가상현실 중간고사 정리

▼ ✅중간고사 Tip.

1. 실습 손코딩 문제 1~2개 있을 것
2. Metallic / Smoothness 차이

   Metallic - 0 비금속, 1 금속
   Smoothness(재질의 거칠기) - 0 표면이 거칠어 난반사, 1 표면이 매끄러워 정반사
3. HMD가 무엇인가? 정확한 풀네임 설명하기
4. HMD 종류 3가지만 대시오.
5. **메타버스 정의 암기**
   a. 메타버스는 초월을 뜻하는 메타(Meta)와 세상을 의미하는 유니버스(Universe)의 합성어로, 컴퓨터 기술을 통해 가상과 현실의 경계를 허물며 구현한 3차원 가상세계를 의미함.
6. 총 문제 수는 10~15개
7. 문제 수 10 문제 이상이면 쉽고, 10 문제 보다 적으면 어려움

▼ 📌예시 문제 정리

1. VPL Research를 설립한 ___는 가상현실 응용프로그램을 최초로 상용화하고, 가상현실 용어를 대중화시켰다. ___는 누구인가?

   답) 재런 러니어(Jaron Lanier)

2. 가상현실의 아버지라고 불리는 모턴 하일리그는 1957년 양 눈에 서로 다른 영상을 비춰서 입체감을 제공하는 장비를 개발하였다. 해당 장비는 무엇인가?

   답) 센소라마 시뮬레이터(Sensorama Simulator)

3. VR의 필수 요소가 아닌 것은?

   a. 몰입감(immersion)
   b. 임장감(presence)
   c. 상호작용(interactivity)
   d. 흥미성(interest)

   답) d. 흥미성

   풀이) VR 필수 요소: 몰입감, 임장감, 상호작용

4. 이반 서덜랜드가 제안한 디스플레이와 추적 시스템이 결합된 개념을 바탕으로 최초의 HMD로 평가받는 장비는 무엇인가?

   답) The Sword of Damocles

5. VR 구성요소 중에서, 사용자의 정보를 월드 데이터베이스에 반영하여 실시간으로 새롭게 변화된 3D 영상과 음향을 재생하는 것은 무엇인가?

   a. 입력부
   b. 출력부
   c. 시각화 시스템
   d. 네트워크

   답) c. 시각화 시스템

6. VR 입력 장치 중에서, 3D 동작 추적 센서 또는 영상 처리를 통해 사용자의 머리 움직임을 추적하는 장치는 무엇인가?

   a. 데이터 글러브
   b. 3D 마우스
   c. 헤드 트래커
   d. 모션 센서/트래커

   답) c. 헤드 트래커

7. 게임 엔진 중에는 Unity는 VR 저작 도구로 주로 사용된다. Unity를 통해 작성된 콘텐츠는 크게 3가지 단위를 통해 구조화되어 있는데, 해당하지 않은 것은?

a. 프로젝트

b. 씬

c. 게임 오브젝트

d. 컴포넌트

답) a. 프로젝트 (2주차 강의자료)

8. Unity에서 재사용이 가능한 게임 오브젝트를 ___라고 한다. ___를 통해 게임 오브젝트를 생성 후 ___를 변경하면 생성된 게임 오브젝트도 변경사항이 반영된다. ___는 무엇인가?

   답) 프리팹(Prefab)

9. 대표적인 VR 출력 장치로, Google에서 개발하였으며 매우 낮은 가격을 통해 VR 대중화에 기여한 장비는 무엇인가?

   답) Google Cardboard

10. VR 필수 요소 3가지와 각 요소들에 대한 간단한 설명을 적으시오.

    답) VR의 필수 요소는 몰입감(Immersion), 임장감(Presence), 상호작용(Interactivity)

    - 몰입감(Immersion): 사용자가 가상 환경에 빠져들어 실제와 같은 경험을 할 수 있도록 만드는 기술 요소입니다.
    - 임장감(Presence): 가상 환경에서 실제로 존재하는 것 같은 느낌을 주는 기술 요소입니다.
    - 상호작용(Interactivity): 사용자가 가상 환경 속에서 상호작용할 수 있는 기능을 제공하는 기술 요소입니다.

11. HMD의 정의와 장단점을 간단히 적으시오.

    답) HMD(Head-Mounted Display)란 머리에 장착하는 디스플레이 장치로, 사용자가 가상현실을 경험하기 위해 시각적으로 제공해주는 영상 출력 장치이다.

    장점:

    - 현실감 있는 가상현실 경험 제공
    - 시야의 자유도가 높음
    - VR 콘텐츠에 대한 집중도 높일 수 있음

    단점:

    - 높은 가격: 품질 높은 HMD는 가격이 비싸고 대부분 추가적으로 컴퓨터나 스마트폰 등의 기기가 필요
    - 불편함: 무게가 무거워 불편할 수 있고, 사용자 눈에 부담될 수 있음

- 운동장애가 있는 사용자에게 불편: 운동장애가 있는 사용자의 경우 HMD 사용이 어려울 수 있음

12. 전역 좌표계와 지역 좌표계를 각각 간단히 설명하시오.

    답) 전역 좌표계: 프로그램 내 가상 공간을 중심으로 갖는 좌표계

    지역 좌표계: 오브젝트를 중심으로 갖는 좌표계

    (4주차 강의자료)

13. 툴바의 변환 기즈모 토글 버튼 중, Center ↔ Pivot 버튼은 무슨 역할을 하는가? Center 모드 일 때와, Pivot 모드 일 때의 차이점을 서술하시오.

    답) Center 모드일 때는 오브젝트의 중심(중점)을 기준으로 변환 수행,

    Pivot 모드일 때는 오브젝트의 실제 기준점, 그 위치를 기준으로 변환 수행

14. Hierarchy에서 오브젝트를 선택하여, 다른 오브젝트에 드래그하면 어떤 일이 발생하는가?

    답) 선택한 오브젝트가 드롭한 오브젝트의 자식 오브젝트로 이동

15. Scene을 저장 시 저장되는 것을 모두 고르시오.

    a. Scene 및 Hierarchy 내 현재 오브젝트 배치
    b. 프로젝트 내 관련 Scene 및 Asset 묶음 모두
    c. Inspector 설정, 스크립트, Asset, Prefab 구조
    d. Object Transform, 프로젝트 설정, 실행 취소 내역

    답) a, c, d

16. 다음 스크립트에서의 문구는 각각 무엇을 의미하는가?

    a. public Vector3 vect;
    b. public Transform trans;

    답) public Vector3 vect; - public 접근 제어자인 Vector3 타입의 변수 vect 선언

    public Transform trans; - public 접근 제어자인 Transform 타입의 변수 trans 선언

17. Collider 컴포넌트의 ___옵션을 활성화하게 되면, Rigidbody 컴포넌트와 충돌 시 프로그래머가 작성한 스크립트로 충돌 처리를 할 수 있다. ___는 무엇인가?

    답) Is Trigger

18. 유니티의 스크립트 중, 매 프레임 별로 실행되는 함수는 (A)이며, 오브젝트의 활성 여부에 상관 없이 항상 1회 출력하여 생성자 역할을 하는 함수는 (B)이다. A과 B는 무엇인가?

    답) A: Update(), B: Awake()

19. Tool Bar의 Controller 중, 오브젝트의 위치, 회전, 크기를 제어하는 Controller는 무엇인가? (2주차 강의자료)

    답) Transform 도구

20. Tool Bar의 Controller 중, Pivot-Center 상태 변화 및 Local-Global 좌표계 변환을 담당하는 Controller는 무엇인가? (2주차 강의자료)

    답) Transform 기즈모 토글

21. Hierarchy View에서 오브젝트를 다른 오브젝트에 드래그 & 드롭을 할 때 형성되는 관계를 무슨 관계라고 하는가?

    답) Parent-Child Relationship(부모-자식 관계)

22. 모델링의 정의를 적으시오.

    답) 모델링이란 그래픽으로 표현하고자 하는 물체를 정의하는 작업(4주차 강의자료)

23. 3D 엔진의 메쉬는 다각형 평면보다 주로 삼각형 평면을 사용한다. 그 이유는 무엇인가?

    답) 삼각형 평면이 다각형 평면보다 처리 속도가 2배 빠르기 때문이다. (4주차 강의자료 p.7)

24. 카메라의 원근 모드, 직교 모드는 ___이/가 적용되거나 무시되는 모드다.

    답) 원근법 (4주차 강의자료 p.12)

25. 유니티의 화면 구성은 크게 6가지로 구성되어 있다. 구성 요소를 각각 간단히 설명하시오.

    답) (메뉴, 툴바,) 씬뷰, 게임뷰, 계층창, 인스펙터창, 프로젝트 창, 콘솔창

    (2주차 강의자료)

    1. 씬뷰: 씬을 구성하는 게임 오브젝트를 배치할 때 사용
    2. 게임뷰: 씬 내에 카메라 게임 오브젝트가 렌더링한 장면을 나타냄
    3. 계층창: 현재 씬 뷰에 불러와진 게임 오브젝트의 계층을 관리
    4. 인스펙터창: 게임 오브젝트, 컴포넌트, 에셋 등을 설정할 때 사용
    5. 프로젝트창: 크게 좌측 패널과 우측 패널로 구성됨. 좌측 패널은 폴더 구조 목록 표시. 우측 패널은 좌측 패널에서 선택된 폴더가 포함하고 있는 폴더와 파일 표시.

6. 콘솔창: 프로그래밍을 통해 작성된 스크립트나 프로젝트/씬/콘텐츠의 오류/경고/기타메시지 등을 나타냄

26. HMD란 무엇을 의미하는지 설명하고, HMD의 예시 사례를 3가지 이상 작성하시오.

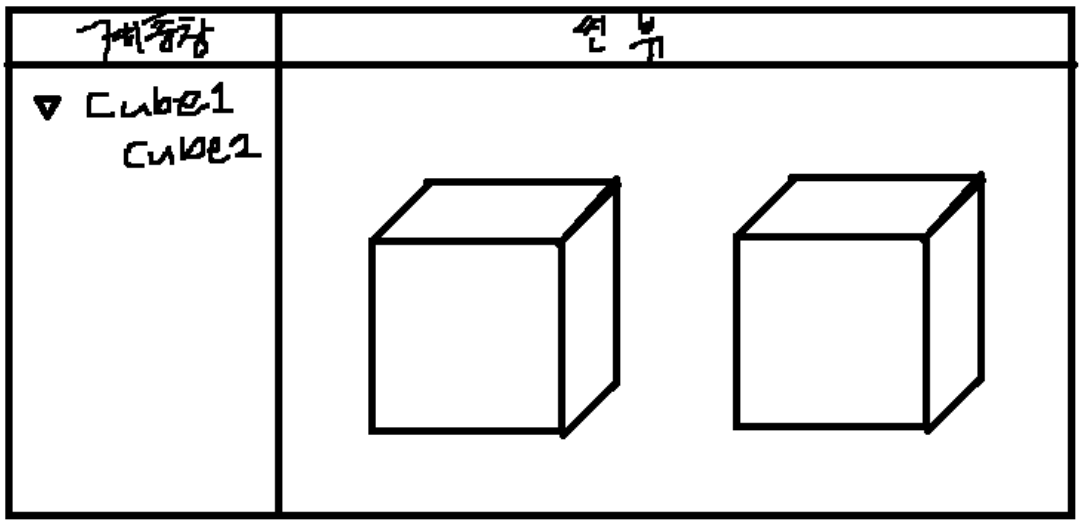
    답) HMD(Head-Mounted Display)란 머리에 장착하는 디스플레이 장치로, 사용자가 가상현실을 경험하기 위해 시각적인 입력을 제공해주는 영상 출력 장치이다.

    1. Oculus Rift: PC 기반 HMD
    2. HTC Vive: SteamVR 플랫폼을 지원하는 HMD
    3. PlayStation VR: PS4 콘솔과 호환되는 HMD

27. 컴퓨터 그래픽에서 전역 좌표계(또는 월드 좌표계)는 프로그램 내 가상 공간의 기준이 되는 좌표계이다. 해당 좌표계를 기준으로 가상 객체들의 위치/회전/크기 상태가 결정되는데, 이때 가상 객체가 가지며, 자신을 참조하는 좌표계를 지역 좌표계라고 한다. 가상 객체가 다른 가상 객체와 (1)-(2) 관계가 형성되면, (2) 가상 객체는 (1) 가상 객체의 지역 좌표계를 참고하게 된다. 이 결과, (1) 가상 객체의 이동/회전/크기 상태를 변경하게 되면, 지역 좌표계에도 영향이 가기 때문에 (2) 가상 객체의 위치/회전/크기 상태도 변하게 된다. (1), (2) 는 무엇을 의미하는가?

    답) (1)은 부모(Parent) 가상 객체를, (2)는 자식(Child) 가상 객체를 의미합니다. 자식 가상 객체는 부모 가상 객체의 지역 좌표계를 참조하게 됩니다.

28. Unity 에디터에서 프리미티브 게임 오브젝트 중 Cube를 이용하여 다음과 같은 간단한 게임 오브젝트들을 설계하였다. (위치/회전/크기의 값은 인스펙터 창에 나타난 값을 의미함)



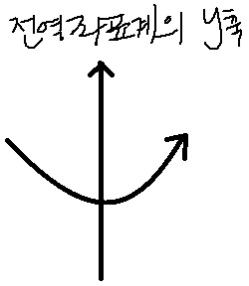
Cube1	값
위치	0, 0, 0
회전	0, 0, 0

Cube2	값
위치	2, 0, 0
회전	0, 0, 0

크기	1, 1, 1

크기	1, 1, 1

Cube1 게임 오브젝트를 회전시키려고 할 때, 전역 좌표계의 y축을 기준으로, Transform 도구의 회전 도구를 사용하여 다음과 같이 Cube1을 60도 회전시켰다.



회전할 때, 툴바의 Transform 기즈모 토글의 모드가 Center, Global 이었다면 회전 이후 Cube1과 Cube2의 위치는 어떤 값을 갖는가?(총 10점, 각 5점, 수식으로 표현 가능)

답)

Cube1 값
위치 cos(60)=0.5, 0, sin(60)=0.866
회전 0, 60, 0
크기 1, 1, 1

Cube2 값
위치 2, 0, 0
회전 0, 0, 0
크기 1, 1, 1

e.g, 만약 x축 기준으로 돌면 위치 값 변화X x=0, y=0, z=0

z축 기준으로 돌면 위치 x=cos(60), y=-sin(60), z=0

29. 스크립트를 이용하여 실시간으로 마우스/키보드 이벤트를 처리할 수 있는 Unity 콘텐츠를 제작하고자 한다. 생명 주기를 이루는 메서드 중 어떠한 메서드를 사용하는 것이 바람직한가?

    답) Update 메서드

30. Unity 에디터에서 기본적으로 사용되는 표준 쉐이더 (Standard Shader) 기반 재질을 이용하고자 한다. 이때, 해당 재질의 Metallic과 Smoothness 속성의 값은 무엇을 의미하는가? (3주차 강의자료)

    a. Metallic - 값 0
    b. Metallic - 값 1
    c. Smoothness - 값 0

d. Smoothness - 값 1

답)

Metallic(재질의 금속성) - 0 비금속, 1 금속

Smoothness(재질의 거칠기) - 0 표면이 거칠어 난반사, 1 표면이 매끄러워 정반사

Albedo(색상, 텍스처 등 게임 오브젝트의 표면을 결정)

31. Unity에서 특정 아이템을 먹으면 플레이어가 조작하는 게임 오브젝트에 PowerUP이라는 사용자 정의 컴포넌트가 추가되며, 일정 시간이 지나면 추가된 PowerUP 컴포넌트가 제거되도록 콘텐츠를 제작하였다. 이때, 새로운 스크립트를 생성하고, 해당 스크립트의 HasPowerUp이라는 메서드를 이용하여 플레이어가 조작하는 게임 오브젝트에 PowerUP이라는 사용자 정의 컴포넌트가 존재하고 있는지 검사하고자 한다. HasPowerUP 메서드가 어떻게 구현되어야 하는지 작성하시오. (파라미터로는 사용자가 조작하는 게임 오브젝트인 player가 제공된다.)

```
bool HasPowerUP(Transform player)
{
    PowerUP powerUp = player.GetComponent<PowerUP>();
    if (powerUp != null)
    {
        return true;
    }
    else
    {
        return false;
    }
}
```

위 코드에서는 Transform 타입으로 받은 player 오브젝트의 GetComponent 메서드를 이용하여 PowerUP 컴포넌트를 찾아내고, 해당 컴포넌트가 null이 아닐 경우 PowerUP이 존재하는 것으로 판단하여 true를 반환합니다. PowerUP이 존재하지 않을 경우 false를 반환합니다.

32. VR의 구성요소는 크게 5가지로 구성되는데, 5가지 구성요소를 각각 간단히 설명하시오.

답) (2주차 강의자료: 핵심기술 및 적용 사례)

입력부 - 참여자의 위치, 방향 및 행위로 인한 이벤트 정보를 전송

출력부 - 입력정보에 대응하는 3차원 영상, 음향, 촉각 등의 출력을 전달

시각화시스템 - 참여자의 정보를 월드 데이터베이스에 반영하여 실시간으로 새롭게 변화한 3D 영상과 음향을 재생

네트워크 - VR 시스템에서 데이터를 송신, 수신

데이터베이스 - 가상현실과 관련된 데이터를 저장, 관리

33. Unity Asset이 무엇인지 설명하고, Asset의 장점에 대해서 서술하시오.

답) (3주차 강의자료) Asset이란 콘텐츠를 제작할 때 사용되는 자원을 의미한다.

Asset의 장점은 다음과 같다.

(1. 개발 시간 단축, 2. 높은 퀄리티, 3. 비용 절감, 4. 높은 생산성)

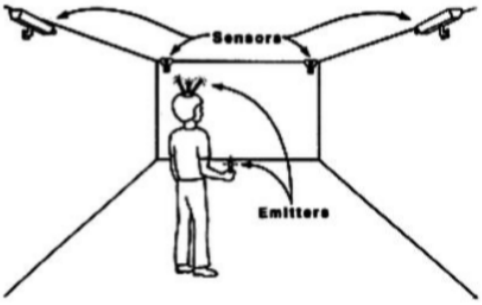
1. 이미 만들어진 자원(Asset)을 사용하기 때문에 개발 시간을 단축 시킬 수 있다.
2. Asset Store에서 판매되는 높은 퀄리티의 Asset을 사용할 수 있다.
3. Asset을 사용하면 개발자가 많은 비용을 들여 자원을 만들지 않아도 된다.
4. Asset은 자원을 손쉽게 사용할 수 있도록 하므로 생산성을 높여준다.

34. 3D 그래픽 기반의 VR 개발을 위한 4가지의 핵심 개념들에 대해서 각각 간략히 설명하시오.

    답)

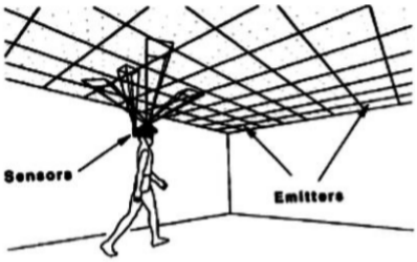
    1. 모델링: 그래픽으로 표현하고자 하는 물체를 정의하는 작업
    2. 좌표계: 물체의 위치를 특정한 하나의 점으로써 유일하게 가리키기 위한 체계
    3. 카메라: 출력 장치에 출력할 영상을 생성
    4. 렌더링: 카메라가 보고 있는 오브젝트들을 이미지로 변환하여 그려내는 작업

35. 각 그림의 VR 트래킹 방식의 명칭과 해당 각 트래킹 방식에 대해서 설명하시오.



아웃사이드-인 트래킹

외부 센서가 직관적으로 사용자의 위치를 측정하기 때문에 정확하지만 설치가 번거롭다.



인사이드-아웃 트래킹

HMD에 트래킹 장치가 부착되어있어 HMD 자체에서 사용자의 위치를 측정합니다.

예) HMD앞에 카메라가 달려있는 경우

36. (A)는 2개 이상의 카메라를 이용하여 3차원 마커의 위치 및 방향을 측정하며, 3차원 마커의 위치 및 방향을 (B)으로 추출한다. 이는 인식 속도가 빠르고 정확하다는 장점이 있지만, 사용자에게 마커를 부착하고 고비용의 카메라가 필요하다는 단점이 있다. A와 B는 무엇인가?

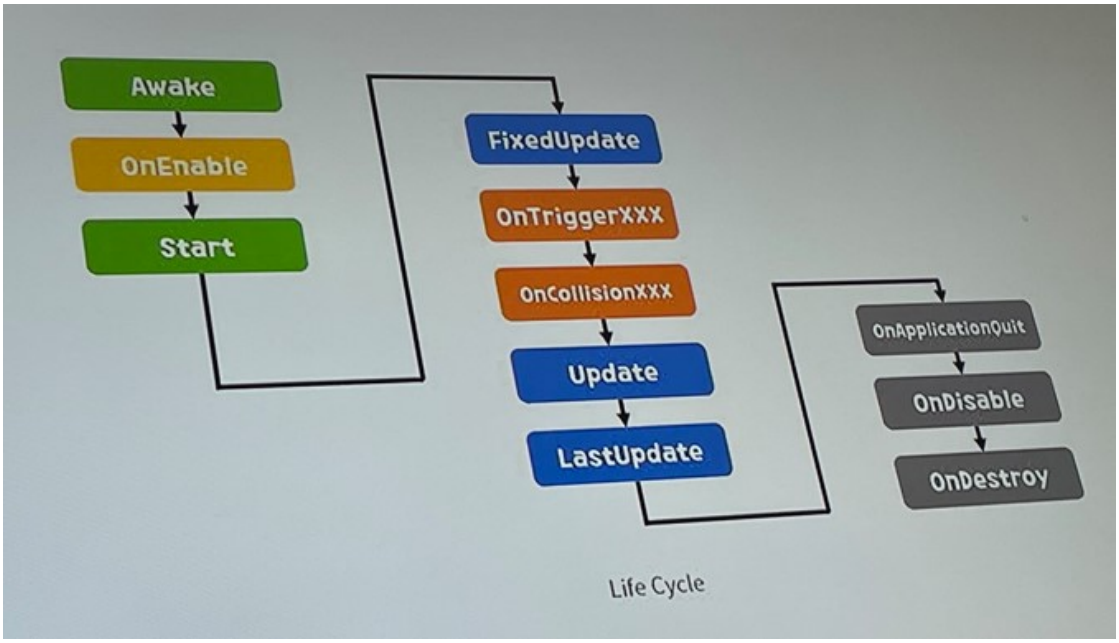
    답) A: 다중 카메라 기반 마커 트래킹, B: 삼각측량법

37. 다음 나열된 이벤트 함수를 Unity 라이프 사이클에 따라서 가장 먼저 실행되는 이벤트 함수부터 마지막 이벤트 함수까지 순차적으로 나열하시오.

```
FixedUpdate, OnDisable, Start, OnDestroy, Awake, Update, LastUpdate
```

답) Awake - Start - FixedUpdate - Update - LateUpdate - OnDisable - OnDestroy

풀이)



Awake - OnEnable - Start - FixedUpdate - OnTriggerXXX - OnCollisionXXX - Update - LastUpdate - OnApplicationQuit - OnDisable - OnDestroy

38. 유니티에서 객체의 회전은 내부적으로 Vector3가 아닌 (A)로 사용된다. 이는 회전 시, 좌표 측이 겹쳐 정보가 손실되는 (B) 현상 때문이다. A와 B는 무엇인가?

답) A: Quaternion(쿼터니언), B: Gimbal lock(짐벌락)

39. 어떤 Cube 오브젝트가 게임 씬 내에 존재한다. 현재 Cube 오브젝트의 Rotation 값을 모르는 상태이며, 게임 실행 시, Cube 오브젝트의 현재 회전에 추가적으로 x축으로 30도만 좀 더 회전시키려고 한다. 아래의 스크립트는 Cube 오브젝트에 추가된 스크립트이다. 아래의 Start 함수 내에 빈칸을 채우시오.

```
void Start()
{
  Quaternion cubeRotation = [A] // Cube 오브젝트의 회전을 저장
  Vector3 cubeRotationVec = [B] // cubeRotation을 Vector3로 변경
  Vector3 AddRotationVec = cubeRotationVec + new Vector3(30,0,0);
  transform.rotation = [C] // AddRotationVec을 쿼터니언으로 변경 후 지정
}
```

답)

A: transform.rotation;

B: cubeRotation.eulerAngles; //cubeRotation의 현재 회전 값을 넣음

C: Quaternion.Euler(AddRotationVec);

40. 방향키(WASD) 키입력을 받아서 Player를 만들려고 한다. 이때, 아래의 스크립트는 Player 오브젝트에 추가할 스크립트로 빈칸을 채우시오. (단, WASD를 조건별로 나누지 말 것)

```
void Update()
{
  float inputX = [A] //수평방향
  float inputY = [B] //수직방향
  transform.GetComponent<Rigidbody>().velocity = new Vector3(inputX*10f,0,inputY*10f)
}
```

답

A: Input.GetAxis(”Horizontal”);

B: Input.GetAxis(”Vertical”);

41. Cube 오브젝트가 Space 키입력 받을 시, 계속해서 1초에 y축으로 100도 회전하도록 구현하고자 한다. Cube 오브젝트에 추가하려는 스크립트를 작성하시오.

```
void Update()
{
    if (Input.GetKey(KeyCode.Space))
    {
        transform.Rotate(Vector3.up * 100f * Time.deltaTime);
    }
}
```

위 스크립트는 Space 키를 계속 누르고 있으면, 매 프레임마다 y축으로 100도 회전하는 코드입니다. `Input.GetKey(KeyCode.Space)`
은 Space 키가 눌려져 있으면 true를 반환하며, `transform.Rotate()`
함수를 사용하여 현재 오브젝트를 회전시킵니다. 이때, `Vector3.up`
은 y축을 의미하며,

`Time.deltaTime`을 곱해주어 시간에 따른 회전을 부드럽게 처리합니다.

42. 다음 나열된 VR 트래킹에서 센서 기반 트래킹(A)과 비젼 기반 트래킹(B)을 분류하시오.

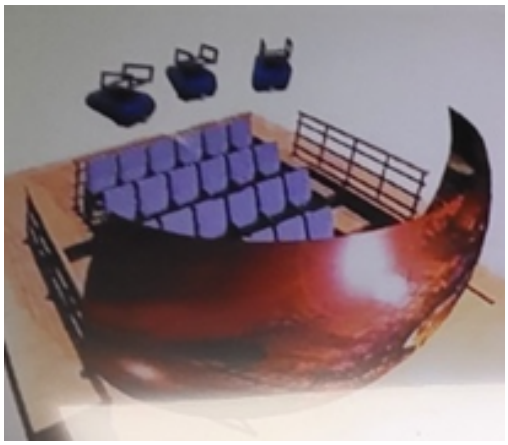
> 라이트하우스 트래킹 시스템, 관성항법센서, 다중카메라 기반 마커 트래킹, Leap Motion, 전자기 트래커, 기계식 관절 트래커, Kinect

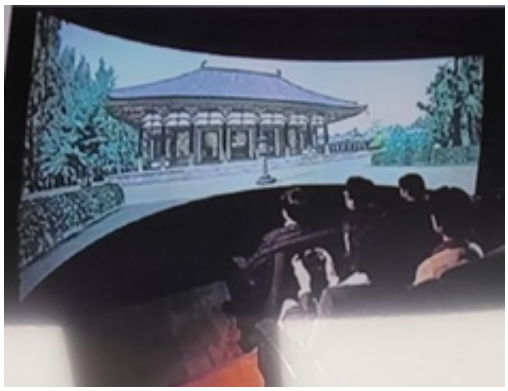
답)

센서 기반 트래킹(A): 기계식 관절 트래커, 전자기 트래커, 관성항법센서, 라이트하우스 트래킹 시스템

비젼 기반 트래킹(B): 다중카메라 기반 마커 트래킹, Leap Motion, Kinect, Project Tango

43. 아래의 그림은 VR을 제공하는 Output Device 중에 한 형태이다. 아래의 Output Device는 어떤 이름(형식)인가.



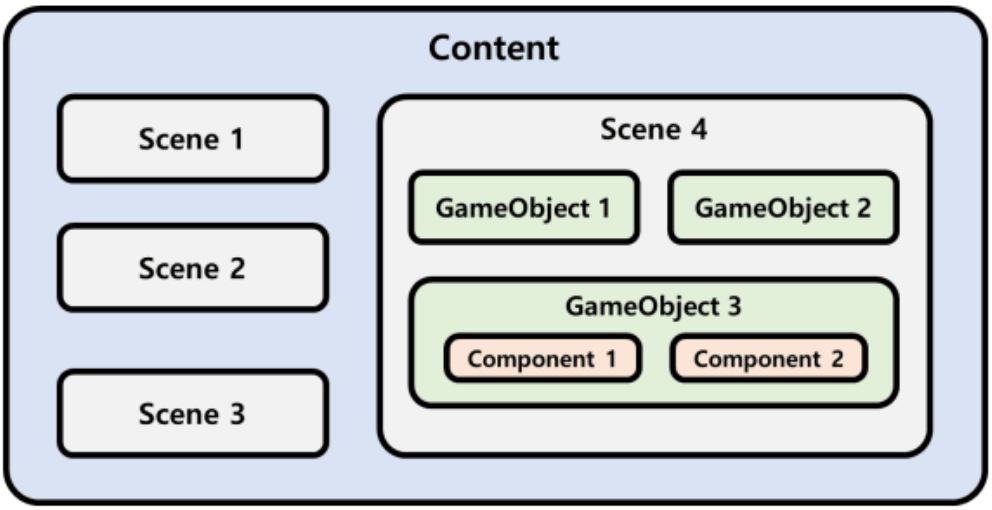


답) Curved Projection, 곡면 디스플레이

▼ 🚩강의 자료 정리

<유니티 구조>

- 씬: 콘텐츠를 구성하는 단위, 게임 오브젝트들을 포함하는 객체
- 게임 오브젝트: 씬을 구성하는 단위, 컴포넌트들을 포함하는 객체
- 컴포넌트: 게임 오브젝트를 구성하는 단위, 독립적인 기능 수행



<유니티 인터페이스 화면 구성>

1) 메뉴 : File, Edit, Assets, GameObject, Component, Windows, Help

2) 툴바: Transform 도구, Transform 기즈모 토글, 플레이 모드, 콜라보레이트, 서비스, 계정, 레이어, 레이아웃

3) 씬뷰: 씬을 구성하는 게임 오브젝트를 배치할 때 사용

4) 게임뷰: 씬 내에 카메라 게임 오브젝트가 렌더링한 장면을 나타냄

5) 계층창: 현재 씬 뷰에 불러와진 게임 오브젝트들의 계층 관리

6) 인스펙터창: 게임 오브젝트 / 에셋 / 컴포넌트 설정할 때 사용

7) 프로젝트창: 좌측 패널: 폴더 구조 목록 표시, 우측 패널: 좌측 패널에서 선택된 폴더가 포함하고 있는 폴더, 파일 표시

8) 콘솔창: 프로그래밍을 통해 작성된 스크립트나 프로젝트/씬/콘텐츠의 오류/경고/기타 메시지를 나타내는 곳

<VR 핵심 기술>

- VR 기술은 디스플레이나 인식 및 가시화와 같은 소프트웨어 기술 이외에도 다양한 분야의 기술들이 종합적이고 유기적으로 융합된 복합 기술임.
- 주요 기술 분야:
  - Brain Science
  - Sociology
  - Ergonomics
  - Human Computer Interaction

- Haptics
- Mechanical Engineering
- Robotics
- 3D Displays
- Computer Graphics(Real time, interactive)
- Software Engineering
- Computer Vision
- Motion Tracking& Recognition
- 3D Interfaces
- 3D Sounds
- Psychology

<VR 사례>

- 군수
- 의료: 가상 의료(수술, 진단)
- 엔터테인먼트: 가상 스튜디오
- 엔터테인먼트: 테마파크 - 4D Rider
- 산업: 가상 모델 하우스
- 산업: 가상 피팅룸
- 산업: 로봇 시뮬레이션
- 교육: 가상 운전 교육 시뮬레이터
- 교육: 가상 PLC 교육 시스템(시뮬레이터)
- 교육: 촉각 장치

<게임 엔진>

- 게임 엔진은 그래픽을 시각화 하기 위한 그래픽 엔진, 음향 효과를 위한 사운드 엔진, 물리 연산을 위한 물리 엔진, UI 시스템 등을 포함하며, VR 저작에 주로 사용됨.
- 즉, 게임 엔진 = 그래픽 엔진, 사운드 엔진, 물리 엔진, UI 시스템

<Unity vs Unreal>

구분	Unreal	Unity
장점	- 상대적으로 높은 그래픽 성능 - PC 콘텐츠 제작에 적합	- 낮은 개발 난이도 - 모바일 콘텐츠 제작에 적합
단점	- 높은 개발 난이도 - 고사양 개발 환경 필요	- 상대적으로 낮은 그래픽 성능
라이선스	- 무료 - 연간 수익 100만 달러 이	- 무료 - 월 $40 (매출 20만 달러 이하) - 월 $150 (매출

	상 시, 각 콘텐츠에 5% 로열티 지불	20만 달러 이상) - 연 $4,000 (솔루션 제공, 기업용)

<Unity 선호하는 이유>

- 회사 전년도 매출이 10만 달러 이하일 경우 라이선스가 무료
- Unreal 과 달리 로열티가 없음
- 다양한 플랫폼으로 이식이 가능

<VR Applications in Sports>

- 스크린 골프
- 스크린 축구
- 스크린 야구
- 양궁 시뮬레이터
- 승마 시뮬레이터
- 테니스 시뮬레이터
- 탁구 시뮬레이터
- 2018 Winter Olympic (롯데월드몰, 평창 올림픽)

<씬 관리>

- 씬은 “.unity” 확장자를 갖는 파일로 관리됨
- 프로젝트 처음 생성 시, 기본으로 “SampleScene” 씬이 활성화됨
- 생성 화면에서는 빈 씬(empty)과 기본 씬(basic)이 포함됨
  - 빈 씬: 어떤 게임 오브젝트도 구성되지 않은 씬
  - 기본 씬: 카메라 게임 오브젝트와 광원 게임 오브젝트로 구성된 씬
- 자동으로 생성된 SampleScene 씬은 기본 씬(basic)에 해당함

<프리미티브 게임 오브젝트(Primitive GameObject)>

- Unity에 내장된 모델링 소프트웨어에 의해 생성될 수 있는 3D 모델 형태의 게임 오브젝트

<Transform 컴포넌트>

- Unity의 모든 게임 오브젝트는 Transform 컴포넌트를 필수로 가지고 있음
- Transform 컴포넌트는 게임 오브젝트의 위치, 회전, 스케일 속성을 가지고 있음
- 빈 게임 오브젝트는 Transform 컴포넌트만을 가지고 있는 오브젝트임

<재질>

- 컴퓨터그래픽스: 3차원 그래픽에서 3차원 물체의 색, 텍스쳐, 광원 등을 표현하는 방법
- Unity: 3차원 모델을 표현하는 게임 오브젝트를 표현하는 방법

<에셋 스토어 기반 에셋 구매 방법>

- On Sale Only 버튼을 체크 해제

<오브젝트>

- 3D 엔진에서는 모델링을 통해 정의된 물체

<메쉬(Mesh)>

- 모델링을 통해 정의된 평면

<전역 좌표계와 지역 좌표계의 특징>

- 각 오브젝트는 위치/회전/크기 변환 시, 전역 좌표계를 기준으로 수행함
- 지역 좌표계를 기준으로 변환 시에는 전역 좌표계로 일치 시키는 단계가 필요

<VR HMD에서의 렌더링 방법> - 양안을 고려하여 2개의 카메라 필요

1. 한 쪽 눈 렌더링 시작
2. 한 쪽 눈 렌더링 종료 후 다른 쪽 눈 렌더링 시작
3. 렌더링 완료

<유니티 특징>

- 크로스 플랫폼
  - 하나의 개발 결과에서 다양한 플랫폼으로 이식이 가능
- 라이선스 정책
  - 개인 개발자는 월 매출이 10만 달러 이하일 시, 라이선스가 무료
- 통합 에셋 시스템
  - 에셋 스토어(Asset Store)를 통해 다양한 에셋(3D 모델, 애니메이션 등)을 무료 또는 유료로 제공하기 때문에 개발 시간이 단축됨

<스크립트 개념>

- 스크립트 생성 시 기본적으로 “MonoBehaviour” 이라는 클래스를 상속받은 클래스 형태로 스크립트가 생성됨
- MonoBehaviour은 생명 주기(Life Cycle)에 따라서 특정 메서드를 호출한다는 특징이 있음

<스크립트 적용 시 유의점>

- 스크립트 파일 이름과 MonoBehaviour를 상속받은 클래스의 이름이 동일해야 함

<Debug.Log(문자열)>

- 콘솔 창에 문자열 출력하는 메서드
- Start 메서드에서 수행

```
void Start()
{
  Debug.Log("문자열");
}
```

<Fire 스크립트>

- 탱크 오브젝트에 추가
- Bullet Prefab 변수, Spawn Bullet 변수 설정
- Instantiate(오브젝트, 위치, 회전): 오브젝트를 지정한 위치 및 회전에 따라 생성
- Input.GetKeyDown: KeyCode 버튼이 눌렸는지 확인

```
public class Fire : MonoBehaviour
{
  public GameObject bulletPrefab;
  public Transform spawnBullet;

  void Update()
  {
    if(Input.GetKeyDown(KeyCode.Space))
    {
      var fire = Instantiate(bulletPrefab, spawnBullet.position, Quaternion.identity);
    }
  }
}
```

<Bullet 스크립트>

- 총알 오브젝트에 추가
- Translate(Vector3): Vector3의 방향으로 이동

```
public class Bullet : MonoBehaviour
{
  void Update()
  {
    transform.Translate(Vector3.forword);
  }
}
```

<ObjectControlModule 스크립트>

- Start 메서드 사용

```
public class ObjectControlModule : MonoBehaviour
{
    void Start()
    {
        //3차원 값을 저장하는 변수 position에
        //Transform 컴포넌트의 위치 값(position)을 저장
        Vector3 position = transform.position;
        //회전 값을 저장하는 변수 rotation에
        //Transform 컴포넌트의 회전 값(rotation)을 저장
        Quaternion rotation = transform.rotation;
        //3차원 값을 저장하는 변수 scale에
        //Transform 컴포넌트의 크기 값(localScale)을 저장
        Vector3 scale = transform.localScale;
    }
}
```

<오브젝트 이동>

- // 앞 방향으로 3만큼 이동
- 방법 1: transform.position = transform.position + transform.forward * 3;
- 방법 2: transform.Translate(Vector3.forward * 3);
- 단위 벡터 모음

Vector3.forward	Vector3(0,0,1)
Vector3.back	Vector3(0,0,-1)
Vector3.left	Vector3(-1,0,0)
Vector3.right	Vector3(1,0,0)
Vector3.up	Vector3(0,1,0)
Vector3.down	Vector3(0,-1,0)
Vector3.one	Vector3(1,1,1)
Vector3.zero	Vector3(0,0,0)

<PlayerController1 스크립트>

- Player 오브젝트에 추가
- WASD 방향키에 따라 움직임

```
public class PlayerController1 : MonoBehaviour
{
  void Update()
  {
    if(Input.GetKey(KeyCode.W)
    {
      transform.Translate(Vector3.forward * Time.deltaTime * 10);
    }
    if(Input.GetKey(KeyCode.S)
    {
      transform.Translate(Vector3.back * Time.deltaTime * 10);
    }
    if(Input.GetKey(KeyCode.A)
    {
      transform.Translate(Vector3.left * Time.deltaTime * 10);
    }
    if(Input.GetKey(KeyCode.D)
    {
      transform.Translate(Vector3.right * Time.deltaTime * 10);
    }
  }
}
```

<PlayerController2 스크립트>

- Player 오브젝트에 추가
- WASD 방향키에 따라 움직임
- Input.GetAxis() 메서드 사용

```
public class PlayerController2 : MonoBehaviour
{
  public float speed = 10f;

  void Update()
  {
    //수평 방향의 키보드 입력 시, -1 ~ 1
    float inputX = Input.GetAxis("Horizontal");
    //수직 방향의 키보드 입력 시, -1 ~ 1
    float inputZ = Input.GetAxis("Vertical");

    Vector3 player = new Vector3(inputX * speed, 0, inputZ * speed);
    //velocity 변수(Vector3) -> 속도를 지정
    transform.GetComponent<Rigidbody>().velocity = player;
  }
}
```

<물리 엔진>

- 물리 엔진은 오브젝트의 중력 또는 오브젝트 간의 충돌과 같은 물리적 현상을 시뮬레이션

- 물리 엔진 사용을 위한 2가지 조건
  - 물리적 현상이 적용될 오브젝트는 강체(Rigidbody) 컴포넌트가 필요
  - 강체는 충돌체(Collider) 컴포넌트를 가진 오브젝트에게만 충돌 등이 발생

<물리 엔진 적용: 충돌체 설정>

- Box Collider 컴포넌트 추가
- 충돌 영역이 생성됨
- 충돌체(Collider)인 몬스터(Monster)에 Box Collider 컴포넌트 추가함

<물리 엔진 적용: 강체 설정>

- Rigidbody 컴포넌트 추가
- 강체(Rigidbody)인 총알(Bullet)에 Rigidbody 컴포넌트 추가함

<강체의 모드에 따른 물리 엔진>

- Not Trigger 모드: 유니티 기본 물리 엔진 수행
- Trigger 모드: 프로그래머가 직접 충돌 이벤트 처리
  - 충돌 이벤트 직접 처리: Sphere Collider 컴포넌트의 “Is Trigger” 체크
  - 강체(Rigidbody)인 총알(Bullet)에 있는 Is Trigger 체크함

<Fire2 스크립트>

- 강체에서의 이벤트 처리
- 탱크 오브젝트에 추가
- Spawn Bullet 변수, Bullet 변수 설정

```
public class Fire2 : MonoBehaviour
{
  private int bulletPower = 1000;
  public Transform spawnBullet;
  public GameObject bullet;

  void Update()
  {
    if(Input.GetKeyDown(KeyCode.Space))
    {
      // 총알 생성
      GameObject bullet_Instance = Instantiate(bullet, spawnBullet.position, spawnBullet.rotation);
      // 생성된 총알의 Rigidbody 컴포넌트 불러옴
      Rigidbody bullet_Rigidbody = bullet_Instance.GetComponent<Rigidbody>();
      // 물리적인 힘 부여
      bullet_Rigidbody.AddForce(bullet_Rigidbody.transform.forward * bolletPower);
    }
```

```
  }
}
```

<BulletEventModule 스크립트>

- 강체에서의 이벤트 처리
- 총알 오브젝트에 추가

```
public class BulletEventModule : MonoBehaviour
{
  private void onTriggerEnter(Collider other)
  {
    // 충돌한 오브젝트의 이름이 Monster일 경우
    if(other.gameObject.name === "Monster")
    {
      // 충돌한 오브젝트 파괴
      Destroy(other.gameObject);
      // 자기 자신도 파괴
      Destroy(this.gameObject);
    }
  }
}
```

<CollisionDetectModule 스크립트>

- 충돌체에서의 이벤트 처리
- Return 오브젝트에 추가

```
public class CollisionDetectModule : MonoBehaviour
{
  private void onCollisionEnter(Collision collision)
  {
    // 충돌한 오브젝트의 이름이 탱크일 경우
    if(collision.gameObject.name === "탱크")
    {
      collision.transform.position = Vector3.zero;
    }
  }
}
```

<강체에서의 이벤트 처리>

함수	파라미터	용도
onTriggerEnter();	Collider(충돌체)	강체가 충돌체와 충돌이 시작될 때 감지
onTriggerStay();	Collider(충돌체)	강체가 충돌체와 충돌 중일 때 감지
onTriggerExit();	Collider(충돌체)	강체가 충돌체와 충돌이 종료될 때 감지

<충돌체에서의 이벤트 처리>

함수	파라미터	용도
onCollisionEnter();	Collision(충돌정보)	충돌체에 강체가 들어올 때 감지
onCollisionStay();	Collision(충돌정보)	충돌체에 강체가 머물 때 감지
onCollisionExit();	Collision(충돌정보)	충돌체로부터 강체가 나갈 때 감지

<Q, E 키 입력으로 포탑 회전>

```
if(Input.GetKey(KeyCode.Q)
{
  FireObject.transform.Rotate(Vector3.down * Time.deltaTime * 100);
}
if(Input.GetKey(KeyCode.E)
{
  FireObject.transform.Rotate(Vector3.up * Time.deltaTime * 100);
}
```

<CameraControl 스크립트>

- 마우스를 통해 카메라 회전

```
public class CameraControl : MonoBehaviour
{
  public float RotationSpeed = 200;

  void Update()
  {
    float xInput = Input.GetAxis("Mouse X");
    float yInput = Input.GetAxis("Mouse Y");

    Vector3 rotationDirection = new Vector3(-yInput, xInput, 0);
    // 오일러 각(x,y,z) 기반 회전
    transform.eulerAngles += rotationDirection * Time.deltaTime * RotationSpeed;
  }
}
```