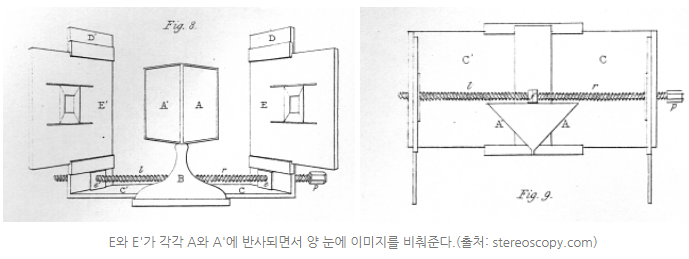
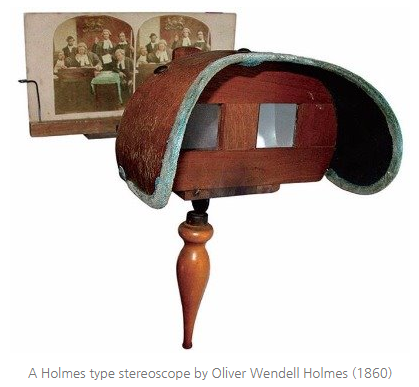
VR프로젝트 중간고사 정리

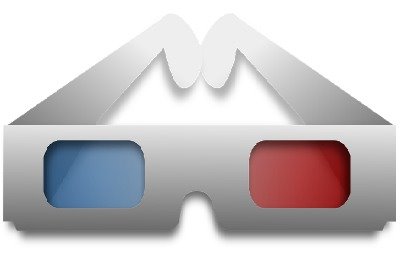
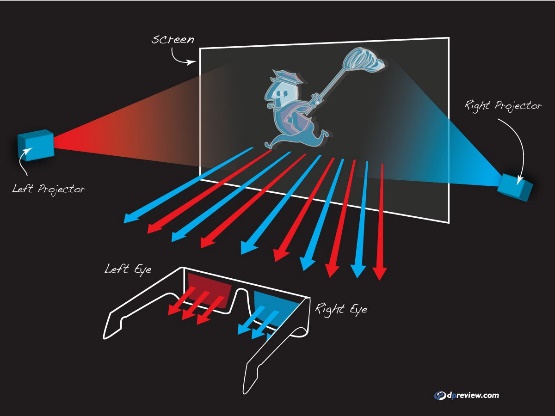
1. 객관식 5문제(각 1점), 주관식 5문제(각 2점), 서술형 3문제(각 5점)
2. VR 역사 (인물, 기술, 용어, 등)
3. VR기기 구성
4. 유니티 화면 및 메뉴, 오브젝트
5. 클래스 멤버 접근 수식자
6. c# 스트립트 종류 이해(제너레이터, 컨트롤러, 감독)
7. 프리팹 내용
8. 게임 설계 절차 5단계
9. UI 내용
10. VR 관련 역사 인물 및 기술 방식
11. VR(Virtual Reality)
12. 가상 현실의 의미
13. : 가상(사실이 아님) + 현실(실제로 존재하는 사실이나 상태)
14. 인공적인 기술로 만들어낸 실제와 유사하지만 실제가 아닌 어떤 특정한 환경이나 상황 혹은 그 기술 자체
15. 가상 현실의 개념 대중화
16. 미국의 컴퓨터 과학자이자 현재 마이크로소프트 연구소에서 재직 중인 재런 래니어(Jaron Lanier)
17. 1985년 VPL Research를 설립해 HMD와 장갑 형태의 가상현실용 컨트롤러를 개발
18. 실감 나는 이미지와 영상 (1838~1950)
19. Stereoscopic Vision



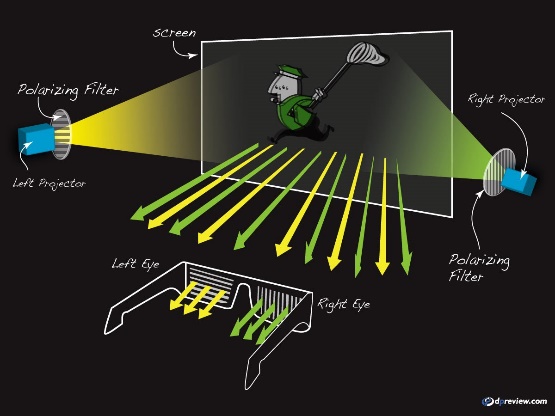
1. 1838년 영국의 물리학자 찰스 휘트스톤(Charles Wheatstone)은 현재 3D 기술의 시작점이 되는 논문을 발표
2. 왼쪽과 오른쪽의 눈이 평균 6.5cm가 떨어져 있어, 각 눈에 보이는 이미지의 차이로 입체감이 느껴진다는 양안시차(Stereoscopic Vision) 원리를 도구를 만들어 실험
3. Stereoscope



1. 양안시차의 원리는 당시 사진술(Photography)의 발달과 맞물려 재미있는 물건을 탄생시키는데 그것이 바로 스테레오스코프(Stereoscope, 입체경)
2. 당시 사진 기술로 응용 기술들이 등장하면서 사진 촬영이 좀 더 쉬워졌고 이를 이용한 콘텐츠들이 등장
3. Anaglyph

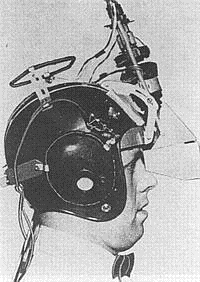
1. 이후 1850년대에는 애너글리프(Anaglyph) 방식이 등장
2. 애너글리프 방식이란 적색과 청색의 필터를 가진 안경을 쓰고, 이에 맞춰 구성된 화면을 보면 입체감을 느낄 수 있도록 만든 기술
3. 적, 청 색깔이 있는 별도의 안경을 착용하면 안경이 해당 색깔을 걸러내고 좌우 영상을 분리
4. 각각의 눈이 다른 이미지를 보게 되면서 입체감이 생기는 원리
5. Polarized Method



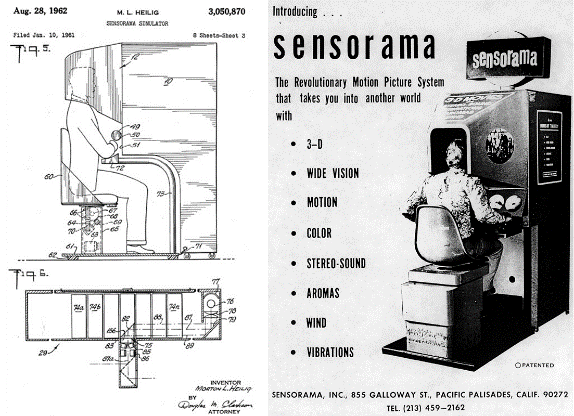
1. 1950년대에 큰 인기를 끌었던 3D영화들은 컬러 필터 방식의 애너글리프 방식이 아니라, 회색의 폴라로이드 필터를 사용한 편광 방식이 주로 사용
2. 편광 방식은 최근의 3D영화에도 꾸준히 사용되고 있음
3. Reel

1. 릴(Reel)은 가운데 원을 기준으로 서로 매칭 되는 이미지들이 각각 왼쪽, 오른쪽 눈에 비치면서 최종적으로 사용자에게 입체 이미지를 보여줌
2. 본격적으로 등장하는 가상현실 기기들 (1960~1990)
3. Telepresence

1. 필코(Philco)에서 1961년도에 제작한 텔레프레전스(Telepresence) 시스템
2. 머리의 움직임에 맞춰 외부의 비디오카메라가 움직이면서 영상을 볼 수 있는 방식
3. 머리에 쓸 수 있었지만 무거웠기 때문에 별도 고정 장치를 사용
4. Sensorama



1. 1962년 개발한 센소라마(Sensorama)라는 입체 영상 기기
2. 3D Stereoscopic 기반의 영상 외에도 모션과 향기, 바람, 진동 등을 지원
3. 스테레오 사운드 기능까지 제공
4. Experimental 3-D display



1. 1961년 제작된 필코(Philco)의 Headsight 외에도 초창기 HMD(Head Mounted Display)라고 볼 수 있는 기기가 1968년 등장
2. Videoplace



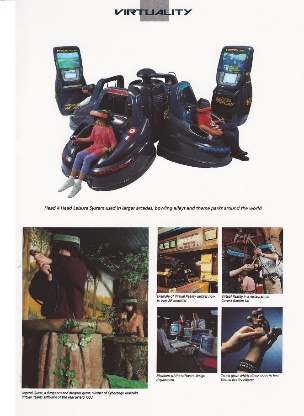
1. 1970년대에는 마이론 크루거가 ‘비디오플레이스(Videoplace)’ 라는 실험 공간을 제작
2. 비디오플레이스 안에서는 마치 공상과학 영화의 한 장면처럼 체험자의 손이 컴퓨터 스크린에 나타나고, 손을 이용하여 가상의 물체와 상호작용
3. 크루거는 이러한 환경을 ‘인공현실(Artificial Reality)’이라고 표현했는데, 오늘날의 가상현실과 그 의미가 매우 유사
4. LEEP

1. 1979년에는 에릭 휴렛(Eric Howlett)이 의도적으로 색수차(chromatic aberration)를 준 입체 사진을 찍고 이 필름을 전용 뷰어를 통해서 보면 렌즈가 다시 색수차를 보정하는 과정에서 넓은 시야각을 얻을 수 있는 LEEP 시스템을 만듦
2. 기존 사진들의 시야각이 평균 20도에서 50도였는데 이를 획기적으로 늘림
3. 관련 기술들은 이후 NASA의 초기 HMD 제작 및 이후 상용 제품들에게 큰 영향을 줌
4. VIEW



1. 1984년 NASA Ames 산하의 VIEW(Virtual Interface Environment Workstation) Lab에서 가상현실 역사의 시계를 더 빠르게 돌리는데 크게 공헌
2. HMD를 통해 고개의 움직임을 일치시켜 주변을 살펴보고, 조이스틱을 이용해 로봇을 원격으로 이동
3. 일반 소비자 시장에 도전하는 기기들 (1990~2000)
4. 아케이드 시장 출시

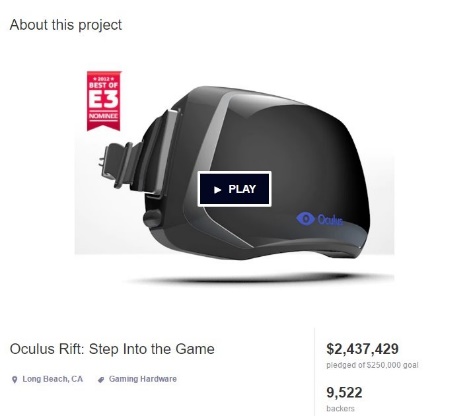


1. 1991년 버추얼리티 그룹(Virtuality Group)에서 제작한 여러 종류의 게임들이 아케이드에 설치
2. 한국에도 1992년 이 기기가 압구정과 에버랜드에서 운영
3. 당시 기기 수입 가격이 대당 7천만원
4. 가정용 게임기 기반 VR 기기

실내, 안경, 보안경이(가) 표시된 사진

자동 생성된 설명

1. 가정용 게임기 기반의 VR기기에 도전한 업체는 많았지만 기술적인 문제로 인해 시장에 발매되지 못하거나 조기 판매 종료
2. 새로운 업체들의 등장 (2000~)
3. Oculus Rift



1. 2012년 E3에서 큰 주목을 받은 오큘러스 리프트(Oculus Rift)는 펀딩을 시작
2. 단 하루 만에 $250,000 목표치를 달성, 최종적으로 $2,437,429로 마감하면서 엄청난 화제
3. 이후 2014년 페이스북이 20억달러로 오큘러스를 인수
4. VIVE

물품, 정렬된, 여러개이(가) 표시된 사진

자동 생성된 설명 실내, 작업대이(가) 표시된 사진

자동 생성된 설명

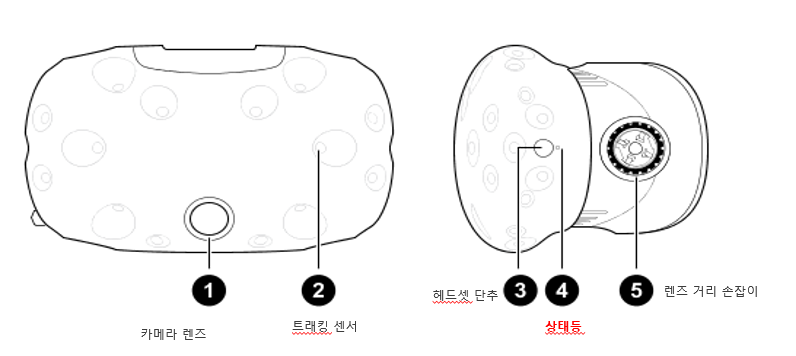
1. HTC와 밸브(Valve)가 2년여 동안 협업하여 Vive를 출시
2. 가상현실을 두 발로 걸어 다닐 수 있는 ‘룸스케일(Room Scale)’ 기능 추가
3. PS VR



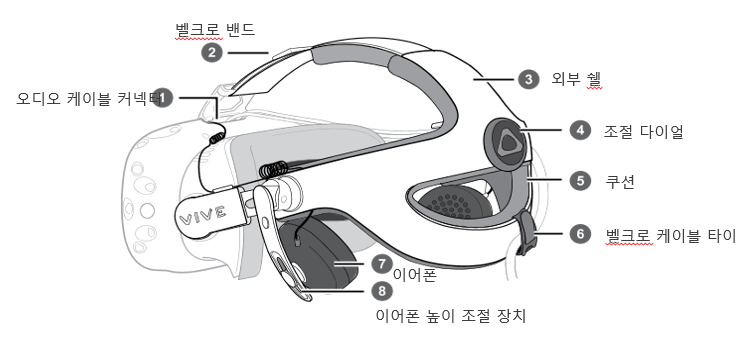
1. Sony의 가정용 게임 기기인 PlayStation4 와 결합한 PS VR 출시
2. 가격대를 낮춰 소비자들에게 호의적인 반응을 이끌어냄
3. 가상현실 관련 시장, 사례 소개
4. 영상 콘텐츠
5. 페이스북이나 유튜브에 접속하면 가상 현실 관련 콘텐츠를 직접 체험 가능
6. 스포츠 중계도 가상 현실을 이용해 시청 가능
7. 콘텐츠 산업
8. 가상현실 증강현실을 주제로 한 게임 및 콘텐츠 관련 박람회
9. 게임 산업
10. 가상 현실 콘텐츠를 가장 많이 만들어낼 수 있는 시장
11. 성인 산업
12. 2020년 성인 엔터테인먼트 시장이 영화와 게임을 이어서 3위 규모의 가상 현실 시장이 될 거라고 예상
13. 헬스케어 분야
14. 초기 가상 현실 산업을 발전시킨 영상과 게임 같은 콘텐츠를 적용하기에 가장 전망이 있는 산업
15. 기타 산업
16. 놀이기구, 가상현실 체험존
17. 혼합 현실(가상 현실 + 증강 현실)
18. 오딧세이/바이브 기본 구성 요소
19. VIVE



1. VIVE 헤드셋
2. 헤드셋은 VR 환경에서 창문 역할
3. 헤드셋에는 베이스 스테이션에서 추적하는 센서가 있음



1. VIVE 디럭스 오디오 스트랩
2. VIVE 디럭스 오디오 스트랩을 사용하면 통합된 이어폰을 통해 고품질 사운드로 VIVE VR 시스템을 더욱 실감 나게 즐길 수 있음

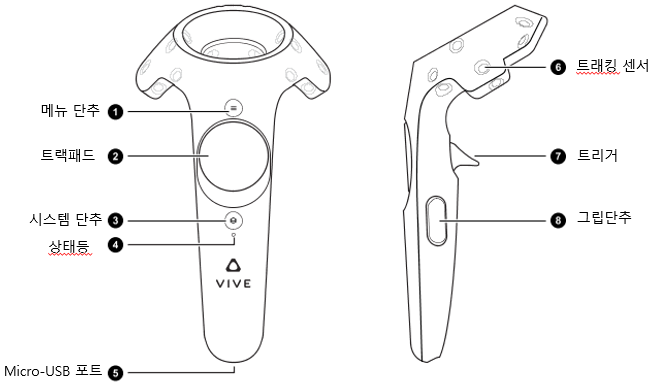


1. VIVE 베이스 스테이션
2. 베이스 스테이션은 VR 경험을 위한 360도 모션 추적 제공 장치

텍스트, 전자기기이(가) 표시된 사진

자동 생성된 설명

1. VIVE 컨트롤러
2. 컨트롤러를 사용하여 가상 현실(VR) 세계의 개체와 상호 작용
3. 베이스 스테이션에서 추적하는 센서가 있음



1. Odyssey



1. 오브젝트 제어 기능 버튼 이름 및 기능
2. 조작도구
3. 화면 제어

R1280x0 (297×28)

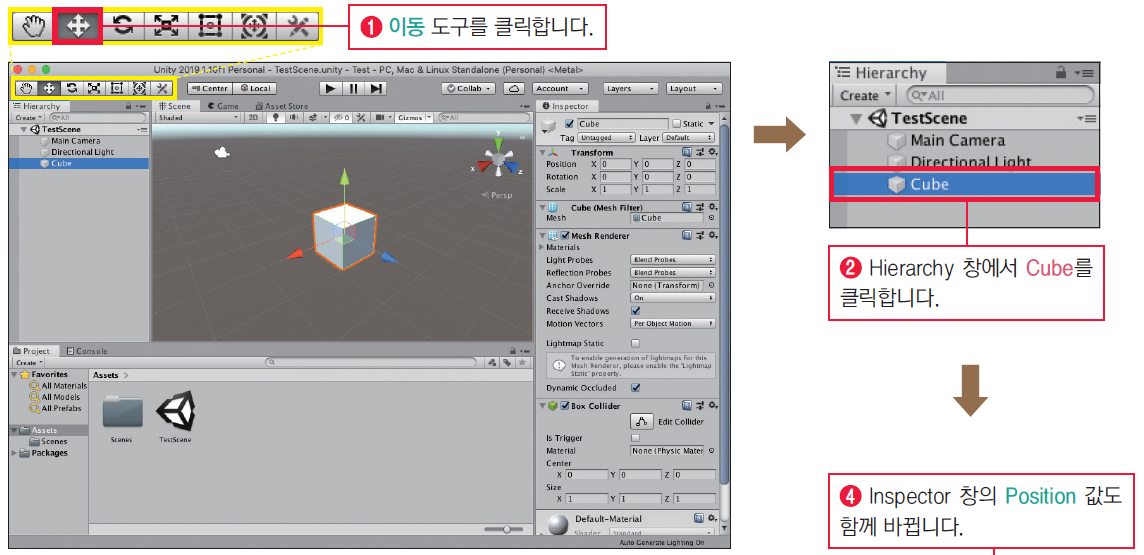
1. 핸드 툴(Hand Tool)
2. 씬 뷰 화면 전체를 이동하는 툴
3. 게임 오브젝트 제어



1. 무브 툴(Move Tool), 로데이트 툴(Rotate Tool), 스케일 툴(Scale Tool), 사각 트랜스폼 툴(RectTransform Tool), 트랜스폼 툴(Transform Tool)
2. 커스텀 제어



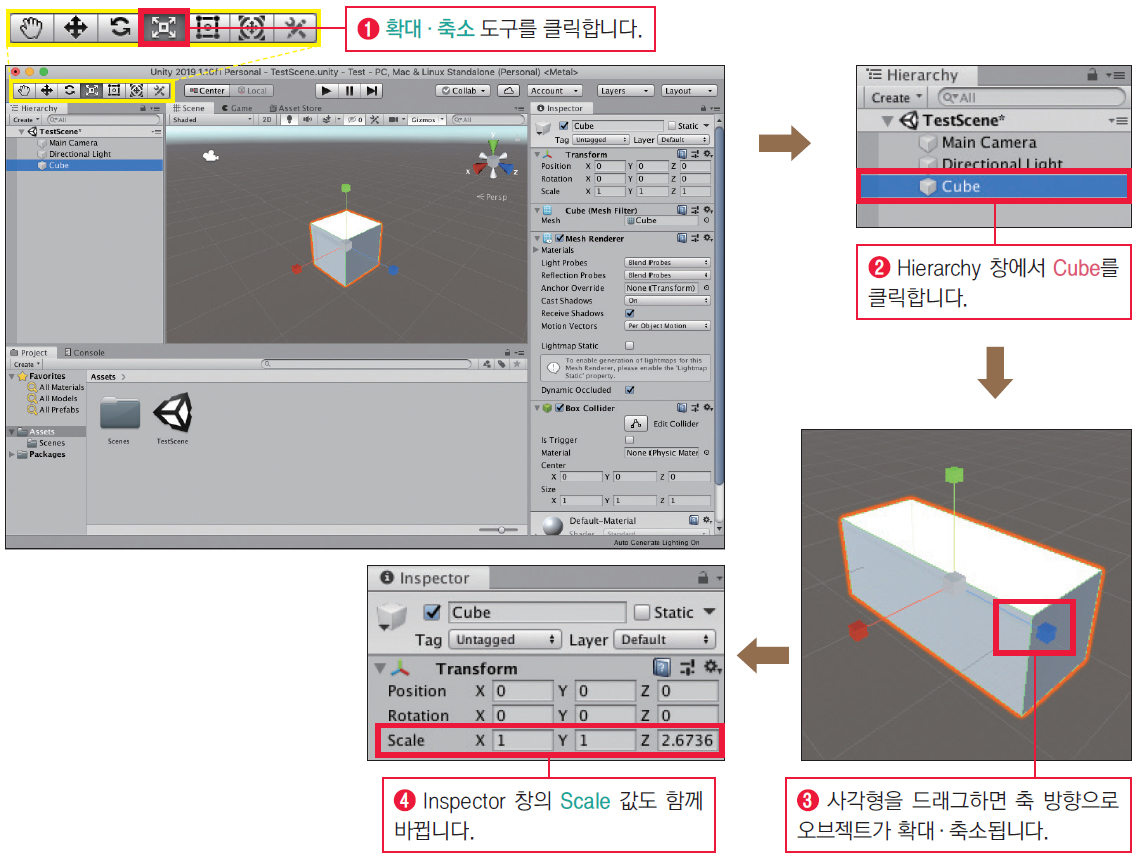
1. 커스텀 툴(Available Custom Editor Tools)
2. 마우스 오른쪽 버튼으로 클릭하여 커스텀 툴에 액세스할 수 있음
3. 게임 오브젝트 제어
4. 이동 도구(Move Tool)
5. 오브젝트를 옮기려면 이동 도구를 사용



1. 회전 도구(Rotate Tool)
2. 오브젝트를 회전시키려면 회전 도구를 사용

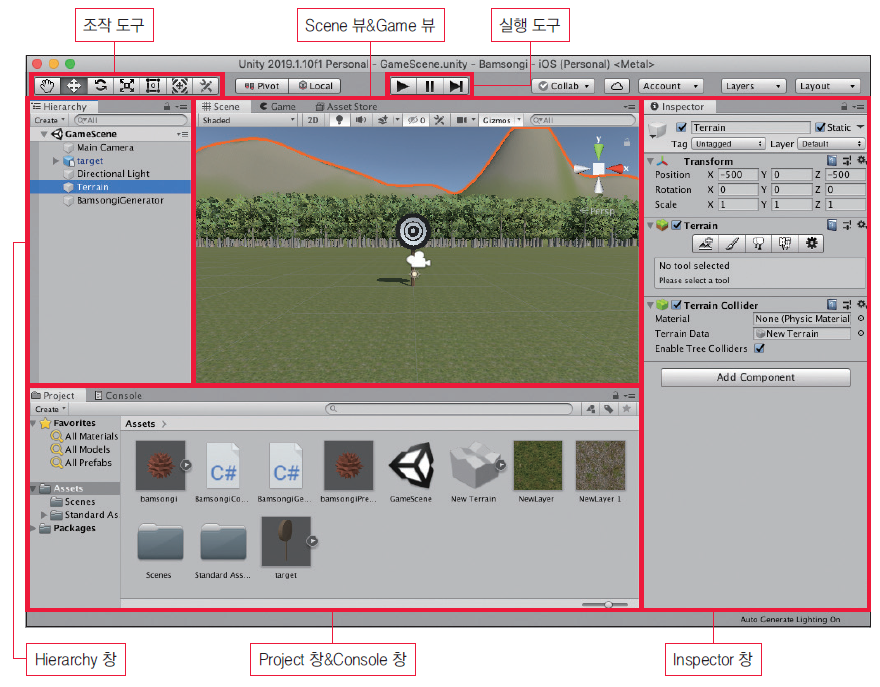


1. 확대·축소 도구(Scale Tool)
2. 오브젝트를 확대·축소하려면 확대·축소 도구를 사용



1. 사각 트랜스폼 툴(RectTransform Tool)
2. 트랜스폼 툴(Transform Tool)
3. 이동 도구 + 회전 도구 + 확대·축소 도구

1. 기타 기능
2. 레이아웃 변경
3. 게임 실행 화면의 크기 변경
4. 프로파일러
5. 유니티 2D와 3D
6. 5가지 화면 이름 및 기능
7. 유니티 화면 구성



1. Scene 뷰(씬 뷰)
2. 게임을 구성하는 메인 화면
3. 주로 리소스를 배치해 게임 씬을 작성
4. 뷰 위쪽의 탭을 클릭해 Game 뷰 등을 볼 수 있음
5. Game 뷰(게임 뷰)
6. 게임을 실행했을 때 보이는 방향을 확인할 수 있음
7. 게임의 처리 속도나 과부하가 걸리는 곳 등을 분석할 수 있음
8. Hierarchy 창(하이어라키 창, 계층 구조 창)
9. Scene 뷰에 배치한 오브젝트 이름을 목록으로 표시
10. 오브젝트 사이의 계층 구조를 표시하거나 편집할 수도 있음
11. Project 창(프로젝트 창)
12. 게임에서 사용하는 리소스를 관리
13. 이미지나 음원 등 리소스를 드래그 & 드롭하면 게임 리소스로 추가됨
14. Console 창
15. 프로그램에 오류가 있을 때 그 내용을 표시
16. 프로그램에 따라 임의의 수치나 문자열을 표시할 수도 있음
17. Inspector 창
18. Scene 뷰에서 선택한 오브젝트의 상세 정보가 나타남
19. Inspector 창에서 오브젝트의 좌표, 회전, 크기(스케일), 색, 모양 등을 설정
20. 조작 도구
21. Scene 뷰에 배치한 오브젝트의 좌표, 회전, 크기를 조절할 수 있음
22. Scene 뷰의 보이는 방향을 조정할 수 있음
23. 실행 도구
24. 게임을 실행하거나 정지시킴
25. 스크립트 3가지 종류
26. 컨트롤러 스크립트
27. 제너레이터 스크립트
28. 감독 스크립트
29. c# 클래스 멤버 접근 수식자
30. public
31. 모든 클래스에서 접근 가능
32. protected
33. 같은 클래스와 해당 클래스의 서브클래스에서 접근 가능
34. private
35. 같은 클래스에서만 접근 가능
36. 접근 수식자를 생략하면 private으로 간주되므로 공개하고 싶은 변수와 메서드가 있다면 public 수식자를 붙이는 것이 좋음
37. 게임 기획/설계하는 5단계 과정(서술형)
38. 화면에 놓일 오브젝트를 모두 나열
39. 오브젝트를 움직일 수 있는 컨트롤러 스크립트를 정함
40. 오브젝트를 자동으로 생성할 수 있도록 제너레이터 스크립트를 정함
41. UI를 갱신할 수 있도록 감독 스크립트를 준비
42. 스크립트를 만드는 흐름을 생각
43. 오브젝트, UI, 프리팹 적용하는 과정(서술형)
44. 오브젝트
45. Scene 뷰에 오브젝트를 배치
46. 오브젝트를 움직이는 방법을 쓴 스크립트를 작성
47. 작성한 스크립트를 오브젝트에 적용
48. UI
49. UI 부품을 Scene 뷰에 배치
50. UI를 갱신하는 감독 스크립트를 작성
51. 빈 오브젝트를 만들고 작성한 스크립트를 적용
52. 프리팹
53. 이미 있는 오브젝트를 사용해 프리팹을 만듬
54. 제너레이터 스크립트를 만듬
55. 빈 오브젝트에 제너레이터 스크립트를 적용
56. 제너레이터 스크립트에 프리팹을 전달
57. 아웃렛 접속
58. 스크립트 쪽에 콘센트 구멍을 준비하기 위해 스크립트 변수 앞에 public 지정자를 붙임
59. public 지정자를 붙인 변수가 Inspector에 보임
60. Inspector의 콘센트 구멍에 대입할 오브젝트를 끼움(드래그&드롭)
61. 자신 이외의 오브젝트를 가진 컴포넌트로 접근
62. Find 메서드로 오브젝트를 찾음
63. GetComponent 메서드로 오브젝트의 컴포넌트를 구함
64. 컴포넌트가 가진 데이터로 접근
65. GameObject director = GameObject.Find(“GameDirector”);
66. director.GetComponent<GameDirector>().DecreaseHp();
67. 효과음
68. 효과음을 낼 오브젝트에 AudioSource 컴포넌트를 적용
69. AudioSource 컴포넌트에 효과음을 설정
70. 효과음을 울리고 싶은 시간에 Play 메서드를 호출
71. GetComponent<AudioSource>().Play();
72. 5주차 실습소스
73. 컨트롤러 스크립트
74. RouletteController

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

public class RouletteController : MonoBehaviour

{

    float rotSpeed = 0; *// 회전 속도*

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

    void Start()

    {

    }

*// Update is called once per frame*

    void Update()

    {

*// 클릭하면 회전 속도를 설정한다.*

*if*(Input.GetMouseButtonDown(0)) {

            this.rotSpeed = 100;

*// 효과음을 재생한다.*

            GetComponent<AudioSource>().Play();

*// 파티클을 표시한다.*

            GetComponent<ParticleSystem>().Play();

        }

*// 회전 속도만큼 룰렛을 회전시킨다.*

        transform.Rotate(0, 0, this.rotSpeed);

*// 룰렛을 감소시킨다.*

        this.rotSpeed \*= 0.98f;

    }

}

1. 6주차 실습소스
2. 컨트롤러 스크립트
3. CarController

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

*/\* 스와이프 길이에 따라 자동차 이동 거리 바꾸기 \*/*

public class CarController : MonoBehaviour

{

    float speed = 0;

    Vector2 startPos;

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

    void Start()

    {

    }

*// Update is called once per frame*

    void Update()

    {

*// 스와이프 길이를 구한다.*

*if* (Input.GetMouseButtonDown(0)) {

*// 마우스를 클릭한 좌표*

            this.startPos = Input.mousePosition;

        }

*else* *if* (Input.GetMouseButtonUp(0)) {

*// 마우스 버튼에서 손가락을 떼었을 때 좌표*

            Vector2 endPos = Input.mousePosition;

            float swipeLength = (endPos.x - this.startPos.x);

*// 스와이프 길이를 처음 속도로 변경한다.*

            this.speed = swipeLength / 500.0f;

*// 효과음을 재생한다.*

            GetComponent<AudioSource>().Play();

        }

        transform.Translate(this.speed, 0, 0); *// 이동*

        this.speed \*= 0.98f; *// 감속*

    }

}

1. 감독 스크립트
2. GameDirector

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI; *// UI를 사용하는 데 필요하다.*

public class GameDirector : MonoBehaviour

{

    GameObject car;

    GameObject flag;

    GameObject distance;

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

    void Start()

    {

        this.car = GameObject.Find("car");

        this.flag = GameObject.Find("flag");

        this.distance = GameObject.Find("Distance");

    }

*// Update is called once per frame*

    void Update()

    {

        float length = this.flag.transform.position.x - this.car.transform.position.x;

*if* (length > 20) {

            this.distance.GetComponent<Text>().text = "목표 지점으로 출발!";

        }

*else* *if* (length > 0 && length <= 20) {

            this.distance.GetComponent<Text>().text = "목표 지점까지 " + length.ToString("F2") + "m";

        }

*else* {

            this.distance.GetComponent<Text>().text = "목표 지점 도착!";

        }

    }

}

1. 7주차 실습소스
2. 컨트롤러 스크립트
3. PlayerController

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

*/\* 키를 조작해 플레이어 움직이기*

*버튼을 누르면 움직이도록 추가하기 \*/*

public class PlayerController : MonoBehaviour

{

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

    void Start()

    {

    }

    public void UButtonDown()

    {

        transform.Translate(0, 3, 0); *// 위쪽으로 [3] 움직인다.*

    }

    public void DButtonDown()

    {

        transform.Translate(0, -3, 0); *// 아래쪽으로 [3] 움직인다.*

    }

    public void LButtonDown()

    {

        transform.Translate(-3, 0, 0); *// 왼쪽으로 [3] 움직인다.*

    }

    public void RButtonDown()

    {

        transform.Translate(3, 0, 0); *// 오른쪽으로 [3] 움직인다.*

    }

*// Update is called once per frame*

    void Update()

    {

*// 위쪽 화살표가 눌렸을 때*

*if* (Input.GetKeyDown(KeyCode.UpArrow)) {

            transform.Translate(0, 3, 0); *// 위쪽으로 [3] 움직인다.*

        }

*// 아래쪽 화살표가 눌렸을 때*

*if* (Input.GetKeyDown(KeyCode.DownArrow)) {

            transform.Translate(0, -3, 0); *// 아래쪽으로 [3] 움직인다.*

        }

*// 왼쪽 화살표가 눌렸을 때*

*if* (Input.GetKeyDown(KeyCode.LeftArrow)) {

            transform.Translate(-3, 0, 0); *// 왼쪽으로 [3] 움직인다.*

        }

*// 오른쪽 화살표가 눌렸을 때*

*if* (Input.GetKeyDown(KeyCode.RightArrow)) {

            transform.Translate(3, 0, 0); *// 오른쪽으로 [3] 움직인다.*

        }

    }

}

1. ArrowController

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

*/\* 화살 떨어트리기*

*충돌 판정 추가히기*

*DecreaseHP 메서드를 호출하는 처리 추가하기 \*/*

public class ArrowController : MonoBehaviour

{

    GameObject player;

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

    void Start()

    {

        this.player = GameObject.Find("player");

    }

*// Update is called once per frame*

    void Update()

    {

*// 프레임마다 등속으로 낙하시킨다.*

        transform.Translate(0, -0.1f, 0);

*// 화면 밖으로 나오면 오브젝트를 소멸시킨다.*

*if* (transform.position.y < -5.0f) {

            Destroy(gameObject);

        }

*// 충돌 판정 (추가)*

        Vector2 p1 = transform.position; *// 화살 중심 좌표*

        Vector2 p2 = this.player.transform.position; *// 플레이어 중심 좌표*

        Vector2 dir = p1 - p2;

        float d = dir.magnitude;

        float r1 = 0.5f; *// 화살 반경*

        float r2 = 1.0f; *// 플레이어 반경*

*if* (d < r1 + r2) {

*// 감독 스크립트에 플레이어와 화살이 충돌했다고 전달한다.*

            GameObject director = GameObject.Find("GameDirector");

            director.GetComponent<GameDirector>().DecreaseHP();

*// 효과음을 재생한다.*

            GetComponent<AudioSource>().Play();

*// 충돌한 경우는 화살을 지운다.*

            Destroy(gameObject);

        }

    }

}

1. 제너레이터 스크립트
2. ArrowGenerator

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

*/\* 화살을 생성하는 제너레이터 스크립트 \*/*

public class ArrowGenerator : MonoBehaviour

{

    public GameObject arrowPrefab;

    float span = 1.0f;

    float delta = 0;

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

    void Start()

    {

    }

*// Update is called once per frame*

    void Update()

    {

*// Update 메서드는 프레임마다 실행*

*// 앞 프레임과 현재 프레임 사이의 시간 차이는 Time.deltaTime에 대입*

        this.delta += Time.deltaTime;

*if* (this.delta > this.span) {

            this.delta = 0;

*// Instantiate 메서드는 매개변수로 프리팹을 전달하면 반환값으로 프리팹 인스턴스를 돌려줌*

*// Instantiate 메서드는 가장 기본적인 ‘Object형’을 반환*

*// 캐스트라는 강제 형 변환을 사용해 Object형을 GameObject형으로 바꿀 수 있음*

            GameObject go = Instantiate(arrowPrefab) as GameObject;

*// 화살의 X 좌표는 -6부터 6 사이에 불규칙하게 위치하도록 Random 클래스의 Range 메서드 사용*

*// Range 메서드는 첫 번째 매개변수보다 크고 두 번째 매개변수보다 작은 범위에서 무작위 수를 정수로 반환*

            int px = Random.Range(-6, 7);

            go.transform.position = new Vector3(px, 7, 0);

        }

    }

}

1. 감독 스크립트
2. GameDirector

using System.Collections;

using System.Collections.Generic;

using UnityEngine;

using UnityEngine.UI; *// UI를 사용하므로 잊지 말고 추가한다.*

*/\* UI를 감독하는 스크립트 \*/*

public class GameDirector : MonoBehaviour

{

    GameObject hpGauge;

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

    void Start()

    {

        this.hpGauge = GameObject.Find("hpGauge");

    }

    public void DecreaseHP()

    {

        this.hpGauge.GetComponent<Image>().fillAmount -= 0.1f;

    }

*// Update is called once per frame*

    void Update()

    {

    }

}