큼이나 크기 때문에 만일 스마트폰으로 촬영할 공간이 협소하다면 프리팹의 스케일을 0.3으로 축소
- 새로운 프리팹으로 만들기 위해 이름도 “SportCar”로 변경

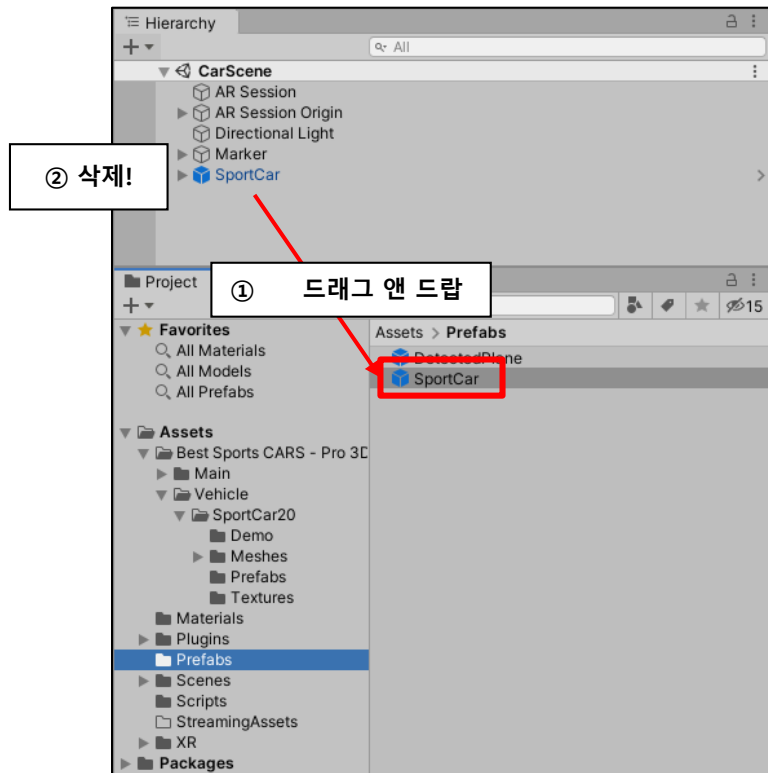


2-1-2. 자동차 모델링 생성하기

2. 지형 인식을 이용한 자동차 카탈로그

◆ 자동차 프리팹 설정

- 이름과 스케일을 조정한 SportCar 프리팹은 기존에 DetecedPlane 프리팹을 저장했었던 Prefabs 폴더에 드래그 앤 드롭하여 새로운 프리팹(Original Prefab)으로 다시 저장
- 프리팹이 저장되었으면 씬에 배치되어 있는 프리팹은 키보드의 [Delete] 키를 눌러서 삭제



2-1-2. 자동차 모델링 생성하기

2. 지형 인식을 이용한 자동차 카탈로그

◆ 터치 입력 코드

- 스포츠카 모델링을 생성하기 위한 조건으로는 Indicator 오브젝트가 활성화된 상태일 것(즉, 바닥 추적 상태일 것)과 사용자가 스마트폰 화면을 터치하였을 때의 두 가지 조건을 모두 만족하여야 한다.
- 터치 상태를 검사하기 위해 터치 상태를 가져온다.

```
public class CarManager : MonoBehaviour
{
    . . . (생략) . . .

    void Update()
    {
        // 바닥 감지 및 표식 출력용 함수
        DetectGround();

        // 인디케이터가 활성화 중일 때 화면을 터치하면 자동차 모델링이 생성되게 하고 싶다!

        // 만일, 인디케이터가 활성화 중이면서 화면 터치가 있는 상태라면...
        if(indicator.activeInHierarchy && Input.touchCount > 0)
        {
            // 첫 번째 터치 상태를 가져온다.
            Touch touch = Input.GetTouch(0);
        }
    }
    . . . (생략) . . .
}
```

CarManager.cs

2-1-2. 자동차 모델링 생성하기

2. 지형 인식을 이용한 자동차 카탈로그

◆ 터치 입력 코드

- 스포츠카 프리팹은 터치가 시작될 때 한 번만 생성되면 되므로 Began 상태일 때에 실행
- 자동차의 스포츠카 프리팹의 피벗이 바닥으로 되어 있어서 Indicator 오브젝트의 위치와 동일하게 하더라도 Indicator 이미지의 위쪽에 생성됨

```
public class CarManager : MonoBehaviour
```

CarManager.cs

```
{  
    public GameObject indicator;  
    public GameObject myCar;
```

```
    . . . (생략) . . .
```

```
    void Update()  
    {
```

```
        . . . (생략) . . .
```

```
        // 만일, 인디케이터가 활성화 중이면서 화면 터치가 있는 상태라면...
```

```
        if(indicator.activeInHierarchy && Input.touchCount > 0)
```

```
        {
```

```
            // 첫 번째 터치 상태를 가져온다.
```

```
            Touch touch = Input.GetTouch(0);
```

```
            // 만일, 터치가 시작된 상태라면 자동차를 인디케이터와 동일한 곳에 생성한다.
```

```
            if(touch.phase == TouchPhase.Began)
```

```
            {
```

```
                Instantiate(myCar, indicator.transform.position, indicator.transform.rotation);
```

```
            }
```

```
        }
```

```
    }
```

```
    . . . (생략) . . .
```

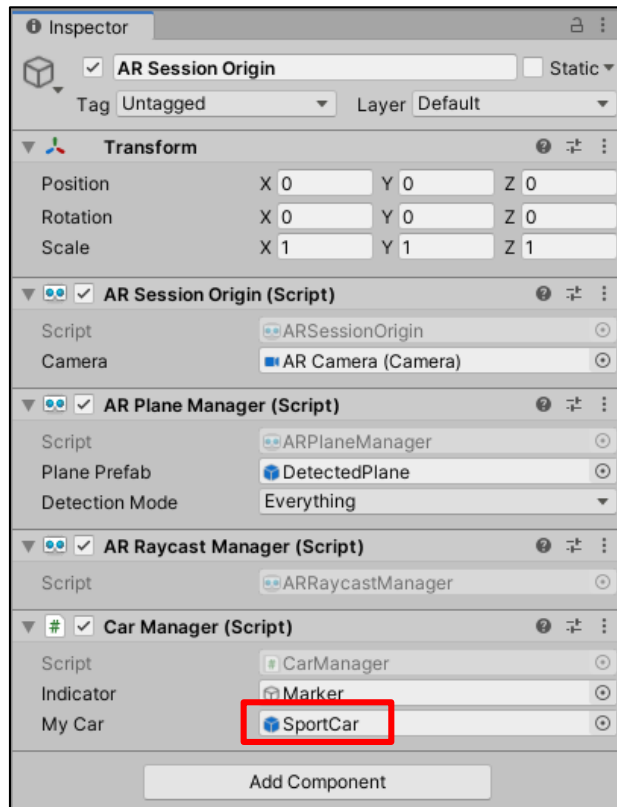
```
}
```

2-1-2. 자동차 모델링 생성하기

2. 지형 인식을 이용한 자동차 카탈로그

◆ 자동차 프리팹 설정

- CarManager 컴포넌트에 새로 생긴 myCar 항목에다가 SportsCar 프리팹을 할당
- 빌드하여 테스트 실행

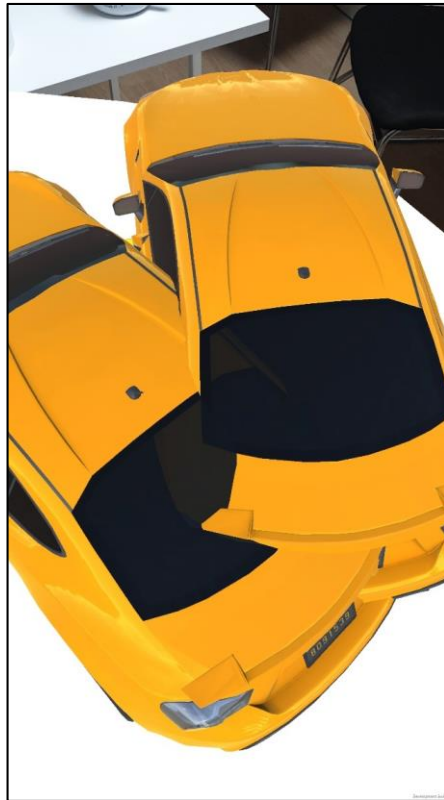


2-1-2. 자동차 모델링 생성하기

2. 지형 인식을 이용한 자동차 카탈로그

◆ 오브젝트 중복 생성 문제 발생

- Indicator 이미지가 나타난 후 화면을 터치하면 스포츠카 모델링이 잘 표시됨
- 하지만, 그 상태에서 다른 곳을 비춘 뒤에 다시 터치를 하면 기존에 생성되었던 자동차는 사라지지 않고 또 새로운 스포츠카가 생성되는 문제가 발생함



2-1-2. 자동차 모델링 생성하기

2. 지형 인식을 이용한 자동차 카탈로그

◆ 오브젝트 중복 생성 문제 발생

- 다시 CarManager 스크립트를 다시 연 다음 스포츠카 중복 체크를 위해서 현재 생성된 오브젝트가 있는지를 판별할 GameObject 변수를 선언

```
public class CarManager : MonoBehaviour
{
    public GameObject indicator;
    public GameObject myCar;

    ARRaycastManager arManager;
    GameObject placedObject;

    . . . (생략) . . .
}
```

CarManager.cs

2-1-2. 자동차 모델링 생성하기

2. 지형 인식을 이용한 자동차 카탈로그

- 화면 터치 시 오브젝트를 생성하는 코드 부분에서 썬에 생성된 오브젝트가 없는 경우와 있는 경우를 나누어서 처리
- placedObject 변수가 비어있는 경우에는 프리팹을 새로 생성한 뒤 placedObject 변수에 추가하고, placedObject 변수에 할당된 오브젝트가 있을 경우에는 placedObject 변수에 할당된 오브젝트의 위치와 회전 값만 변경
- 빌드하고 앱을 실행하여 여러 번 터치해도 스포츠카 오브젝트가 중복 생성되지 않음을 확인

```
void Update()
{
    ... (생략) ...
```

CarManager.cs

```
// 만일, 터치가 시작된 상태라면 자동차를 인디케이터와 동일한 곳에 생성한다.
```

```
if(touch.phase == TouchPhase.Began)
```

```
{
```

```
// 만일 생성된 오브젝트가 없다면 프리팹을 썬에 생성하고, placedObject 변수에 할당한다.
```

```
if (placedObject == null)
```

```
{
```

```
    placedObject = Instantiate(myCar, indicator.transform.position, indicator.transform.rotation);
```

```
}
```

```
// 생성된 오브젝트가 있다면 그 오브젝트의 위치와 회전 값을 변경한다.
```

```
else
```

```
{
```

```
    placedObject.transform.SetPositionAndRotation(indicator.transform.position, indicator.transform.rotation);
```

```
}
```

```
}
```

```
}
```

2.1-3 자동차 모델링 조작하기

2. 지형 인식을 이용한 자동차 카탈로그

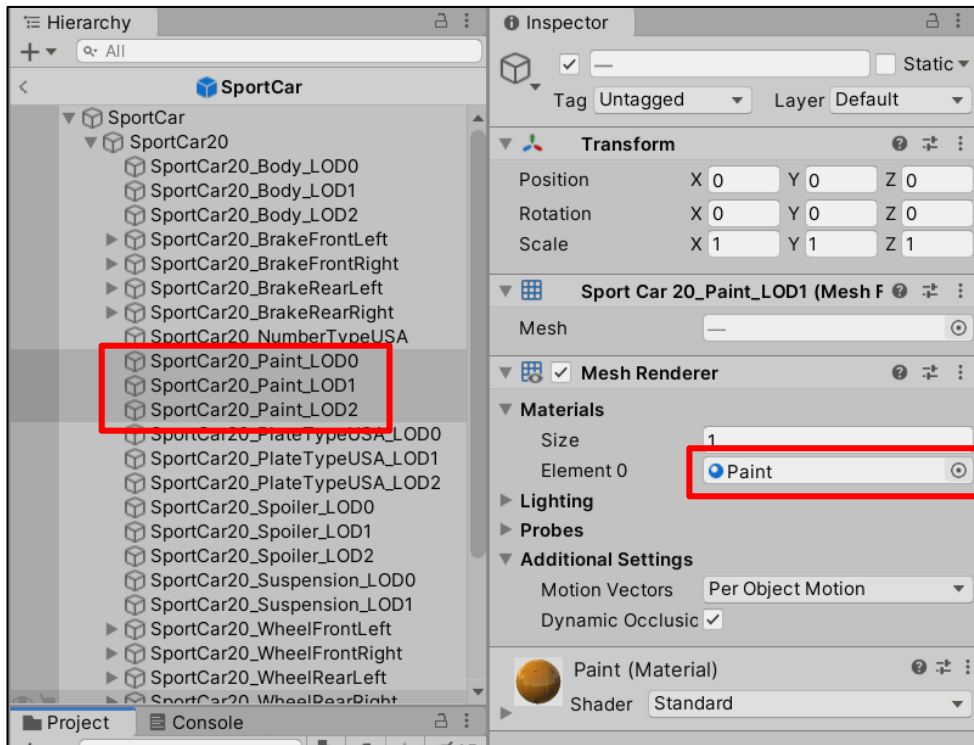
- ◆ 목표 : 생성된 자동차 모델링의 색상을 변경하거나 회전시키고 싶다.
- ◆ 순서 :
 1. 준비된 색상에 따라 UI 버튼을 제작하고, 사용자의 선택에 따라 자동차의 외장 색상을 변경한다.
 2. 자동차 모델링을 터치한 상태에서 손가락을 좌우로 움직이면 그 방향으로 자동차 모델링이 회전한다.

2.1-3 자동차 모델링 조작하기

2. 지형 인식을 이용한 자동차 카탈로그

◆ 자동차 모델링의 색상 변경

- 외관을 담당하는 재질(Material)을 알기 위해서 자동차 프리팹을 더블 클릭하여 프리팹 설정 화면을 열어준다.
- 스포츠카 오브젝트의 자식 오브젝트 중에 “Paint”라는 이름이 들어가 있는 오브젝트 세 가지에 공통적으로 Paint라는 매тери얼이 있음을 확인

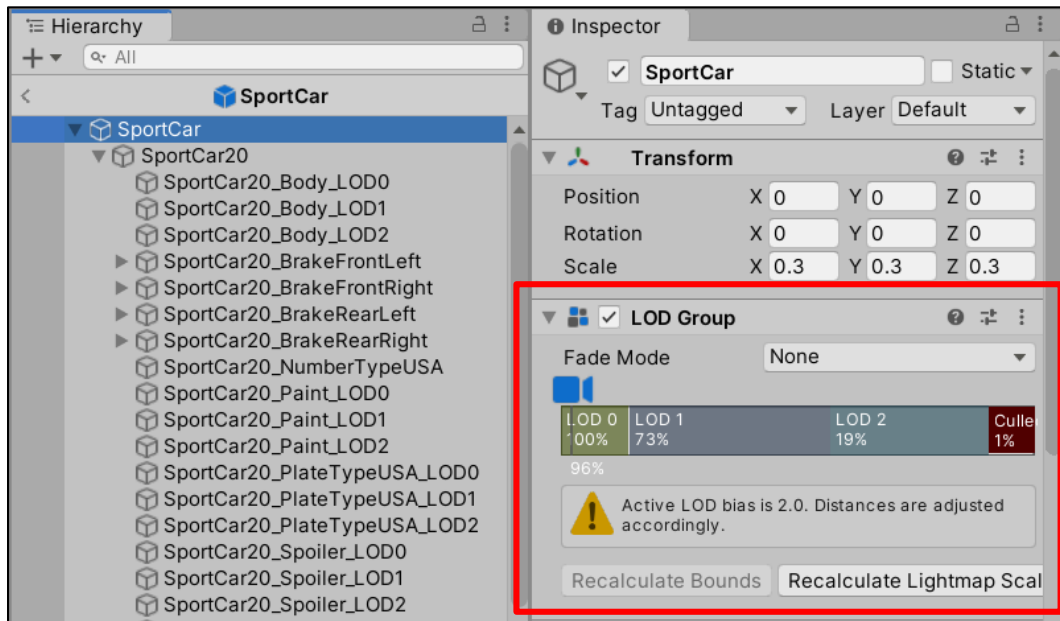


2.1-3 자동차 모델링 조작하기

2. 지형 인식을 이용한 자동차 카탈로그

◆ 자동차 모델링의 색상 변경

- SportCar 오브젝트에는 LOD Group 컴포넌트가 있으므로, 카메라와 자동차의 거리에 관계없이 색상이 유지되게 하려면 SportCar20_Paint_LOD 0 ~ SportCar20_Paint_LOD 2까지 전부 매터리얼의 색상을 바꿔주어야 한다.



2.1-3 자동차 모델링 조작하기

2. 지형 인식을 이용한 자동차 카탈로그

◆ 자동차 모델링의 색상 변경

- “CarController”라는 이름으로 새로운 C# 스크립트를 생성
- LOD별 오브젝트 3개에 대한 GameObject 배열 변수와 각각의 오브젝트 별 Material 배열 변수, 그리고 색상 정보를 담을 Color32 배열 변수를 선언

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

public class CarController : MonoBehaviour
{
    public GameObject[] bodyObject;
    public Color32[] colors;

    Material[] carMats;

    . . . (생략) . . .
}
```

CarController.cs

2.1-3 자동차 모델링 조작하기

2. 지형 인식을 이용한 자동차 카탈로그

◆ 자동차 모델링의 색상 변경

- 매터리얼 배열(CarMats)을 순회하면서 LOD 오브젝트의 매터리얼을 할당
- 현재 자동차 프리팹의 색상을 기본 색상으로 저장
- 기본 색상은 colors 배열의 0번 인덱스에 저장

```
public class CarController : MonoBehaviour
{
    ... (생략) ...

    void Start()
    {
        // carMats 배열을 자동차 바디 오브젝트의 수만큼 초기화한다.
        carMats = new Material[bodyObject.Length];

        // 자동차 바디 오브젝트의 매터리얼 각각을 carMats 배열에 지정한다.
        for(int i = 0; i < carMats.Length; i++)
        {
            carMats[i] = bodyObject[i].GetComponent<MeshRenderer>().material;
        }

        // 색상 배열 0번에는 매터리얼의 초기 색상을 저장한다.
        colors[0] = carMats[0].color;
    }
    ... (생략) ...
}
```

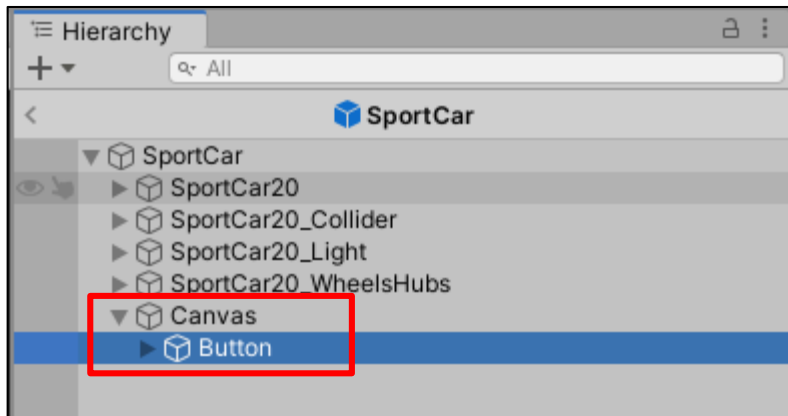
CarController.cs

2.1-3 자동차 모델링 조작하기

2. 지형 인식을 이용한 자동차 카탈로그

◆ 색상 변경 버튼 UI 제작

- 자동차의 색상 변경을 위한 버튼이 필요
- 버튼 UI는 처음에는 보이지 않다가 스포츠카가 화면에 생성되었을 때에만 표시되도록 UI 오브젝트들을 모두 자동차 오브젝트의 자식 오브젝트로 등록
- 추후에 여러 개의 자동차를 생성하더라도 EventSystem이 중복 생성되지 않도록 EventSystem은 자동차의 자식 오브젝트로 등록하지 않는다.

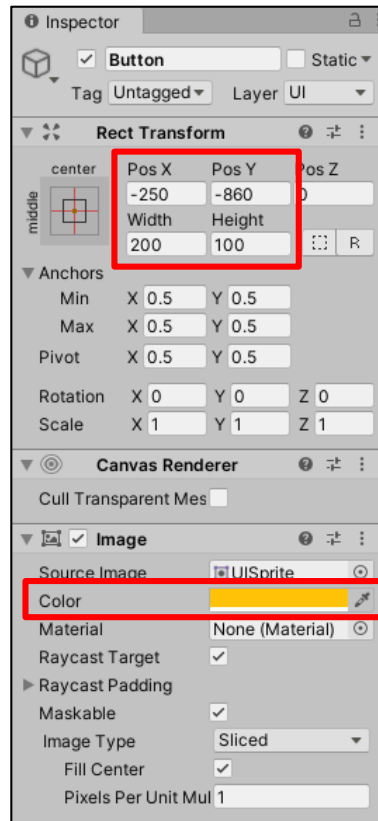


2.1-3 자동차 모델링 조작하기

◆ 색상 변경 버튼 UI 제작

- 버튼 UI는 화면에 하단에 배치
- 버튼의 색상은 자동차의 본래 색상과 비슷하게 설정

2. 지형 인식을 이용한 자동차 카탈로그

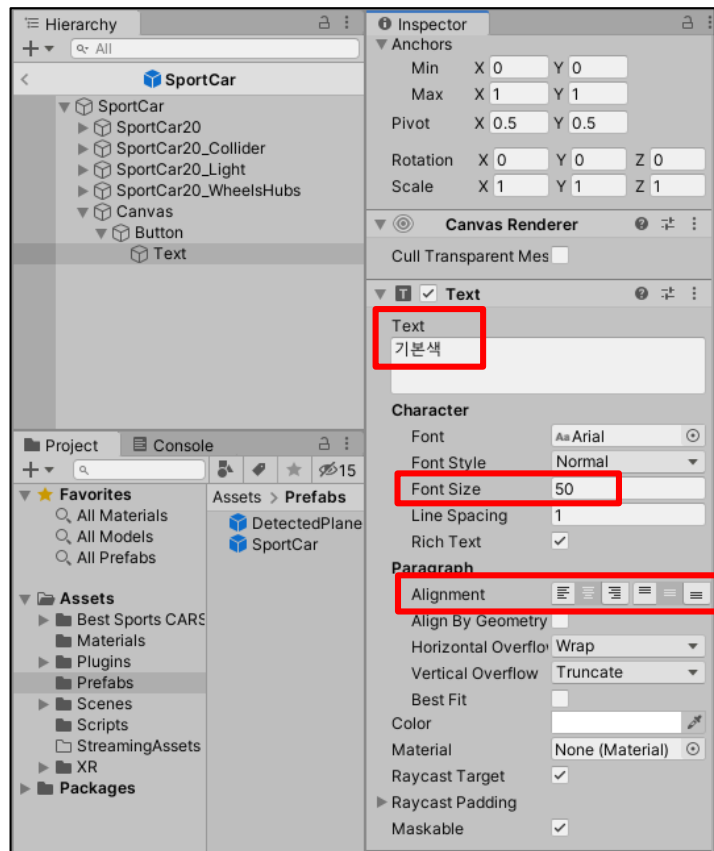


2.1-3 자동차 모델링 조작하기

2. 지형 인식을 이용한 자동차 카탈로그

◆ 색상 변경 버튼 UI 제작

- 버튼의 자식 오브젝트로 있는 Text 오브젝트에서는 글씨 폰트를 50으로 설정하고 “기본색”이라는 문구로 변경
- 폰트의 정렬(Alignment)은 상하 정렬과 좌우 정렬을 모두 중앙 정렬로 설정

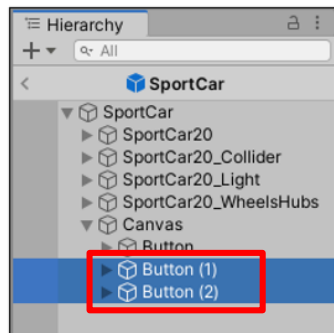
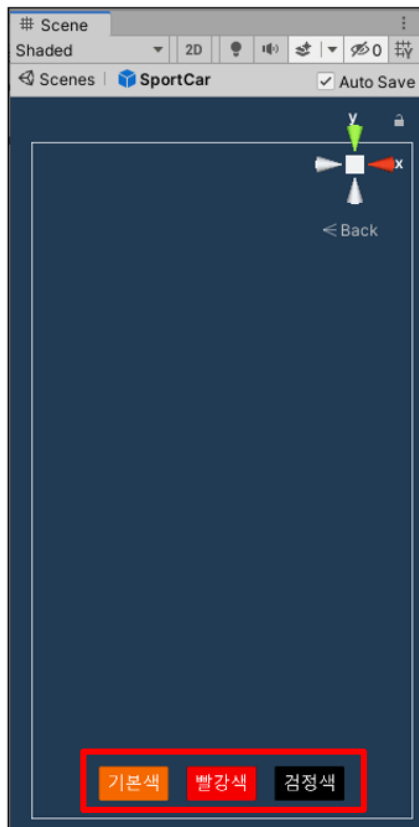


2.1-3 자동차 모델링 조작하기

2. 지형 인식을 이용한 자동차 카탈로그

◆ 색상 변경 버튼 UI 제작

- 기본 색 기능을 담당할 버튼이 완성되었으니 키보드의 [Ctrl] + [D] 버튼을 두 번 눌러서 버튼을 2회 복제
- 복제한 버튼의 색상과 텍스트 내용을 우측 그림처럼 변경



2.1-3 자동차 모델링 조작하기

2. 지형 인식을 이용한 자동차 카탈로그

◆ 색상 변경 버튼 기능 구현

- 버튼의 매개변수 값에 따라 매터리얼의 색상을 변경하는 함수를 선언
- 모든 LOD 매터리얼을 전부 변경하기 위해 반복문을 사용

```
public class CarController : MonoBehaviour
{
    public GameObject[] bodyObject;
    public Color32[] colors;

    Material[] carMats;

    . . . (생략) . . .

    public void ChangeColor(int num)
    {
        // 각 LOD 매터리얼의 색상을 버튼에 지정된 색상으로 변경한다.
        for(int i = 0; i < carMats.Length; i++)
        {
            carMats[i].color = colors[num];
        }
    }
}
```

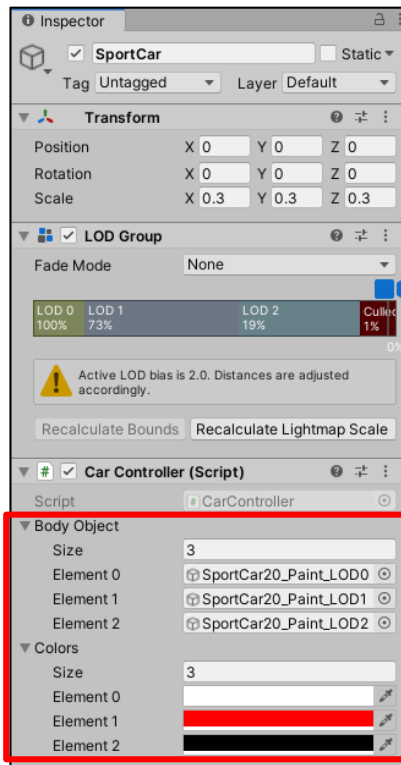
CarController.cs

2.1-3 자동차 모델링 조작하기

2. 지형 인식을 이용한 자동차 카탈로그

◆ 색상 변경 버튼 기능 구현

- CarController 스크립트를 드래그해서 SportCar 프리팹 최상위 부모 오브젝트에 추가
- Body Object 배열 변수에는 자동차 외형 오브젝트 3종(SportCar20_Paint_LOD0~2)을 할당
- Colors 배열 변수의 1번과 2번에는 각각 빨강색과 파랑색을 설정

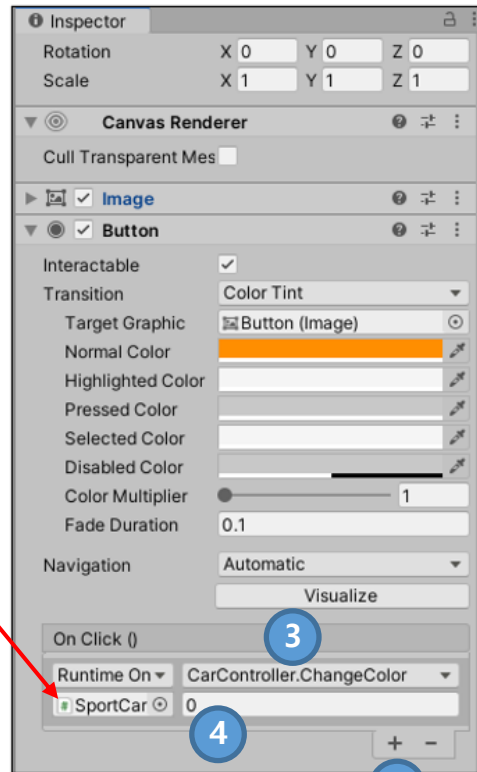
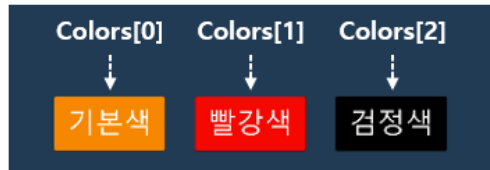
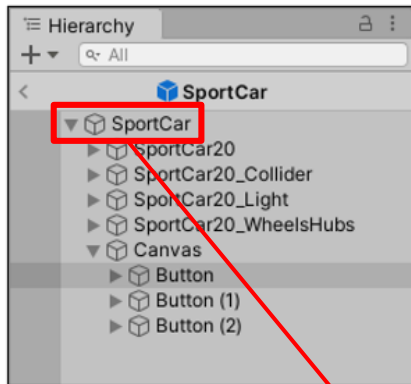


2.1-3 자동차 모델링 조작하기

2. 지형 인식을 이용한 자동차 카탈로그

◆ 색상 변경 버튼 기능 구현

- Button 컴포넌트 하단의 OnClick() 이벤트 부분의 [+] 버튼을 눌러서 빈 목록을 1개 생성
- 빈 목록의 오브젝트 영역에 CarController 스크립트가 있는 SportCar 오브젝트를 드래그 앤 드롭하여 할당
- 함수 선택 콤보 박스를 클릭하고 [CarController - ChangeColor]를 선택하여 만들어 놓았던 함수를 연결
- 매개 변수 입력 영역에는 버튼 순서에 따라 위에서부터 0번 ~ 2번까지 값을 입력



2.1-3 자동차 모델링 조작하기

2. 지형 인식을 이용한 자동차 카탈로그

◆ 버튼 기능과 위치 변경 기능 중복 문제

- 빌드해서 스마트폰에서 테스트 실시
- 버튼을 터치하면 색상이 잘 변경되긴 하지만, 버튼을 누를 때마다 스포츠카의 위치까지도 변경되는 문제가 발생
- 버튼 터치도 스마트폰 화면 터치로 인식하기 때문이므로 버튼 UI를 터치할 때의 예외 처리가 필요



2.1-3 자동차 모델링 조작하기

2. 지형 인식을 이용한 자동차 카탈로그

◆ 버튼 기능과 위치 변경 기능 중복 문제

- CarManager 스크립트를 열고
UnityEngine.EventSystem 네임
스페이스를 using문으로 추가
- 현재 클릭(or 터치)한 오브젝트가 UI
오브젝트일 때에 true를 반환하는 속성
변수 currentSelectedGameObject를
이용하여 하단의 코드를 실행하지 않고
Update() 함수를 종료시킨다.
- 다시 빌드해서 화면 터치와 버튼 터치
입력 테스트

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.XR.ARFoundation;
using UnityEngine.XR.ARSubsystems;
using UnityEngine.EventSystems;

public class CarManager : MonoBehaviour
{
    . . . (생략) . . .

    void Update()
    {
        . . . (생략) . . .

        // 만일, 현재 클릭 or 터치한 오브젝트가 UI 오브젝트라면 Update 함수를 종료한다.
        if(EventSystem.current.currentSelectedGameObject)
        {
            return;
        }

        // 만일, 인디케이터가 활성화 중이면서 화면 터치가 있는 상태라면...
        . . . (생략) . . .
    }
}
```

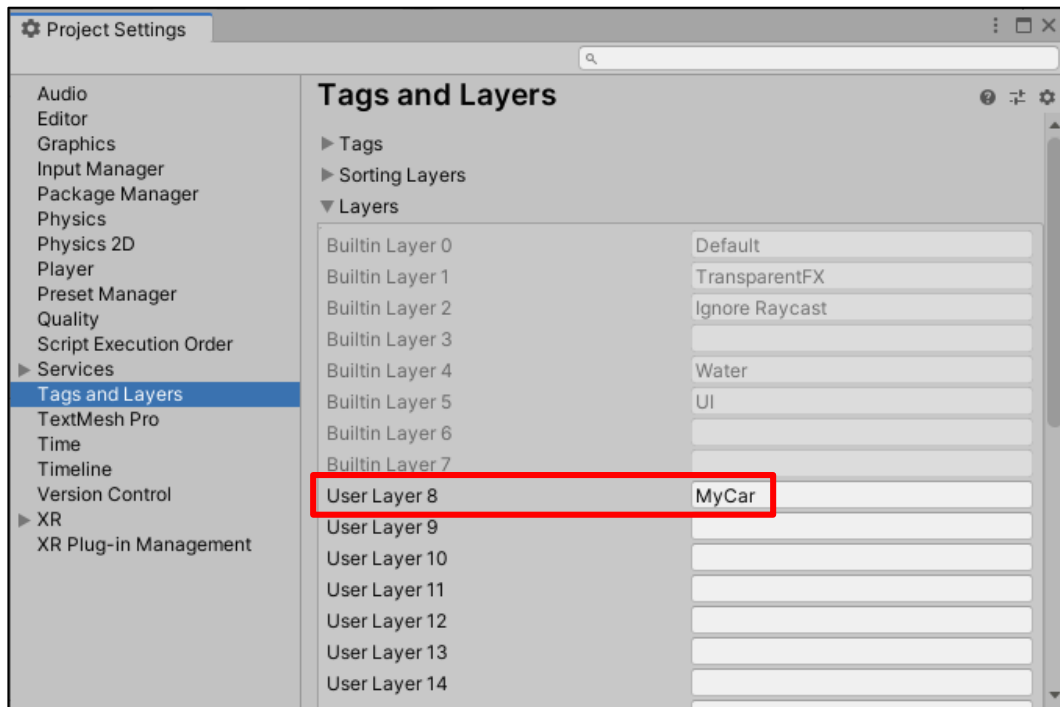
CarManager.cs

2.1-3 자동차 모델링 조작하기

2. 지형 인식을 이용한 자동차 카탈로그

◆ 터치 스와이프(Swipe)로 모델링 회전시키기

- 유니티 에디터 좌측 상단의 [Edit – Project Settings...]를 선택하여 프로젝트 설정 창을 열고 Tags and Layers 탭을 선택
- Layers 항목의 User Layer 8번 우측의 빈 칸에 “MyCar”라는 이름으로 레이어를 추가

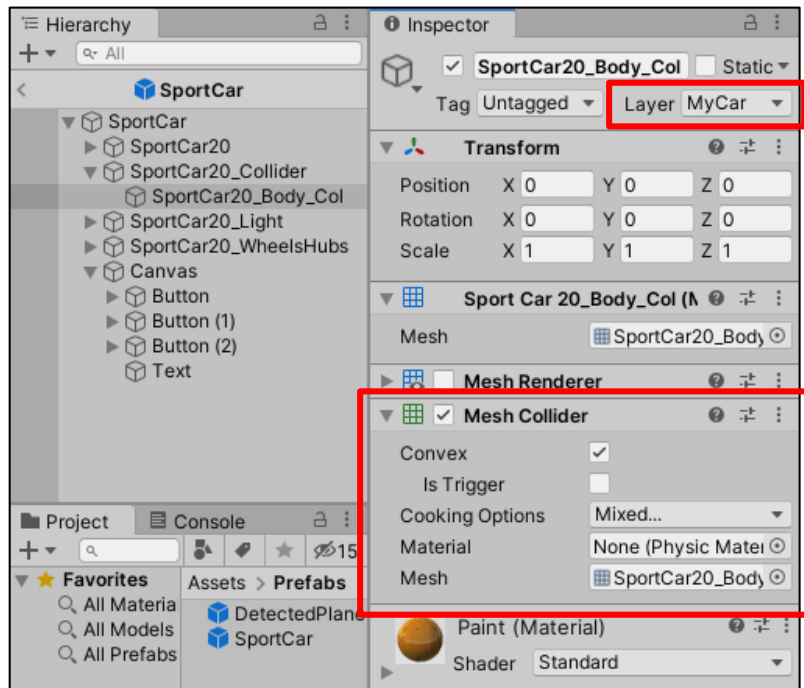


2.1-3 자동차 모델링 조작하기

2. 지형 인식을 이용한 자동차 카탈로그

◆ 터치 스와이프(Swipe)로 모델링 회전시키기

- 스포츠카의 자식 오브젝트 중에 SportCar20_Body_Col 오브젝트에 Mesh Collider 컴포넌트가 존재를 확인
- SportCar20_Body_Col 오브젝트의 레이어를 “MyCar” 레이어로 변경



2.1-3 자동차 모델링 조작하기

2. 지형 인식을 이용한 자동차 카탈로그

◆ 터치 스와이프(Swipe)로 모델링 회전시키기

- CarController 스크립트를 열고, 터치가 발생했을 때 터치 상태가 움직이고 있는 중인지를 조건식으로 체크
- Ray를 발사해서 부딪힌 대상이 자동차일 때를 감지한다.

```
public class CarController : MonoBehaviour
{
    ... (생략) ...

    void Update()
    {
        // 만일, 터치 된 부위가 1개 이상이라면...
        if (Input.touchCount > 0)
        {
            Touch touch = Input.GetTouch(0);

            // 만일, 터치 상태가 움직이고 있는 중이라면...
            if (touch.phase == TouchPhase.Moved)
            {
                // 만일, 카메라 위치에서 정면 방향으로 레이를 발사하여 부딪힌 대상이
                // 8번 레이어라면 터치 이동량을 구한다.
                Ray ray = new Ray(Camera.main.transform.position, Camera.main.transform.forward);
                RaycastHit hitInfo;

                if (Physics.Raycast(ray, out hitInfo, Mathf.Infinity, 1 << 8))
                {
                    Vector3 deltaPos = touch.deltaPosition;

                }
            }
        }
    }
    ... (생략) ...
}
```

CarController.cs

2.1-3 자동차 모델링 조작하기

2. 지형 인식을 이용한 자동차 카탈로그

◆ 터치 스와이프(Swipe)로 모델링 회전시키기

- Transform 클래스의 Rotate() 함수를 사용하여 y축 방향으로 터치 이동량만큼 자동차 오브젝트를 회전시킨다.
- deltaPosition.x 값은 음수(-) 값이 좌측 방향이고 양수(+) 값이 우측 방향인 반면, y축 회전 방향은 음수(-) 값이 우회전 방향이고 양수(+) 값이 좌회전 방향이므로 서로 반대 방향인 점에 유의

```
void Update()
{
    // 만일, 터치 된 부위가 1개 이상이라면...
    if (Input.touchCount > 0)
    {
        Touch touch = Input.GetTouch(0);

        // 만일, 터치 상태가 움직이고 있는 중이라면...
        if (touch.phase == TouchPhase.Moved)
        {
            // 만일, 카메라 위치에서 정면 방향으로 레이를 발사하여 부딪힌 대상이
            // 8번 레이어라면 터치 이동량을 구한다.
            Ray ray = new Ray(Camera.main.transform.position, Camera.main.transform.forward);
            RaycastHit hitInfo;

            if (Physics.Raycast(ray, out hitInfo, Mathf.Infinity, 1 << 8))
            {
                Vector3 deltaPos = touch.deltaPosition;

                // 직전 프레임에서 현재 프레임까지의 x축 터치 이동량에 비례하여
                // 로컬 y축 방향으로 회전시킨다.
                transform.Rotate(transform.up, deltaPos.x * -1.0f);
            }
        }
    }
}
```

CarController.cs

2.1-3 자동차 모델링 조작하기

2. 지형 인식을 이용한 자동차 카탈로그

◆ 터치 스와이프(Swipe)로 모델링 회전시키기

- 너무 빠르게 회전되지 않도록 회전 속도 변수 `rotSpeed`를 선언
- 오브젝트 회전 코드에서 회전 속도 변수를 곱하도록 추가

```
public class CarController : MonoBehaviour
{
    public GameObject[] bodyObject;
    public Color32[] colors;
    public float rotSpeed = 0.1f;

    Material[] carMats;

    . . . (생략) . . .

    void Update()
    {
        . . . (생략) . . .

        if (Physics.Raycast(ray, out hitInfo, Mathf.Infinity, 1 << 8))
        {
            Vector3 deltaPos = touch.deltaPosition;

            // 직전 프레임에서 현재 프레임까지의 x축 터치 이동량에 비례하여
            // 로컬 y축 방향으로 회전시킨다.
            transform.Rotate(transform.up, deltaPos.x * -1.0f * rotSpeed)

        }
    }

    . . . (생략) . . .
}
```

CarController.cs

2.1-3 자동차 모델링 조작하기

2. 지형 인식을 이용한 자동차 카탈로그

- 너무 미세한 카메라 위치 변경 시에도 모델링 위치가 변동되는 민감함을 조정할 필요가 있음
- 최소 이동 범위를 지정하기 위한 float 변수 relocationDistance를 선언
- 최소 이동 범위 값은 1(미터)로 설정
- 기존 위치 변경 코드에서 최소 이동 범위 이상이 될 것을 조건으로 설정

```
public class CarController : MonoBehaviour
{
    public GameObject indicator;
    public GameObject myCar;
    public float relocationDistance = 1.0f;

    ... (생략) ...

    void Update()
    {
        ... (생략) ...

        // 만일, 인디케이터가 활성화 중이면서 화면 터치가 있는 상태라면...
        if (indicator.activeInHierarchy && Input.touchCount > 0)
        {
            ... (생략) ...

            // 생성된 오브젝트가 있다면 그 오브젝트의 위치와 회전 값을 변경한다.
            else
            {
                // 만일 생성된 오브젝트와 인디케이터 사이의 거리가
                // 최소 이동 범위 이상이라면...
                if (Vector3.Distance(placedObject.transform.position, indicator.transform.position) > relocationDistance)
                {
                    placedObject.transform.SetPositionAndRotation(indicator.transform.position, indicator.transform.rotation);
                }
            }
        }
    }
}
```

CarController.cs

맞아??

2.1-3 자동차 모델링 조작하기

2. 지형 인식을 이용한 자동차 카탈로그

◆ 터치 스와이프(Swipe)로 모델링 회전시키기

- 다시 빌드해서 기능 테스트 실시
- 회전 속도와 회전 방향을 확인
- 오브젝트 위치 재배포치 시 1미터 이상 거리 차이가 있어야 재배포치가 되는지 확인

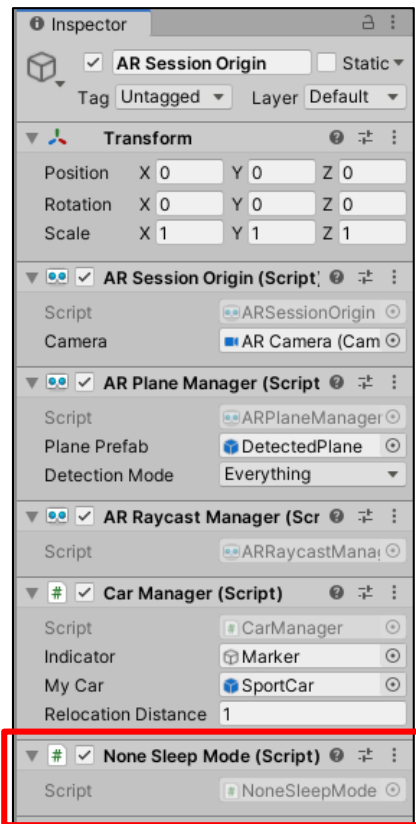


2.1-3 자동차 모델링 조작하기

2. 지형 인식을 이용한 자동차 카탈로그

◆ 스마트폰에서 앱 실행 중에 꺼짐 방지

- 일반적으로 스마트폰에는 배터리 소모율을 줄이기 위해, 일정 시간 동안 사용자의 입력이 없으면, 디스플레이 라이팅을 어둡게 하고, 그 후에도 일정 시간 동안 사용자의 입력이 없으면, 디스플레이를 꺼버리는 절전 모드(Power saving mode)가 설정되어 있음
- AR 프로젝트를 실행하다 보면, AR 카메라가 사물을 인식하거나 하는데, 어느 정도 시간이 소요되기 때문에 슬립 상태로 전환되지 않도록 스크립트를 추가
- “NoneSleepMode”라는 이름으로 C# 스크립트를 생성하고 AR Session Origin 오브젝트에 드래그 앤 드롭하여 추가



2.1-3 자동차 모델링 조작하기

2. 지형 인식을 이용한 자동차 카탈로그

◆ 스마트폰에서 앱 실행 중에 꺼짐 방지

- 디스플레이 절전에 대한 설정은 Screen 클래스의 sleepTimeout 속성 변수를 사용
- 절전 시간(초)를 입력하면 앱 실행 중 절전 모드로의 전환 시간이 변경됨
- 절전 기능을 끄려면 -1 또는 SleepTimeout.NeverSleep(상수)을 입력

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

public class NoneSleepMode : MonoBehaviour
{
    void Start()
    {
        // 앱 실행 중에는 절전 모드로 전환되지 않도록 설정한다.
        Screen.sleepTimeout = SleepTimeout.NeverSleep;
    }
}
```

CarController.cs