끊임없는 **배움**() // 으로

예술과 기술을 **조화**() // 롭게 엮어

극한의 **완성도**() // 를 추구하는 게임 개발자

return **"박지원입니다."**;

public **Phone** 010-6694-9087

public **E-mail** jiwonia21@gmail.com

public **Github** https://github.com/25maru

목차(int index) {

01 소개

- Info // 학력, 전공, 경험
- Tech Stack

02 프로젝트

- 프로젝트 RC // 개인
- 프로젝트 T // 개인
- 플라네타리움 // 팀

}

소개.Info

〈 학력 〉

- 경문고등학교 졸업
- 계원예술대학교 게임미디어과 재학

〈 전공 〉

- 게임 프로그래밍 (주 전공)
- 2D 그래픽

〈 경험 〉

- 고등학교 홍보, 축제, 대회 등에 활용될 다양한 영상 제작
- 대한민국디자인전람회 (청소년) 제품디자인 부문 입상
- 대학교 졸업작품 개발 (프로그래밍, UI 그래픽)
- 3인칭 슈팅 게임 프로젝트 진행중 (1인 개발)

소개.Tech Stack

〈 게임 엔진 〉

- Unity

〈 프로그래밍 언어 〉

- C# / C++

〈 개발 환경 〉

- Visual Studio Code
- JetBrains Rider

〈 버전 관리 〉

- Git / GitHub / Fork

〈 셰이더 〉

- Shader Graph / Visual Effect Graph

〈 디자인 〉

- Blender / 3ds Max
- Photoshop / Illustrator

프로젝트

〈 플라네타리움 〉

- 계원예술대학교 졸업작품
- 장르: 3D 건반형 리듬게임
- 기간: 2024. 3. ~ 2024. 9.
- 인원: 2

팀

〈 프로젝트 RC 〉

- 장르: 레이싱 게임
- 기간: 2023. 5. ~ 2023. 7.

〈 프로젝트 T 〉

- 장르: 3인칭 슈팅 게임
- 기간: 2024. 9. ~ (진행중)

개인

프로젝트 RC ...


Driving Type
1: FWD
2: RWD
3: 4WD
Steering Type
4: FWS
5: RWS


Driving Type
1: FWD
2: RWD
3: 4WD
Steering Type
4: FWS
5: RWS

프로젝트 RC ...

PlayerController.cs (인풋 관리, 등록된 바퀴 관리, 효과음 재생)

- `SetDriveType()` : 키보드 입력을 받고, 구동 & 조향 방식 변경.
- `ApplySteering()` : 미끄러지는 정도를 계산하고, 정도에 따라 스키드마크 표시.
- `Reset()` : 자동차가 주행이 불가능 상태가 되면 초기 위치로 재설정.

```
public class PlayerController : MonoBehaviour
{
    public WheelController[] wheels;
    public WheelCollider[] wcols;

    // 무게중심
    [Header("무게중심")]
    public float centerOfMassY;
    public float centerOfMassZ;

    // 주행
    [Header("주행")]
    public bool canRun;
    public float speed;
    public float forceMultiplier = 1f;
    public float nitrous = 500f;
    bool shiftPressed = false;
    float timer = 0f;
    float duration = 1.5f;
    Rigidbody rb;
    public TrailRenderer[] skidMark;

    // 조향
    [Header("조향")]
    public float minSlipThreshold = 0.5f;
    public float carVelocity;
    public AnimationCurve steerCurve;

    // 사운드
    [Header("사운드")]
    // public AudioClip[] track;
    public AudioSource startSound;
    public AudioSource idleSound;
    public AudioSource runningSound;
    public AudioSource reverseSound;
    public AudioSource nitrousSound;

    // 초기화
    float lastResetTime = Mathf.NegativeInfinity;
    float resetDelay = 3.0f;
    bool isReset = false;
```

```
void SetDriveType()
{
    // 구동 시스템

    // 전륜구동
    if (Input.GetKeyUp(KeyCode.Alpha1) && canRun)
    {
        wheels[0].canDrive = true;
        wheels[1].canDrive = true;
        wheels[2].canDrive = false;
        wheels[3].canDrive = false;
    }
    // 후륜구동
    if (Input.GetKeyUp(KeyCode.Alpha2) && canRun)
    {
        wheels[0].canDrive = false;
        wheels[1].canDrive = false;
        wheels[2].canDrive = true;
        wheels[3].canDrive = true;
    }
    // 4륜구동
    if (Input.GetKeyUp(KeyCode.Alpha3) && canRun)
    {
        wheels[0].canDrive = true;
        wheels[1].canDrive = true;
        wheels[2].canDrive = true;
        wheels[3].canDrive = true;
    }

    // 조향 시스템

    // 전륜조향
    if (Input.GetKeyUp(KeyCode.Alpha4))
    {
        wheels[2].canSteer = false;
        wheels[3].canSteer = false;
    }
    // 4륜조향
    if (Input.GetKeyUp(KeyCode.Alpha5))
    {
        wheels[2].canSteer = true;
        wheels[3].canSteer = true;
    }
}
```

프로젝트 RC ...

WheelController.cs (휠 콜라이더의 토크 & 브레이크 설정)

- `Update()` : 미끄러질 때, 바퀴에서 드리프트 소리 재생.
- `FixedUpdate()` : 방향 입력에 맞게 토크와 브레이크 값을 설정하고, 바퀴 위치 미세 조정.

```
public class WheelController : MonoBehaviour
{
    WheelCollider wcol;
    Transform wmesh;
    public PlayerController player;

    // 토크
    [Header("토크")]
    public float torque;
    public float innerTorque = 500f;
    public float basicTorque = 1000f;
    public float outerTorque = 1500f;

    // 브레이크
    [Header("브레이크")]
    public float brake;
    public float maxBrake = 1000f;

    // RPM
    [Header("RPM")]
    public float maxRpm = 9250f;

    // 조향
    [Header("조향")]
    public float innerAngle = 39f;
    public float outerAngle = 32.5f;
    public float minSlipThreshold = 0.5f;
    public bool isDrifting = false;
    [HideInInspector]
    public float slipSide;
    // bool isSteering;

    [Header("속성")]
    public bool canDrive; // 구동륜
    public bool canSteer; // 조향륜
    public bool isFrontWheel;
    public bool isLeftWheel;

    // 사운드
    [Header("사운드")]
    public AudioSource driftSound;

    float vInput;
    float hInput;
```

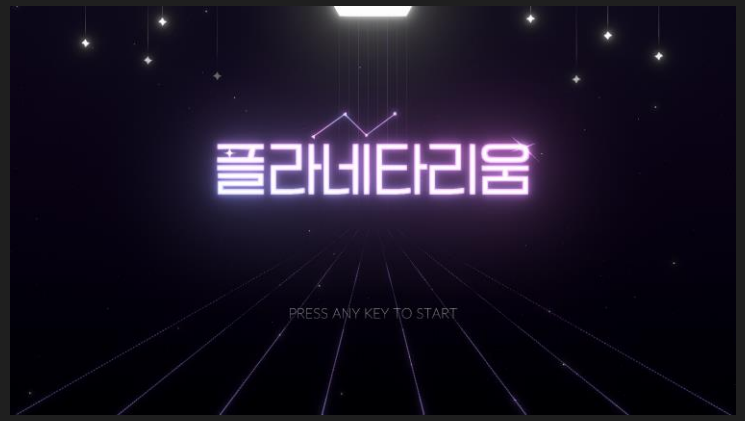
```
void Update()
{
    WheelHit wheelHit;
    wcol.GetGroundHit(out wheelHit);
    float slip = Mathf.Abs(wheelHit.sidewaysSlip);
    slipSide = wheelHit.sidewaysSlip;

    if (slip > minSlipThreshold)
    {
        isDrifting = true;

        if (!driftSound.isPlaying)
        {
            driftSound.Play();
        }
    }
    else
    {
        isDrifting = false;

        if (driftSound.isPlaying)
        {
            driftSound.Stop();
        }
    }
}

public void Accelerate(float torqueValue)
{
    wcol.motorTorque = torqueValue;
}

public void Brake(float brakeValue)
{
    wcol.brakeTorque = brakeValue;
}

public void Steering(float angle)
{
    wcol.steerAngle = angle * hInput;
}
```

프로젝트 T ...

프로젝트 T ...

프로젝트 T ...

플라네타리움 ...


플라네타리움
PRESS ANY KEY TO START


챕터
물고기
물병
양
염소
뱀주인
챕터 변경
나가기
챕터 선택


음악
XI - 물병자리
곡 선택
모드 변경
나가기
플레이


Score 1000000
100.00%
100
HARMONY!
+255
Speed 10.5


결과
- 멜로디 클리어 -
점수: 1000000
최대 콤보: 300
S
HARMONY! 300
GREAT 0
MISS 0
기록
나가기
다시하기


환경설정
게임 플레이
노트 속도
10.5
오프셋 설정
+120ms
판정 정확도 표시
미사용
달성도 마커 표시
사용
HARMONY 판정 이펙트 비활성화
미사용
키 가이드 표시
미사용
키 설정
설정
탭 선택
옵션 선택
옵션 변경
초기화
나가기
저장하기

플라네타리움 ...

GameManager.cs (“플레이 시작”, “일시정지”, “키 입력” 등의 핵심 기능 컨트롤)

- `PlaySong()` : 리듬게임 플레이에 맞춰 UI 불러오기, 점수 초기화, 게임 디렉터 스크립트 호출 등의 역할 수행.

GameDirector.cs (“음악 설정”, “노트 생성”, “속도 조절” 등의 리듬게임 플레이 관련 기능 컨트롤)

- `SetupTrackBindings()` : 타임라인에 노트와 그리드, 음악을 바인딩.
- `SetBpm()` : 음악의 BPM을 불러와서 타임라인에 적용.

GameProcessor.cs (노트 처리, 풀 관리 등의 기능을 이벤트를 통해 컨트롤)

- `TriggerInput()` : 노트 트리거 이벤트를 받아서 스코어 매니저, 이펙트 매니저에 전달.

플라네타리움 ...

ScoreManager.cs (점수 업데이트, 데이터 저장)

EffectManager.cs (이펙트 관리)

UIManager.cs (UI 패널, 팝업 관리)