프로젝트 최종 보고서

-장애물 경주-

컴퓨터과학전공 1814965 김현주

0. 목차

A. 플레이어 움직임

B. 트랙 별 장애물 생성, 움직임과 소멸

C. 날씨 변화와 효과, 부가 생성물의 움직임과 소멸

D. 플레이어의 충돌

E. 그 외의 기능

F. 게임 시작/재 시작 UI

1. 프로젝트 주제: <장애물 경주> 게임 제작

1) 게임 이름: 장애물 경주. Steeple Race

2) 개발 인원: 개인

3) 장르: 시뮬레이션 게임

4) 소개: 정해진 시간 내에 장애물들을 피해 완주하자!

5) 개발 환경: Unity, Visual Studio Code

6) 게임 플랫폼: PC Windows

7) 게임 방법: 트랙은 장애물의 종류에 따라 4구역으로 나눠진다. 캐릭터의 달리기(↑ 키), 점프(space-bar), 좌우 이동(←/→ 키) 모션으로 모든 구역을 통과하여 제한 시간 내에 결승점에 도달하면 된다.

2. 프로젝트 진행 단계

A. 프로젝트 계획서 내용을 토대로 구현할 기능 정의 (중간-완료)

B. 각 기능을 수행할 object와 코드 디자인 (중간-완료)

C. Object 간의 관계와 각 object에 적용될 property, script 매칭 (중간-완료)

D. 각 기능 개발 + 기능별 관계에서의 에러 수정 (완료)

E. 게임 시작/재 시작 버튼 제작 (완료)

3. 프로젝트의 요구 기능 (계획서 요약)

A. Camera

i. 3인칭, 플레이어 추적

B. Player

i. Forward move: 키 누르는 동안 앞으로 달린다. 키를 떼면 바로 멈춘다.

ii. Left/right move: 캐릭터는 일정 거리 이상으로 옆으로 이동할 수 없다.

iii. Jump: Space 키 누르면 점프

iv. Rotation freeze

C. Obstacle

i. Hurdle: track1에서 일정한 간격으로 생성한다.

ii. Stone: track2에서 임의의 위치에서 생성되고 중력에 의해 떨어진다. 바닥에 닿으면 사라진다. 돌멩이의 그림자가 잘 보이도록 해서 어디로 떨어질지 예측할 수 있어야 한다.

iii. Board: track3에서 트랙을 대신한다. 판의 양 옆은 낭떠러지이다.

iv. Stick: track4에서 x축을 기준으로 좌우로 움직인다.

D. Weather

i. 일정 시간마다 랜덤으로 'sun', 'cloud', 'rain', 'snow' 날씨를 구현한다.

ii. Sun: 지열로 인한 아지랑이를 구현한다. Player가 15초 이상 연속으로 달리면 탈진으로 3초간 움직이지 못한다.

iii. Cloud: 천천히 이동하는 구름을 생성한다.

iv. Rain: 구름을 많이 생성한다. 비가 내리고, 바닥에 닿으면 사라진다. Player의 이동속도를 낮춘다. 시야가 감소한다.

v. Snow: 구름을 많이 생성한다. 눈이 내리고, 바닥에 닿으면 사라진다. 눈덩이와 빙판이 랜덤으로 생긴다. 시야가 감소한다.

E. Portal + Start/End point

i. Portal-A: track 1, track 2, track 3의 도착 부분에 위치한다.

ii. Portal-B: track 2, track 3, track 4의 시작 부분에 위치한다.

iii. Portal-C: track 3와 track4의 아래에 위치하여 player가 board, track에서 떨어지면 이 포탈을 통해 track의 처음 시작 위치(=portal-B)로 이동한다.

iv. Start point: 게임의 처음 시작 위치 이자 track 1의 시작 부분

v. End point: 게임의 목적지이자 track 4의 도착 부분

F. Collider

i. Player & hurdle: 부딪히는 소리가 나며 player는 n초간 움직이지 못한다. 각 hurdle은 1번 충돌하면 사라진다.

ii. Player & stone: 부딪히는 소리가 나며 돌멩이는 사라지고, player는 n초간 움직이지 못한다.

iii. Track & stone: 돌멩이는 바닥에서 구르고 2초 후에 사라진다.

iv. Player & board: player의 이동 속도와 점프 높이를 낮춘다. 강한 바람소리가 나며 player의 x 좌표가 미세하게 변한다.

v. Player & stick: player가 튕겨져 나가며 소리가 나고 n초간 움직이지 못한다. Stick은 그대로 움직인다.

vi. Player & ice: player의 이동 속도와 점프 높이를 낮춘다.

vii. Player & snow_set: player의 이동 속도를 낮추고 player의 x 좌표가 미세하게 변한다.

viii. Player & portal: player가 track 끝 부분에 위치한 portal-A에 도착하면 다음 track의 시작 부분에 위치한 portal-B로 이동하며 시각적 이펙트를 준다.

G. Etc.

i. Timer: 게임 종료까지 남은 시간을 화면 상단에 표시한다. 남은 시간이 20초 이하이면 시간을 강조한다.

ii. Distance: player 위치에서 결승선까지 남은 거리를 화면 상단에 표시한다.

iii. Weather: 현재 적용되고 있는 날씨와 그 효과를 화면 상단에 표시한다.

iv. Speed: 현재 player의 속도를 화면 상단에 표시한다.

v. notMove: player가 장애물에 의해 움직이지 못할 때 그 사실을 표시한다.

vi. Game start: 게임을 시작할 때 5초 카운트 다운을 출력한다. 그 동안 player는 움직일 수 없다.

vii. Game end: 제한 시간 내에 end point에 도착하면 성공 메시지를, 그렇지 못하면 실

패 메시지를 출력하고, 그 이후에 player는 움직일 수 없다..

4. GameObject 구조와 Assets 정보 & 중간보고서 대비 추가 구현 현황

:: 회색 처리 : 중간 보고서 처리 내용

:: 검은색 처리 : 중간 보고서 이후 처리 내용

A. Hierarchy

i. Directional Light

ii. (추가) InitObject: Before.cs.

iii. ScriptObject: EmptyObject. InitScript.cs, changeWeather, Start.cs

iv. Player: rigidbody, rotation-freeze, animation, PlayerMove.cs, PlayerCollision.cs

- Main Camera

- Spot Light: Grain, Gsnow일 때 활성화, 그 외에는 비활성화

v. (추가) Canvas

- Track: UI text

- Time: UI text

- Distance: UI text

- Speed: UI text

- Weather: UI text

- Start: UI text. 시작 카운트 다운

- Result: UI text

- notMove: UI text

- str

    - start: UI button. 게임 시작 버튼

    - title: UI text. 게임 이름(Steeple Race)

- end: UI button. 재 시작 버튼

vi. Audio

- Portal
- Snow_Set
- Ice
- Thunder
- Rain
- Wind
- Hurdle
- Stick
- BGM

vii. Field

- Portal-A
- Portal-B
- Track
    - track 1
        - track: prefab
        - hurdle(n): prefab, rigidbody, hurdle.cs
    - track 2: createStone.cs
        - track: prefab
        - stone: Resources
    - track 3
        - board
        - portal-C
    - track 4
        - track: prefab
        - stick(n): prefab, moveStick.cs
        - portal-C

- end Point
- GoalLine

- (추가) Grass: 날씨에 따라 잔디 구성물 바꿈
  - spring: Gsun/Gloud
  - raining: Grain
  - winter: Gsnow

viii. Gsun: Resources
- heat_haze: particle

ix. Gcloud: Resources
- cloud: Resources

x. Grain: Resources
- darkCloud: Resources
- rain: particle

xi. Gsnow: Resources
- darkCloud: Resources
- snow: particle
- snow_set: Resources
- ice: Resources

B. Resources

i. stone: prefab, rigidbody, deleteStone.cs, bounce

ii. portalEffect: Particle System

iii. weather/Gsun: EmptyObject, heat.cs

iv. weather/Gcloud: EmptyObject, createCloud.cs

v. weather/Grain: EmptyObject, createRain.cs

vi. weather/Gsnow: EmptyObject, createSnow.cs

vii. cloud: prefab, deleteCloud.cs, moveCloud.cs

viii. darkCloud: prefab, deleteRain.cs, moveCloud.cs

ix. snow_set: prefab, deleteSnow.cs

x. ice: prefab, deleteSnow.cs

C. Physical Material

i. (추가) Bounce: 돌멩이가 바닥에서 잘 구를 수 있도록 설정

D. Scripts

i. PlayerMove.cs: Player 이동

ii. InitScript.cs: 여러 UI text 출력, 게임 종료

iii. PlayerCollision.cs: 충돌 후 player 상태 변화(speed, notMoveTime)

iv. createStone.cs: stone 자동 생성

v. deleteStone.cs: stone 소멸

vi. moveStick.cs: stick의 움직임

vii. changeWeather.cs: 날씨 변화, 날씨 영향

viii. createCloud.cs: 날씨가 Cloud일 경우, 관련 asset 자동 생성

ix. deleteCloud.cs: 날씨가 Cloud가 아닐 경우 관련 asset 자동 소멸

x. createRain.cs: 날씨가 Rain일 경우, 관련 asset 자동 생성

xi. deleteRain.cs: 날씨가 Rain이 아닐 경우 관련 asset 자동 소멸

xii. createSnow.cs: 날씨가 Snow일 경우, 관련 asset 자동 생성

xiii. deleteSnow.cs: 날씨가 Snow가 아닐 경우 관련 asset 자동 소멸

xiv. moveCloud.cs: 구름, 먹구름 이동

xv. Start.cs: 게임 시작 카운트다운

xvi. (추가) Heat.cs: track3에서 heat_haze 생성x

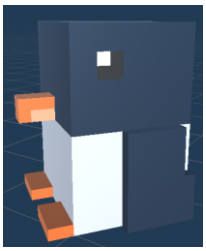
xvii. (추가) Hurdle.cs: hurdle이 player와 1번 충돌하면 destroy

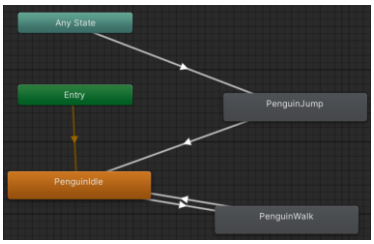
xviii. (추가) Before.cs. 게임 재 시작을 위한 모든 설정 초기화

5. 외부 assets과 자체 구현 assets

A. 외부 assets

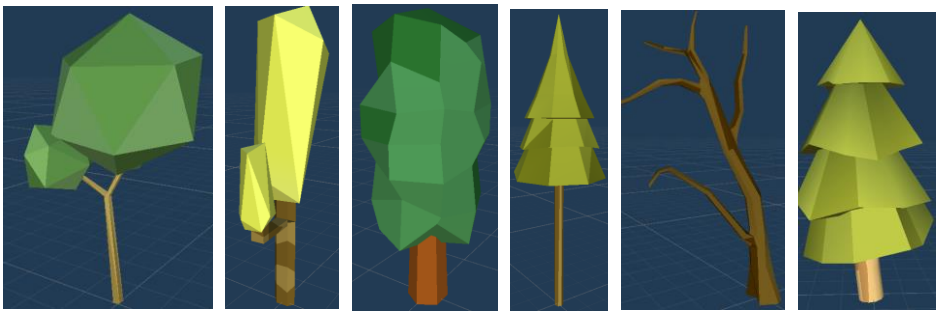
i. Player 모델, 애니메이션

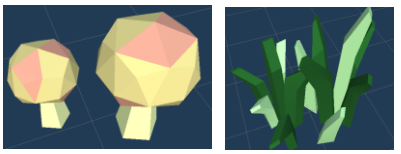


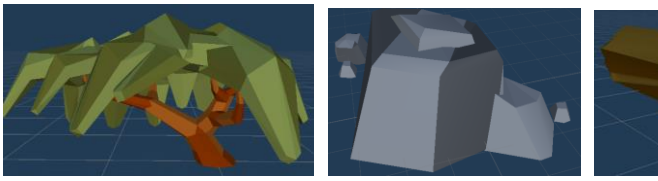


ii. Audio

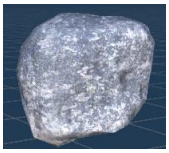
iii. Grass의 나무, 꽃, 바위 등





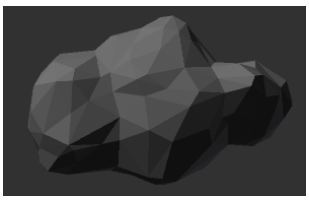


iv. Stone 모델



v. Board 텍스처: wood texture

vi. Cloud 모델



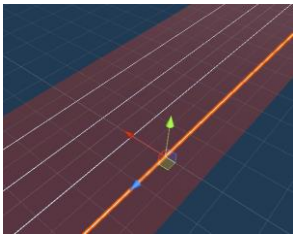
vii. Ice, snow_set 모델: cloud 모델 변형

viii. Snow 텍스처

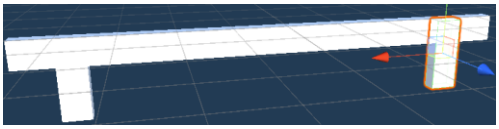
B. 자체 구현 assets

i. Portal-A, B, C, end point: plane 검은색

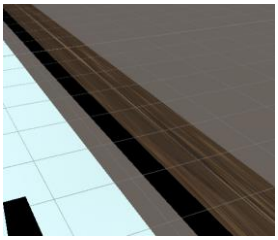
ii. Track: plane 빨간색 + line (plane 흰색 * 4)



iii. Hurdle: cube * 3



iv. Board: cube

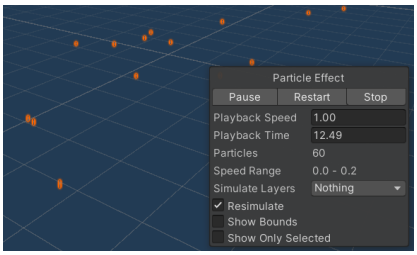


v. Stick: cylinder

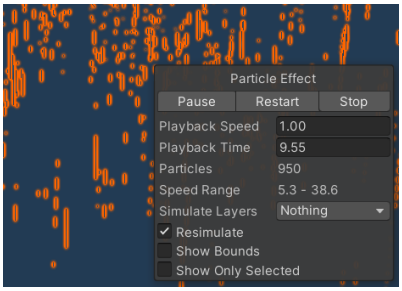
vi. Goal line: cylinder * 2 + plane + 3d text

vii. Grass: plane 초록색 + 나무, 식물, 바위 등 모델 조합

viii. Heat_haze: particle system



ix. Rain, Snow: particle system



x. Bounce physical

xi. 모든 scripts

6. 구현 내용 실행 화면과 설명 & 주요 스크립트 코드

A. 플레이어 움직임

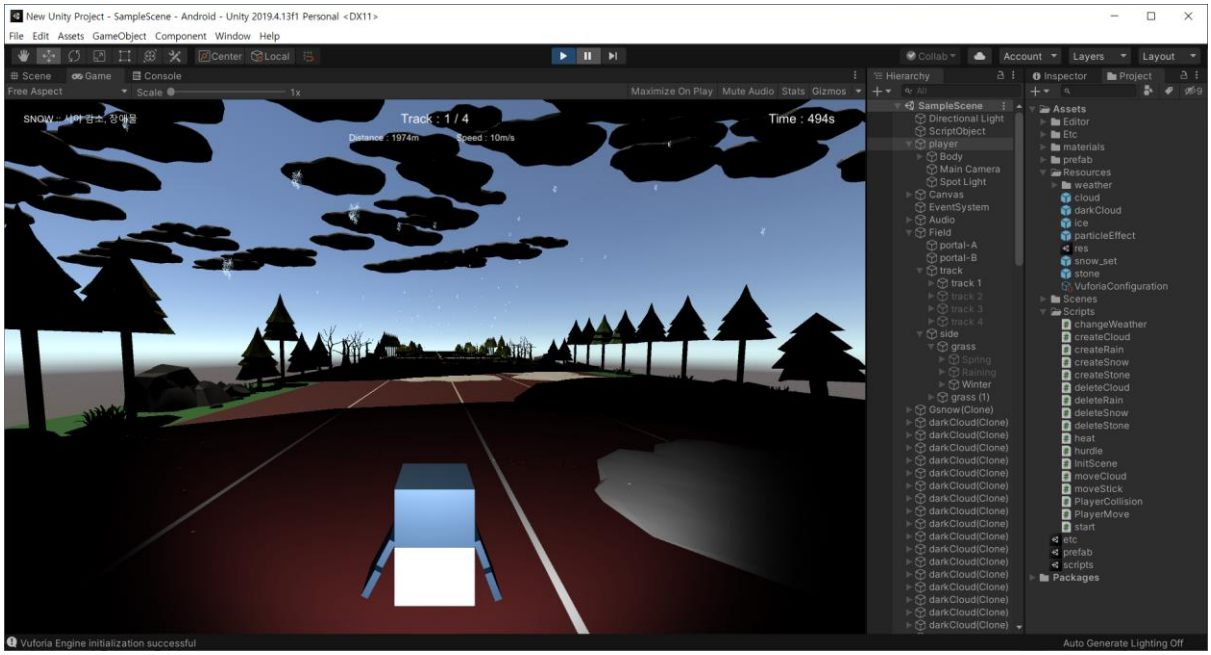


Figure 1 플레이어 좌우/앞 이동, 점프, 애니메이션

플레이어는 위/왼/오른쪽 화살표와 스페이스바로 뒤로 이동을 제외하고는 다 이동할 수 있다. 점프는 플레이어가 점프하지 않았을 때에만 점프가 가능하다. 즉 이단 점프는 불가능하다.

플레이어가 움직일 때 각 이동에 맞는 애니메이션이 실행된다.

```
// PlayerMove.cs :: player
    {

        . . .

        if(Input.GetKey(KeyCode.Space))

            anim.SetTrigger("jump");

        if(Input.GetKey(KeyCode.UpArrow))

            move1 = true;

        else

            move1 = false;

        if(Input.GetKey(KeyCode.LeftArrow))

            move2 = true;

        else if(Input.GetKey(KeyCode.RightArrow))

            move2 = true;
```

```
        else
            move2 = false;
        . . .
    }
    if(move1 || move2)
        anim.SetInteger("Walk", 1);
    else
        anim.SetInteger("Walk", 0);
```

B. 트랙 별 장애물 생성, 움직임과 소멸

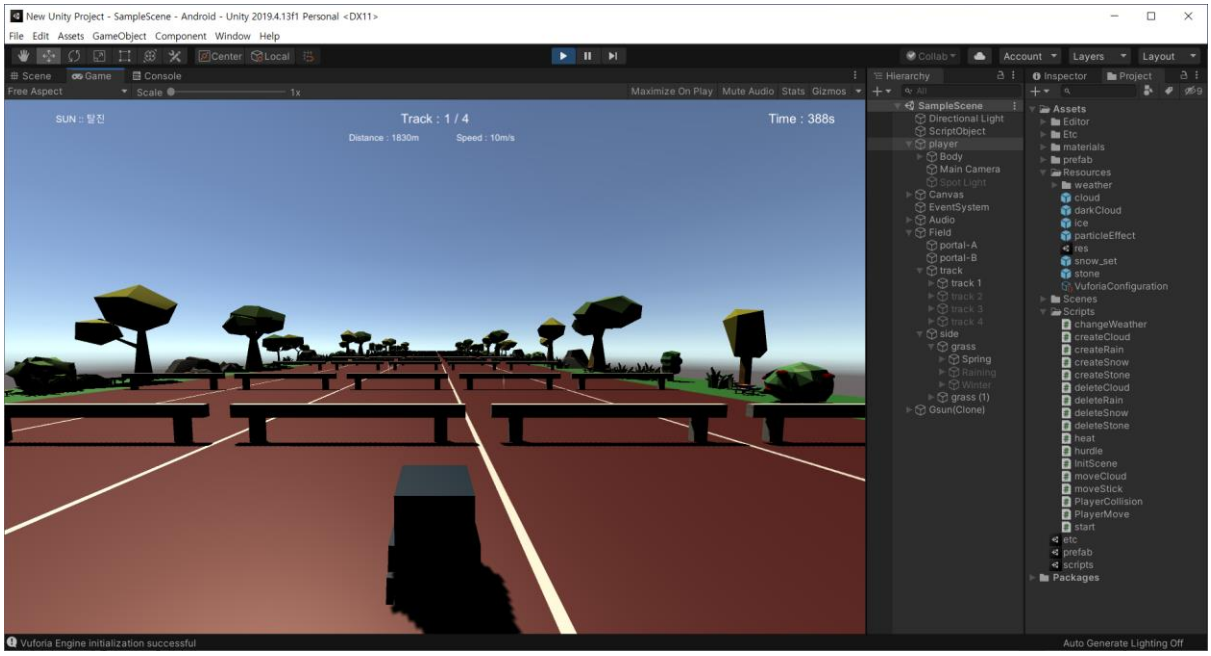


Figure 2 Track1 허들

Track1의 장애물은 허들로, 게임 실행 전에 미리 배치된다. 플레이어와 부딪히면 사라진다.

```
// hurdle.cs :: hurdle
    void OnCollisionEnter(Collision other){
        if(other.gameObject.tag == "Player")
            Destroy(gameObject);
    }
```

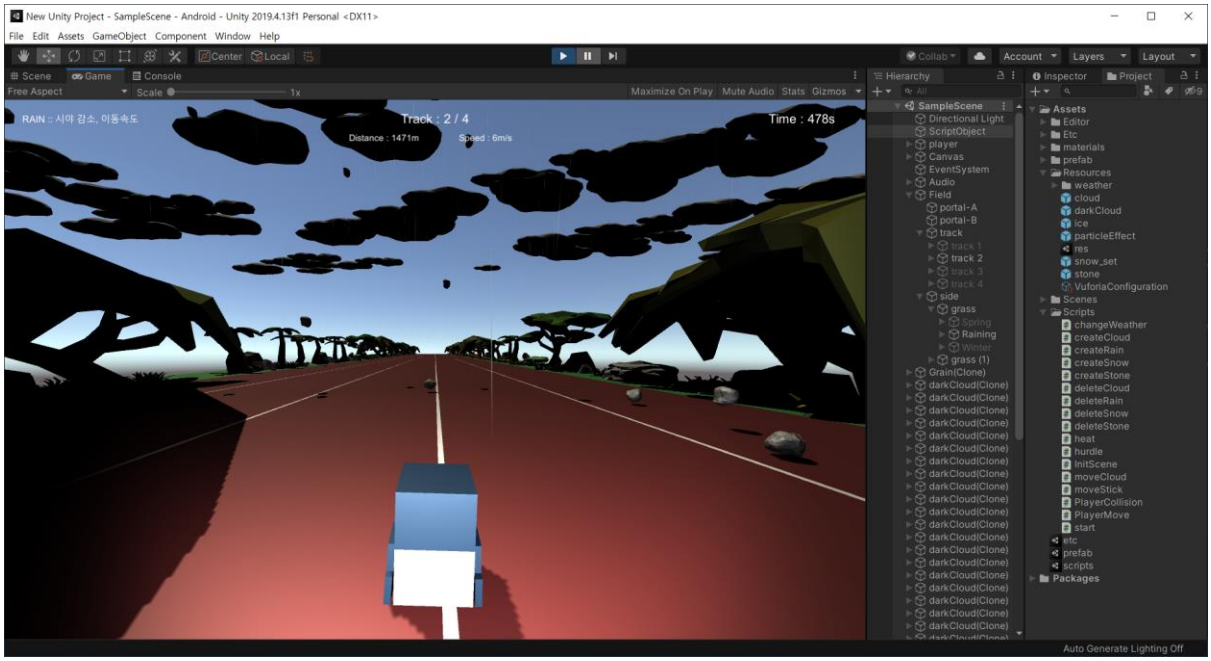


Figure 3 Track2 낙하하는 돌멩이

Track2의 장애물은 낙하하는 돌멩이로, 플레이어의 z 위치에 따라 랜덤으로 생성된다. rigidbody에서 설정한 중력에 의해 낙하하며 플레이어와 부딪히면 바로 사라지고, track에 부딪히면 2초간 바닥을 구르다가 소멸한다. 돌멩이의 그림자를 통해 낙하 지점을 예측할 수 있다.

```
// createStone.cs :: track2
    void FixedUpdate(){
        if(one && player.position.z > -240)
        StartCoroutine(create());
    }

    IEnumerator create(){ // InvokeRepeating 은 active 여부와 관계없이 계속 반복
        one = false;
        while(player.position.z < 200){
            x = Random.Range(-12, 12);
            z = Random.Range((int)player.position.z, (int)player.position.z + 40);
            Instantiate(stone, new Vector3(x*2.0f, 30, z), Quaternion.identity);
            yield return new WaitForSeconds(0.2f);
        }
    }

//deleteStone.cs :: stone
    void OnCollisionEnter(Collision other){
        if(other.gameObject.name == "player")    Destroy(gameObject);
        Else    Destroy(gameObject, 2f);
    }
```

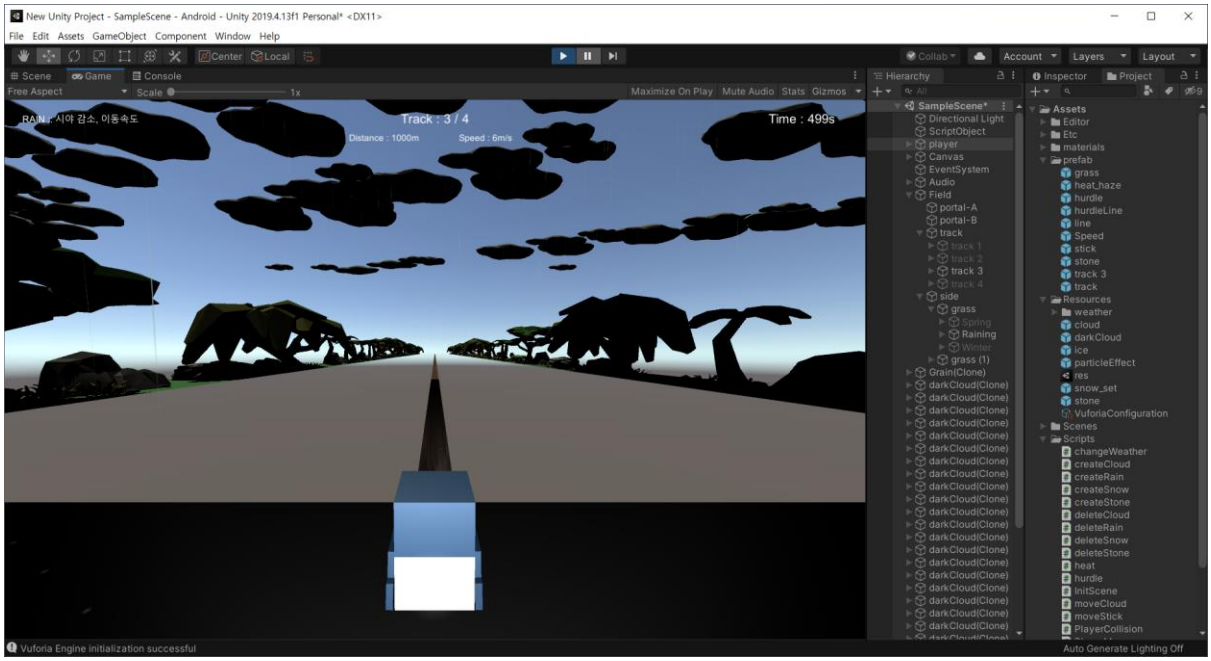


Figure 4 Track3 좁은 판

Track3은 좁은 판이 땅을 대신한다.

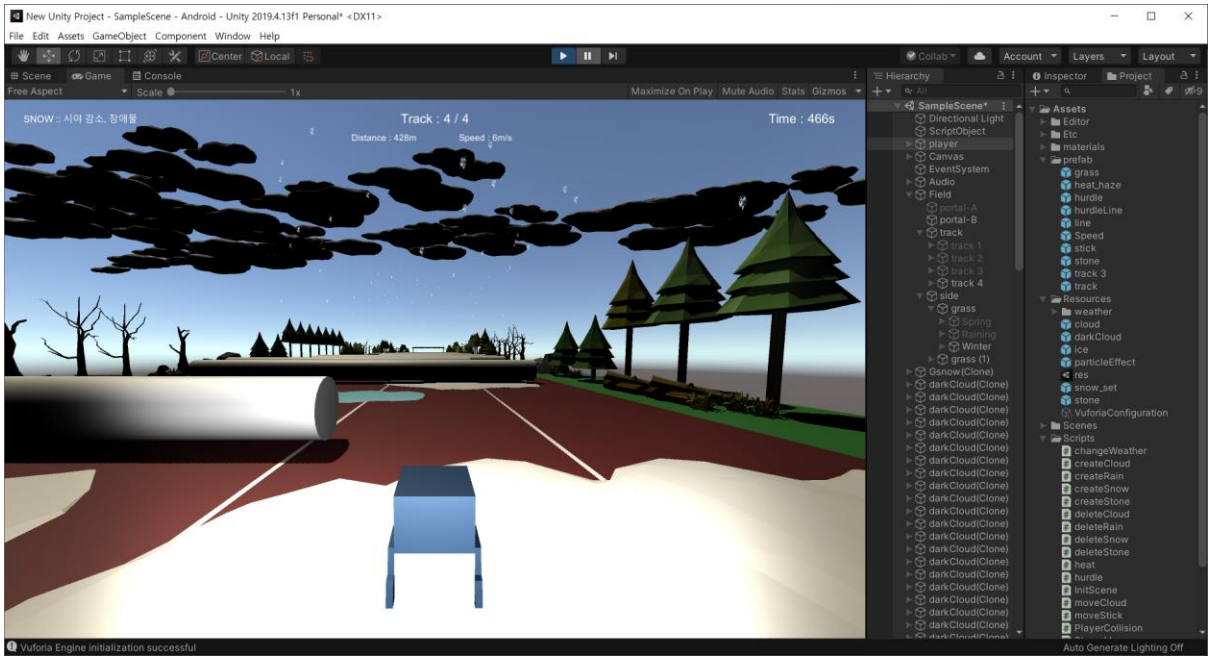


Figure 5 Track4 움직이는 막대기

Track4의 장애물은 움직이는 stick이다. 게임 실행 전에 미리 배치되고, x좌표 값이 변하는 이동(플레이어 기준 좌우 이동)을 한다.

```
// moveStick.cs
    void FixedUpdate(){
        transform.Translate(Vector3.up * speed * Time.deltaTime);
```

```
        if(transform.position.x >= 25 || transform.position.x <= -25)
        speed *= -1;
    }
```

C. 날씨 변화와 효과, 부가 생성물의 움직임과 소멸

날씨는 게임이 진행되는 동안 30.05초마다 랜덤으로 변하는 것을 Resources에 있는 object를 생성함으로써 관리한다. 직전의 랜덤 값과 현재의 랜덤 값이 다르면 생성된 object를 삭제하고 다른 object를 생성하며, 각 날씨에 따라 플레이어의 속도, 점프 높이, 배경 밝기를 변경한다.

```
// changeWeather.cs :: ScriptObject
    void FixedUpdate(){
        if(start){
            start = false;
            rand = Random.Range(0,4);
            pre = -1;
            StartCoroutine(change());
        if(!InitScene.oneTime)
            StopAllCoroutines();
    }

    IEnumerator change(){
        while(InitScene.oneTime){
            if(pre != rand){
                if(rand == 0)
                    instance1 = (GameObject) Instantiate(instance[rand] as GameObject, pos,
 Quaternion.identity);
                if(rand == 1)
                    instance1 = (GameObject) Instantiate(instance[rand] as GameObject, pos,
 Quaternion.identity);
                if(rand == 2)
                    instance1 = (GameObject) Instantiate(instance[rand] as GameObject, pos,
 Quaternion.identity);
                if(rand == 3)
                    instance1 = (GameObject) Instantiate(instance[rand] as GameObject, new
Vector3(0, 0, 0), Quaternion.identity);
            }
            yield return new WaitForSeconds(30.05f);
            delete();
        }
    }

   void delete(){
        if(InitScene.oneTime){
            pre = rand;
            rand = Random.Range(0,4);
```

```
            if(pre != rand)
                Destroy(instance1);
        }
    }
```

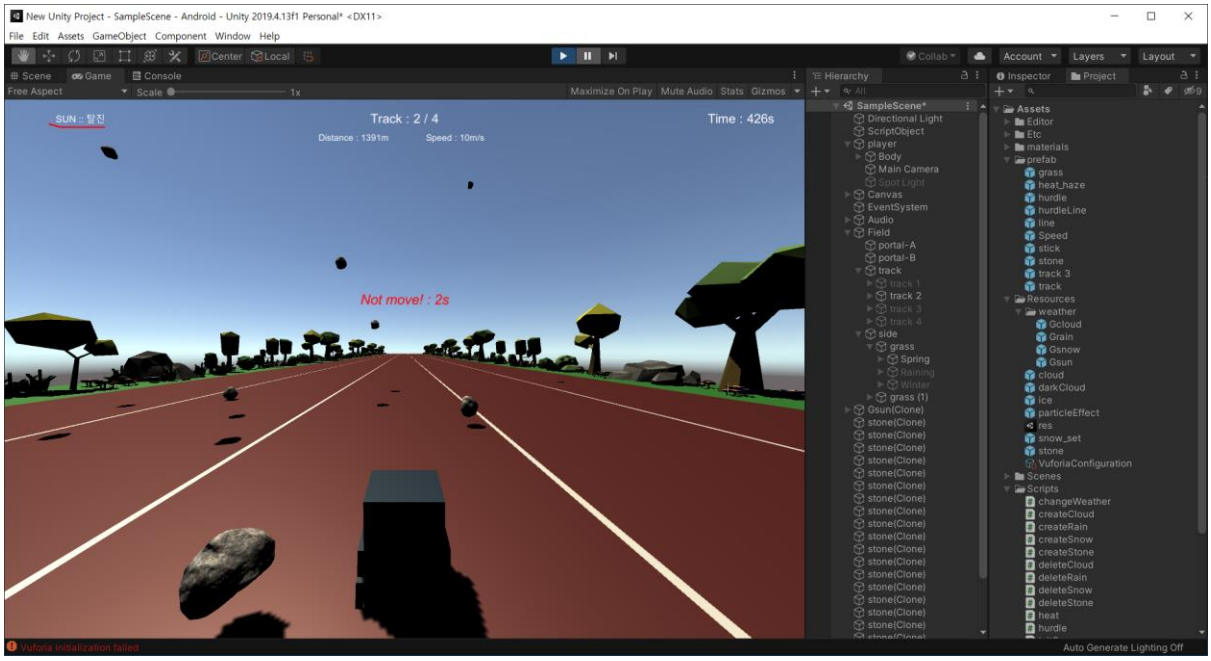


Figure 6 Gsun

Gsun에 의해 heat_haze가 생성된다. Gsun이 소멸되면 같이 소멸되고, 플레이어가 track3에 있다면 생성하지 않는다. 시야가 밝아지고 플레이어가 연속으로 15초 이상 달리면 3초간 움직이지 못한다.

```
// changeWeather.cs :: ScriptObject
    if(rand == 3)
        dl.GetComponent<Light>().intensity = 1.2f;

// heat.cs :: Gsun
    void FixedUpdate(){
        if(GameObject.Find("track 3"))
            heat_haze.Stop();
        else if(!GameObject.Find("track 3"))
            heat_haze.Play();
    }

// PlayerMove.cs :: player
    if(Input.GetKey(KeyCode.UpArrow)){
        transform.Translate(Vector3.forward * speed * Time.deltaTime);
        if(GameObject.FindWithTag("Gsun"))
           sun += Time.deltaTime;
        else
```

```
            sun = 0;
    }
    Else
        sun = 0;
    if(PlayerCollision.notMove == 0)
        sun = 0;
    . . .
    if(sun > 15){
        gameObject.GetComponent<PlayerCollision>().notMove = 3;
        sun = 0;
    }
```

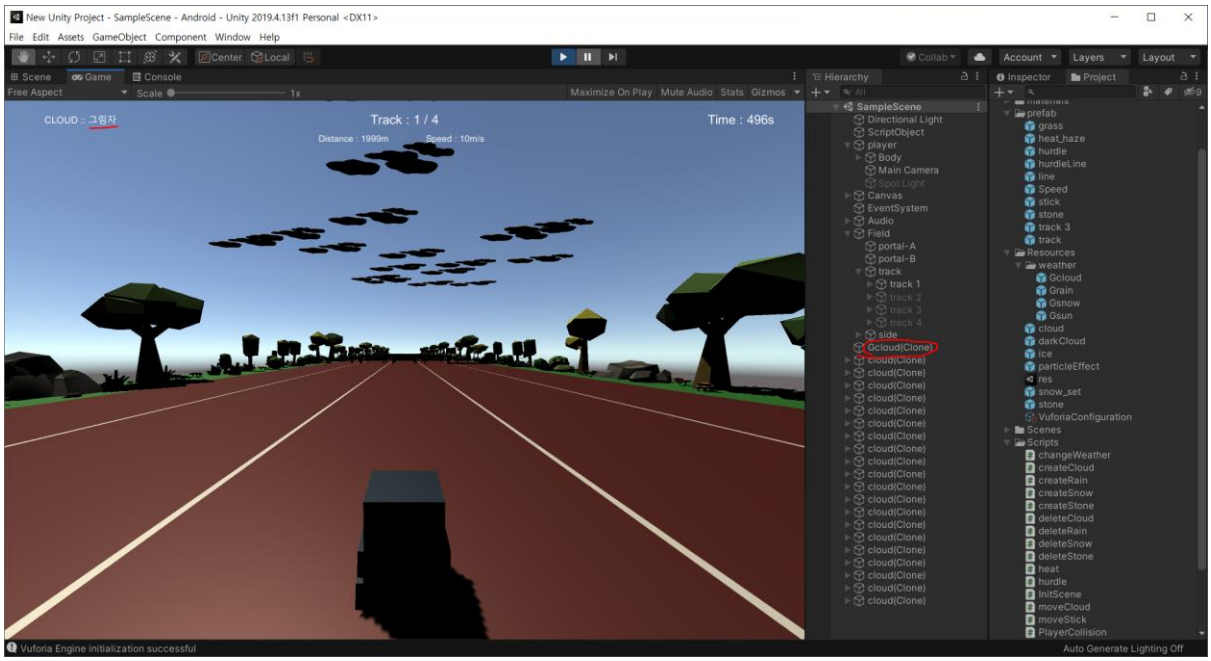


Figure 7 Gcloud

Gcloud에 의해 cloud가 임의의 위치에 생성된다. Cloud는 +x 방향으로 천천히 이동하고, Gcloud가 소멸하면 같이 소멸한다. 구름에 의한 그림자 외에는 날씨 효과가 없다.

```
// changeWeather.cs :: ScriptObject
    if(rand == 0)
        dl.GetComponent<Light>().intensity = 1f;

// createCloud.cs :: Gcloud
    cloud = Resources.Load("cloud") as GameObject;
    for(int i = 0; i < 20; i++){
        x = Random.Range(-10, 10);
        z = Random.Range(-8, 20);
        Instantiate(cloud, new Vector3(x*30.0f, 200, z*30.0f), Quaternion.identity);
    }
```

```
// moveCloud.cs :: cloud
    transform.Translate(Vector3.right * 5f * Time.deltaTime);

// deleteCloud.cs :: cloud
    if(!GameObject.FindWithTag("Gcloud"))
        Destroy(gameObject);
```

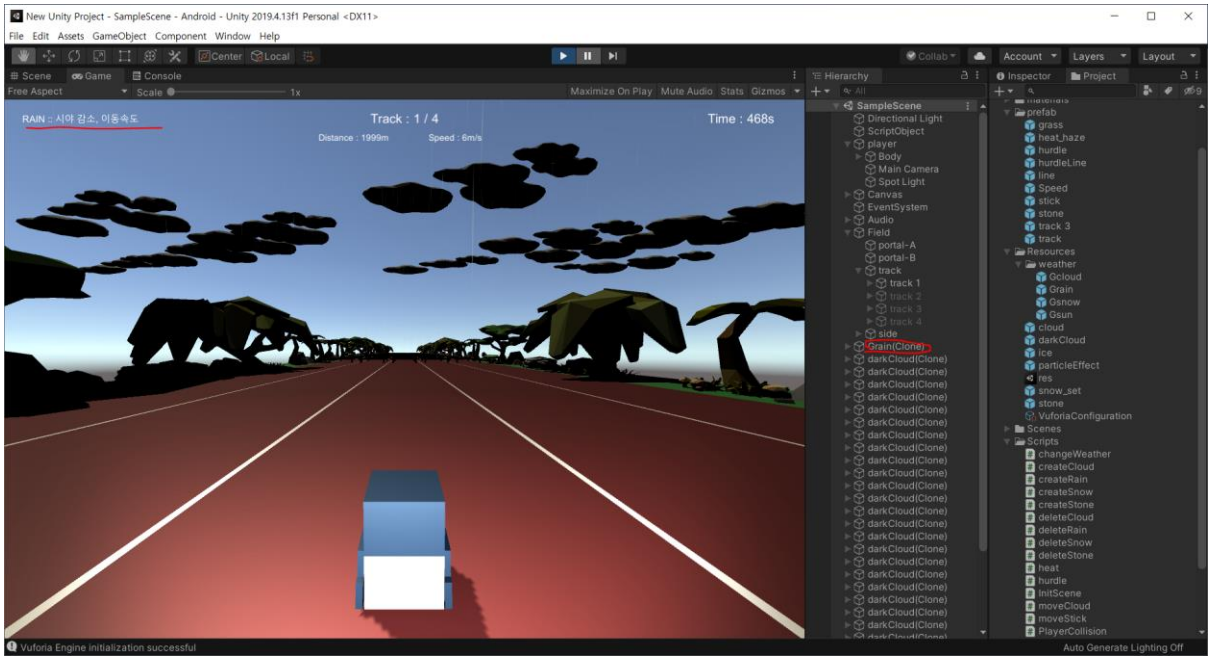


Figure 8 Grain

Grain에 의해 darkCloud가 임의의 위치에 생성되고 비가 내린다. darkCloud는 cloud와 마찬가지로 +x 방향으로 천천히 이동하고, Grain와 Gsnow가 둘 다 없으면 소멸한다. 비는 particle로 구현하여 바닥에 닿을 때쯤 자동으로 사라지게 구현했다. 날씨 효과로는 시야가 감소하고 플레이어의 이동속도도 감소한다.

```
// changeWeather.cs :: ScriptObject
    if(rand == 1){
        dl.GetComponent<Light>().intensity = 0.8f;
        PlayerMove.speed -= 4;
        PlayerCollision.initSpeed -= 4;
    }

// createRain.cs :: Grain
    darkCloud = Resources.Load("darkCloud") as GameObject;
    for(int i = 0; i < 30; i++){
        x = Random.Range(-25, 25);
        z = Random.Range(-7, 22);
        Instantiate(darkCloud, new Vector3(x*40.0f, 300, z*40.0f), Quaternion.identity);
     }
```

```
// moveCloud.cs :: darkCloud
    transform.Translate(Vector3.right * 5f * Time.deltaTime);

// deleteRain.cs :: darkCloud
    if(!GameObject.FindWithTag("Grain") && !GameObject.FindWithTag("Gsnow"))
        Destroy(gameObject);
```

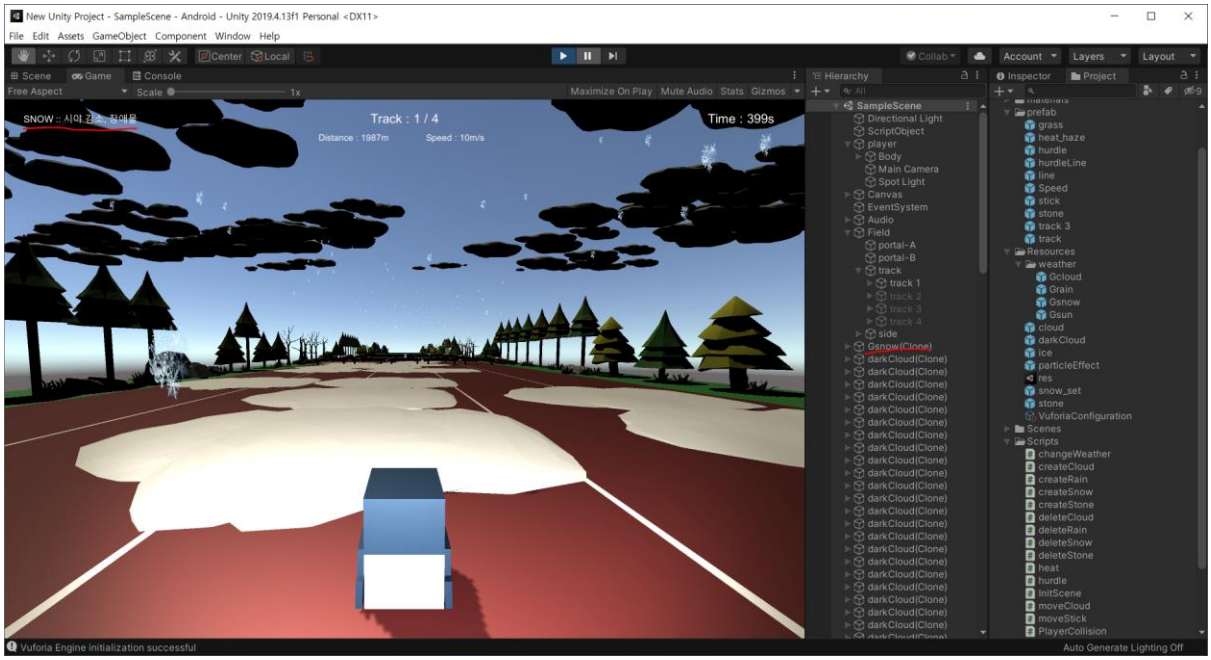


Figure 9 Gsnow

Gsnow에 의해 darkCloud가 임의의 위치에 생성되고 눈이 내리며 track위에 빙판과 눈덩이가 생성된다. 플레이어가 track3에 있다면 빙판과 눈덩이를 소멸시킨다. 눈은 비와 마찬가지로 particle로 구현하여 바닥에 닿을 때쯤 자동으로 사라지게 구현했다. 날씨 효과로는 시야가 감소한다.

```
// changeWeather.cs :: ScriptObject
    if(rand == 2)
        dl.GetComponent<Light>().intensity = 0.8f;

// createSnow.cs
    void FixedUpdate(){
        if(++count == 50)
            CreateSub();
        if(count > 50 && !GameObject.FindWithTag("track") &&!GameObject.FindWithTag("snow_s
et"))
            CreateSub();
    }

    void CreateSub(){
        for(int i = 0; i < 15; i++){
```

```
            x = Random.Range(-10, 10);
            z = Random.Range(-23, 23);
            Instantiate(ice, new Vector3(x*2.0f, 0.02f, z*10.0f), Quaternion.identity);
        }
        for(int i = 0; i < 80; i++){
            x = Random.Range(-10, 10);
            z = Random.Range(-23, 23);
            Instantiate(snow_set, new Vector3(x*2.0f, 0.02f, z*10.0f), Quaternion.identity)
;
        }
    }
```

D. 플레이어의 충돌

i. 포탈을 통한 플레이어의 이동

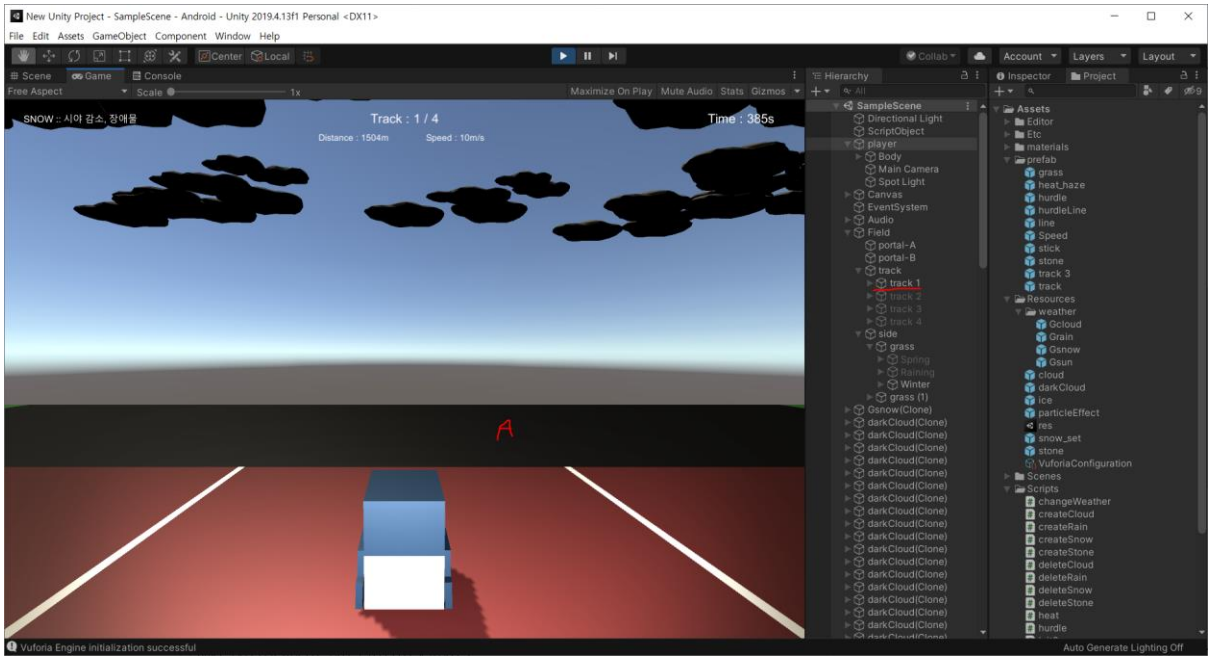


Figure 10 portal-A

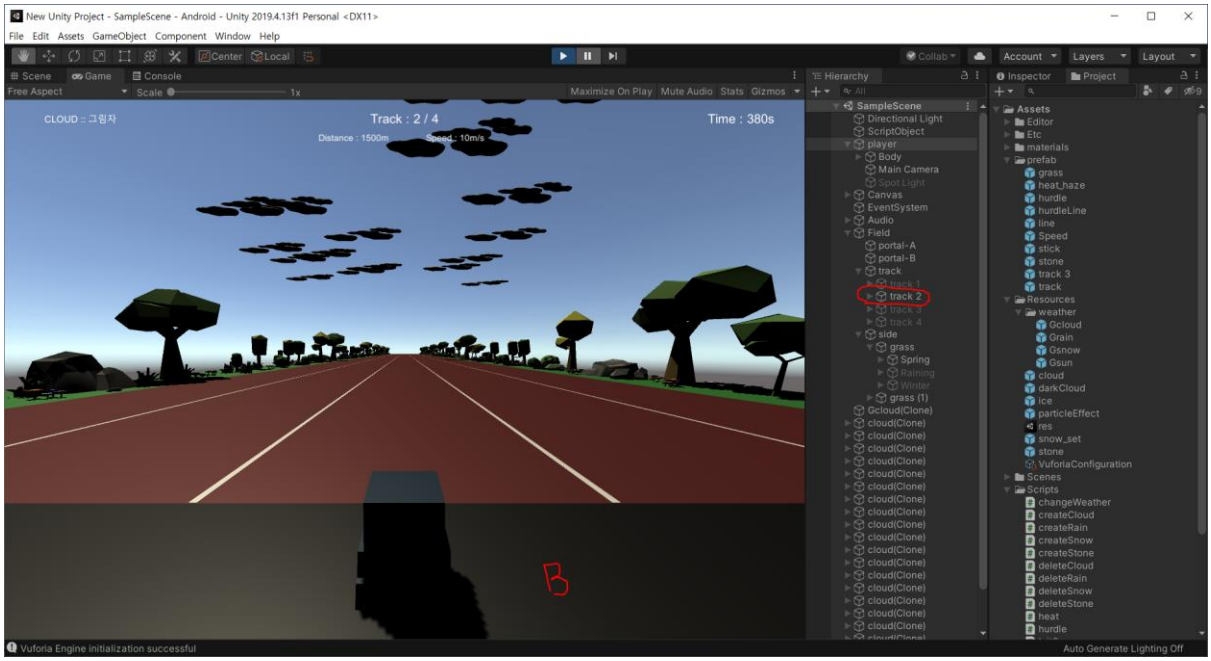


Figure 11 portal-B

Track1, Track2, Track3의 끝 부분에 있는 portal-A를 통해 Track2, Track3, Track4의 시작 부분에 있는 portal-B로 이동할 수 있다. 각 track의 끝 부분에서 portal-A와 충돌하면 바로 직전의 track을 비활성화 시키고, 다음 track을 활성화 시킨다. Track3에는 portal-A도 비활성화 시킨다.

플레이어가 Track3과 Track4에서 떨어져 그 아래에 존재하는 portal-C에 충돌하면 플레이어를 해당 트랙의 처음 위치, 즉 portal-B 위치로 이동시킨다.

```
// PlayerCollision.cs :: Player
    void OnCollisionEnter(Collision other){
        if(other.gameObject.name == "portal-C")
            transform.position = portalB.GetComponent<Transform>().position;
        if(other.gameObject.name == "portal-A"){
            if(track1.activeSelf){
                track1.SetActive(false);
                track2.SetActive(true);
            }
            else if(track2.activeSelf){
                track2.SetActive(false);
                track3.SetActive(true);
            }
            else if(track3.activeSelf){
                track3.SetActive(false);
                portalA.SetActive(false);
                track4.SetActive(true);
            }
            transform.position = portalB.GetComponent<Transform>().position;
        }
```

```
        if(onePort && other.gameObject.name == "portal-B"){
            instance = Instantiate(effect, new Vector3(transform.position.x, 0.5f, transfor
m.position.z-1), Quaternion.Euler(-90, 0, 0));
            Destroy(instance, 5f);
            onePort = false;
        }
        Else
            onePort = true;
    }
```

ii. 장애물

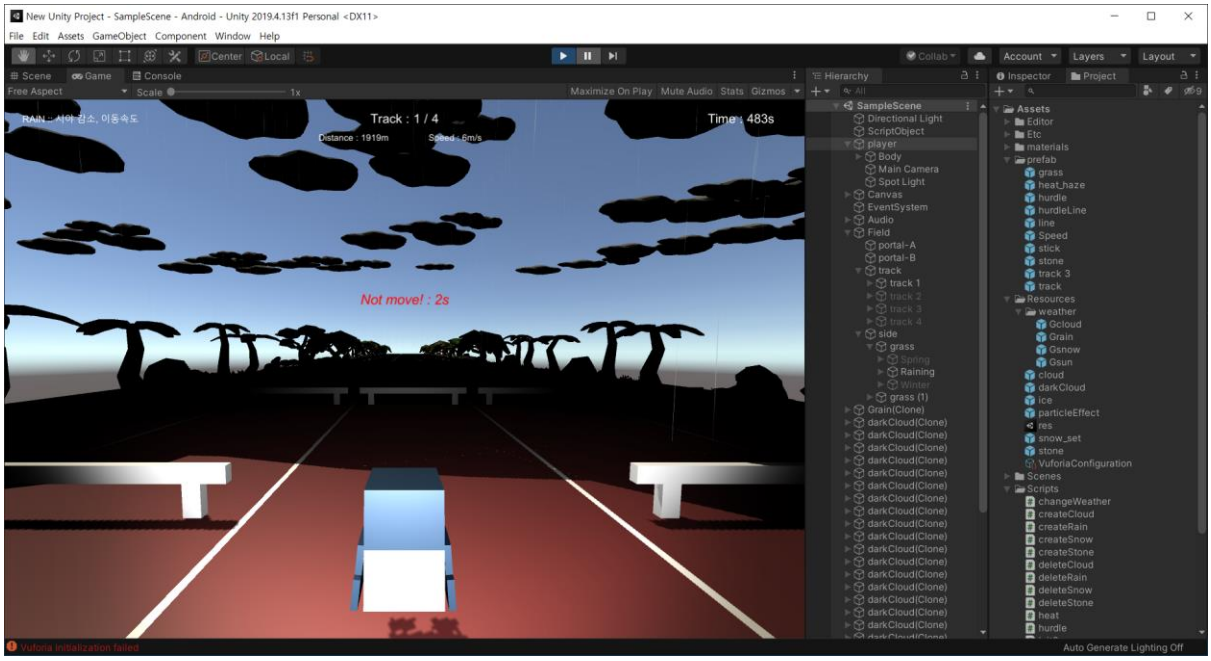


Figure 12 hurdle과 충돌

플레이어가 허들과 충돌하면 플레이어는 뒤로 튕겨지고 충돌한 순간부터 3초간 움직일 수 없다.

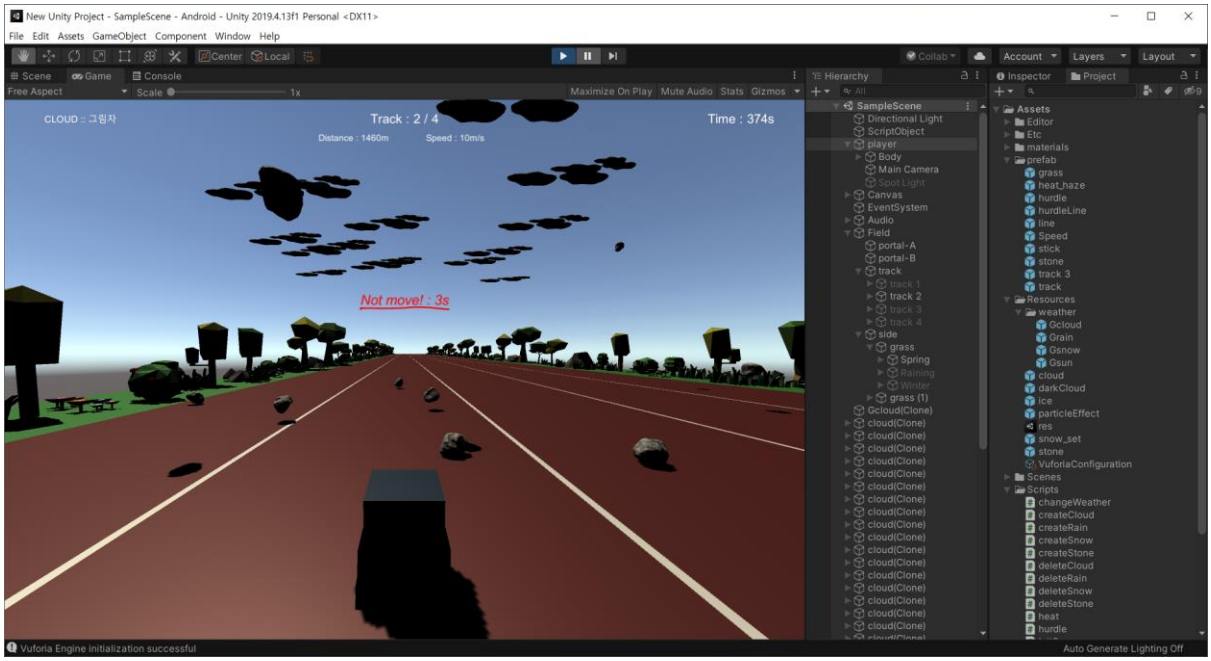


Figure 13 stone과 충돌

낙하 중인 돌멩이와 충돌하든 바닥에 굴러다니는 돌멩이와 충돌하든, 돌멩이와 충돌하면 플레이어는 그 자리에서 3초간 움직일 수 없다.

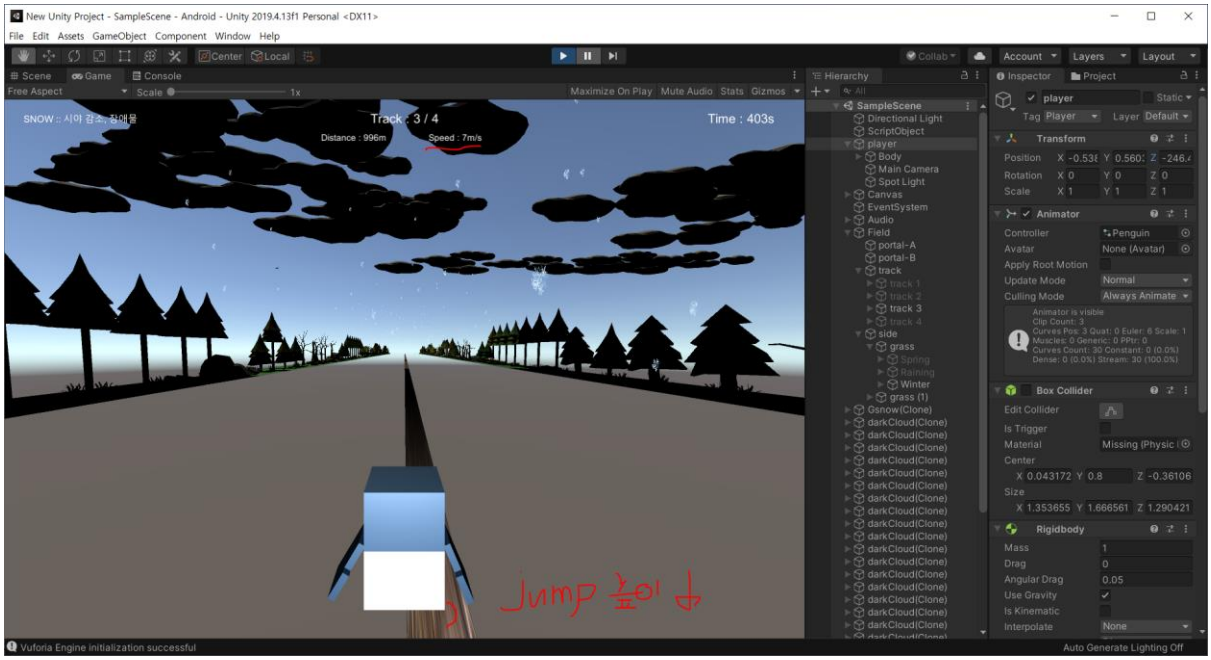


Figure 14 board와 충돌

플레이어가 board 위에 있을 때 플레이어의 이동속도와 점프 높이가 낮아지고 플레이어의 x 좌표가 미세하게 변한다.

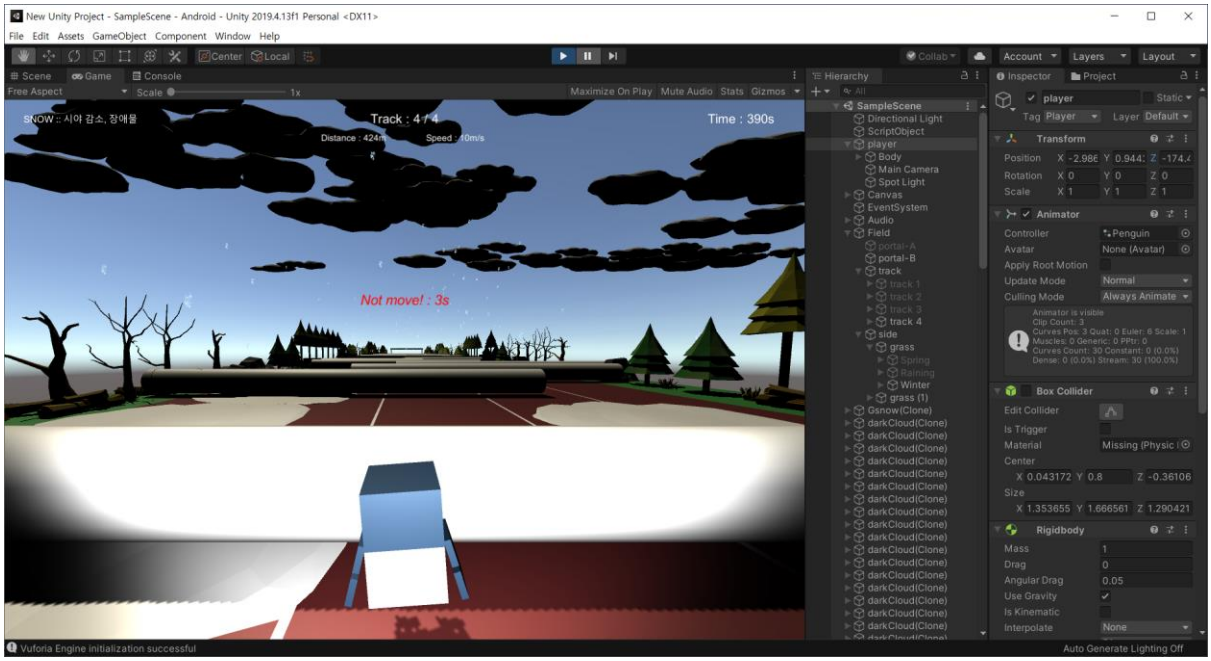


Figure 15 stick과 충돌

플레이어가 막대기와 충돌하면 바로 뒤로 튕겨 나가고, 충돌한 순간부터 3초간 움직일 수 없다.

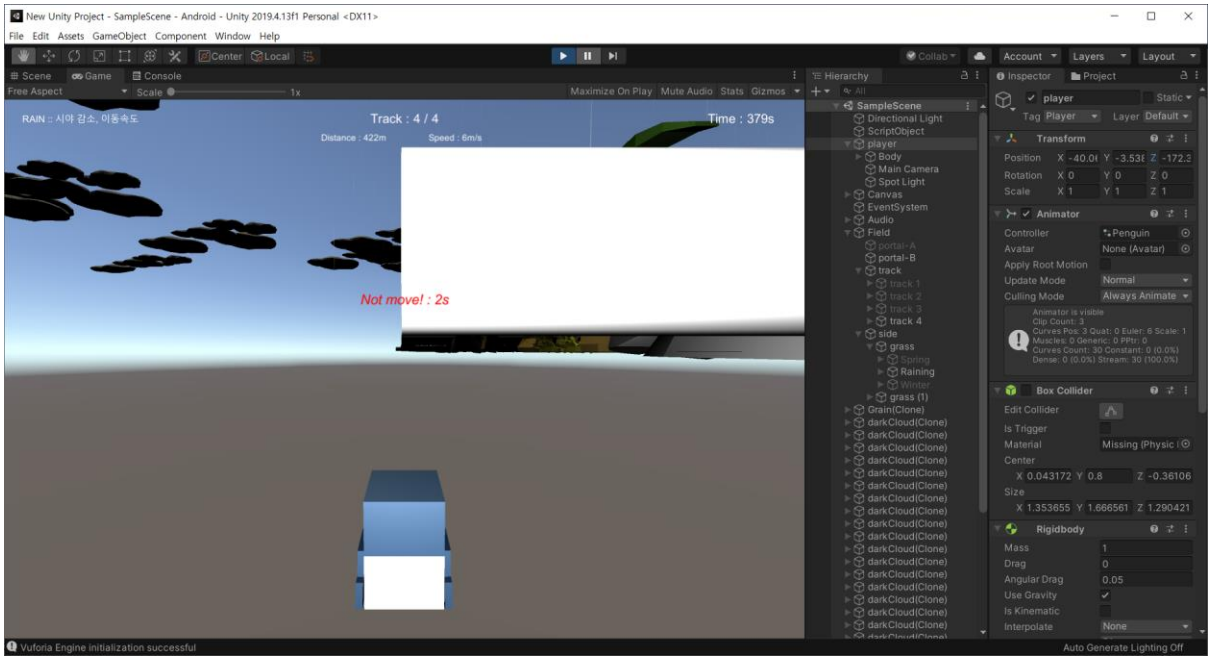


Figure 16 stick에 의해 track에서 떨어지는 플레이어

Track3에서 board에서 떨어지는 것 외에도 플레이어는 Track4의 stick에 의해 밀려나서 떨어질 수 있다.

```
// PlayerCollision.cs :: Player
    void OnCollisionEnter(Collision other){
        if(other.gameObject.tag == "hurdle"){
```

```
            GetComponent<Rigidbody>().AddForce(Vector3.back * 150f);
            notMove = notMoveTime;
        }
        if(other.gameObject.tag == "stone")
            notMove = notMoveTime;
        if(other.gameObject.tag == "stick"){
            GetComponent<Rigidbody>().AddForce(Vector3.back * 150f);
            notMove = notMoveTime;
        }
        if(other.gameObject.name == "board"){
            PlayerMove.jump = 30f;
            PlayerMove.speed = initSpeed - 3;
        }
    }

    void OnCollisionStay(Collision other){
        if(other.gameObject.name == "board")
            transform.position = new Vector3(transform.position.x + Random.Range(-
3, 3) * 0.01f, transform.position.y, transform.position.z);
    }
```

iii. 날씨 부가 생성물

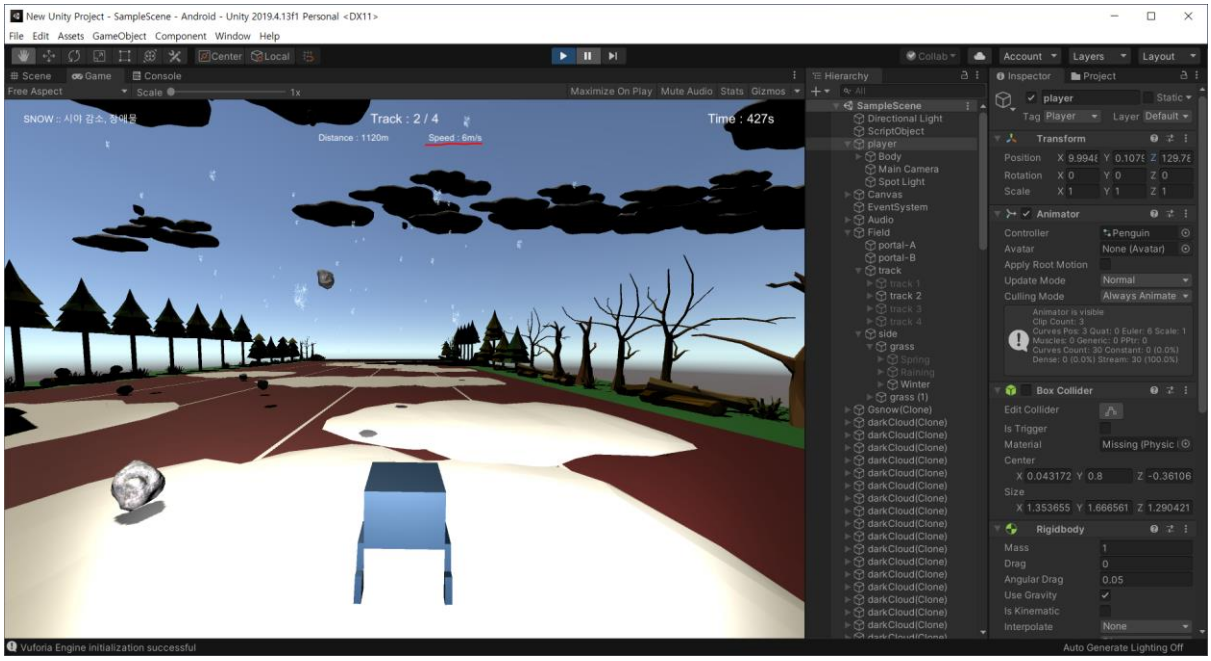


Figure 17 snow_set과 충돌

Snow_set 위에서 플레이어의 이동 속도가 감소하고, 플레이어의 x 좌표가 미세하게 변한다.

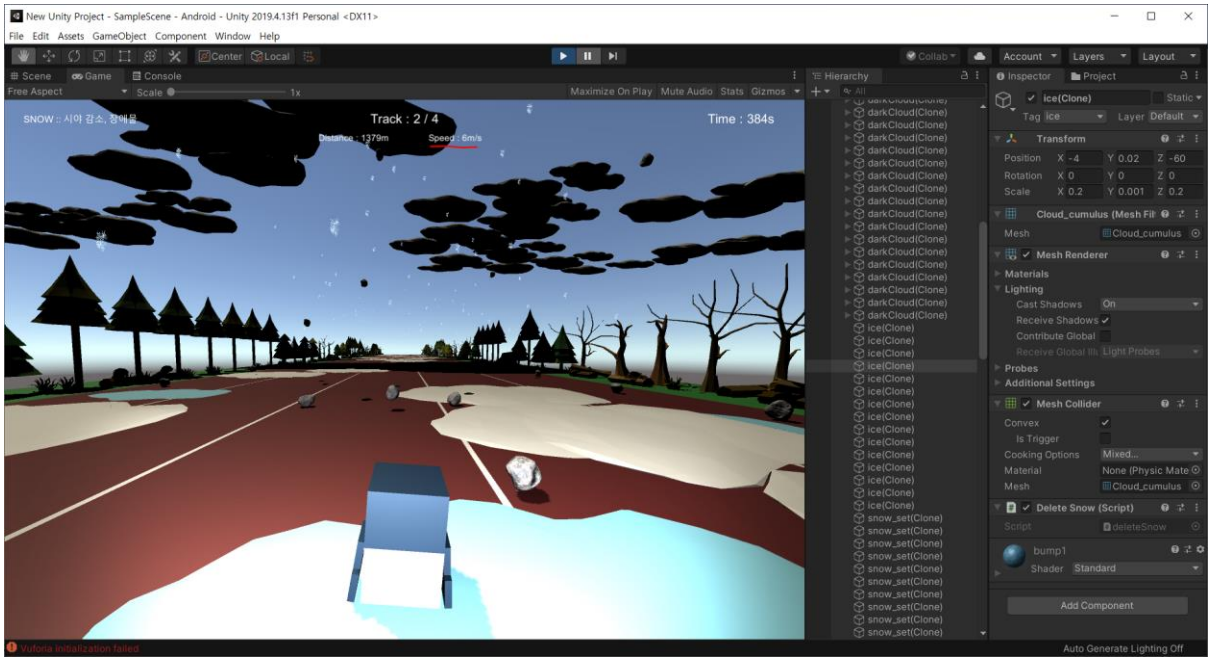


Figure 18 ice와 충돌

빙판 위에서 플레이어의 이동속도와 점프력이 감소한다.

```
// PlayerCollision.cs :: Player
    void OnCollisionEnter(Collision other){
        if(other.gameObject.tag == "snow_set")
            PlayerMove.speed = initSpeed - 4;
        if(other.gameObject.tag == "ice"){
            PlayerMove.jump = 30f;
            PlayerMove.speed = initSpeed - 4;
        }
    }

    void OnCollisionStay(Collision other){
        if(other.gameObject.tag == "snow_set")
            transform.position = new Vector3(transform.position.x + Random.Range(-
3, 3) * 0.01f, transform.position.y, transform.position.z);
    }
}
```

E. 그 외의 기능

i. 게임 시작 UI 동작

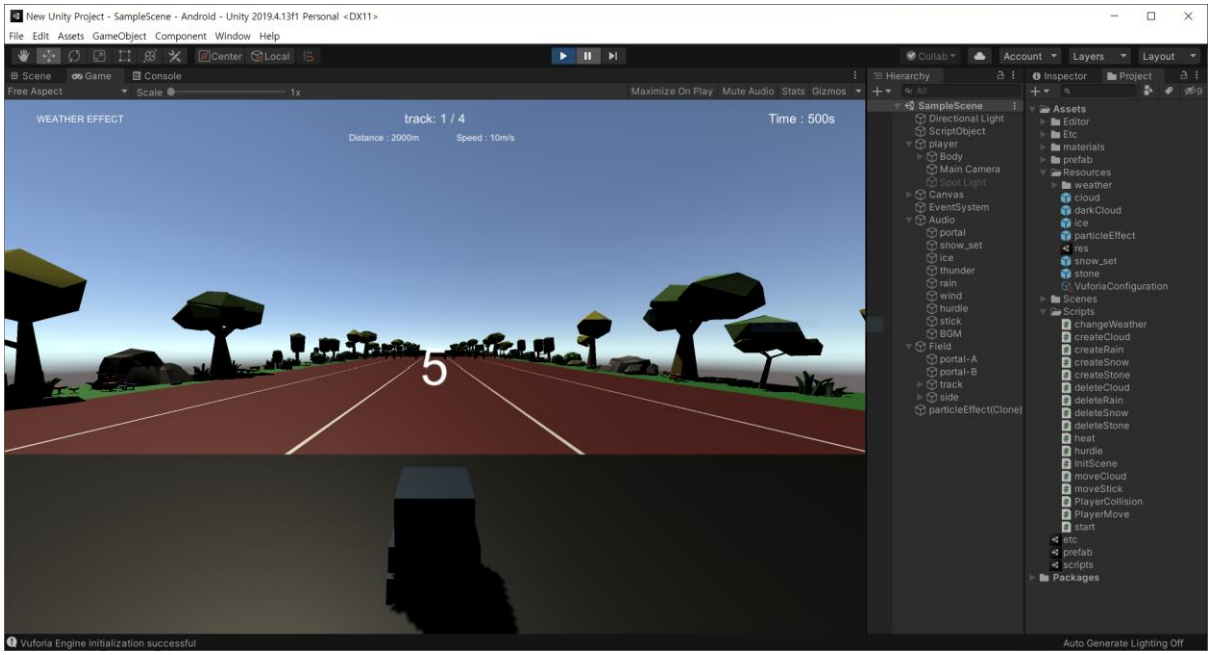


Figure 19 시작 화면 5초에서부터 카운트 다운

시작할 때 start.cs에서 5초간의 카운트 다운을 진행한다. 이때 플레이어는 움직일 수 없다.

Track1에서부터 시작하며, 제한시간은 500초, 거리는 2000m, 처음 시작 속도는 10m/s이다.

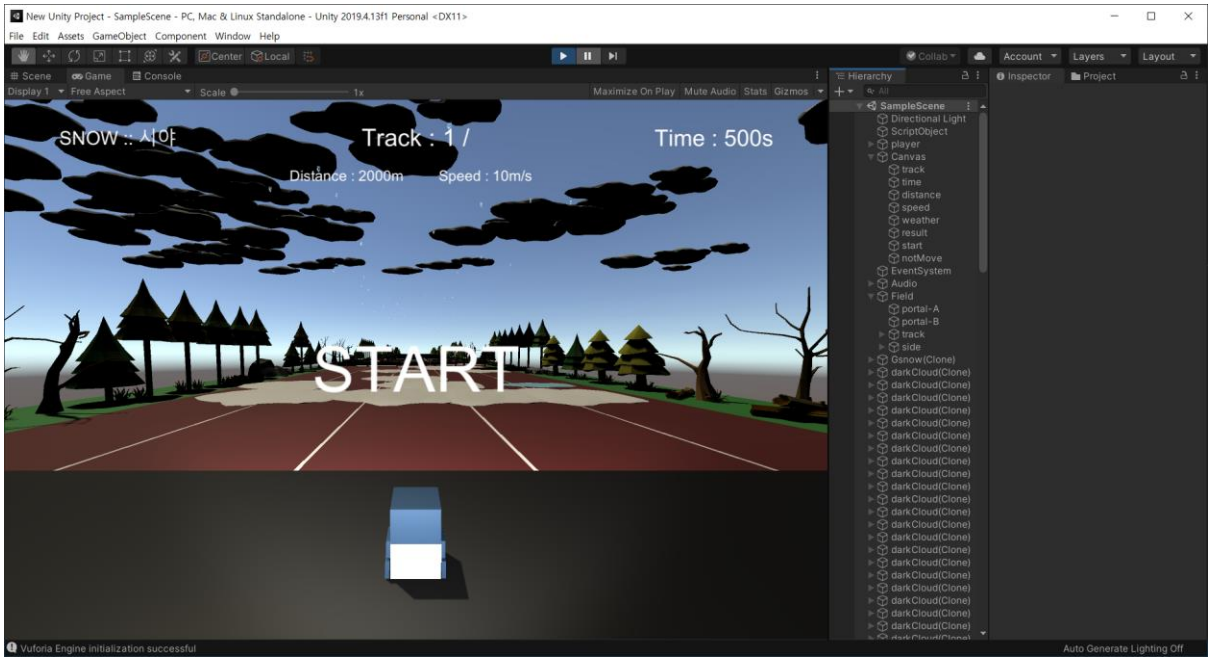


Figure 20 시작화면 카운트 다운 종료, "START"

"START" 문구가 나오면 bgm이 재생되고, 날씨 변화가 시작되며 움직일 수 있다.

```
// start.cs
    void FixedUpdate(){
        if(++timer % 50 == 0){
            if(timer % 50 == 0 && timer < 50*6)
                st.text = (--PlayerCollision.notMove - 1) + "";
            if(timer % 50 == 0 && timer == 50*6){
                st.text = "READY";
                --PlayerCollision.notMove;
            }
            if(timer % 50 == 0 && timer == 50*7){
                st.text = "START";
                --PlayerCollision.notMove;
                GetComponent<InitScene>().enabled = true;
                GetComponent<changeWeather>().enabled = true;
                GameObject.Find("player").GetComponent<PlayerMove>().bgmplay();
            }
            if(timer % 50 == 0 && timer == 50*8){
                st.enabled = false;
            }
        }
    }
```

```
// InitScene.cs :: ScriptObject
    void ShowText(){
        if(oneTime){
            trackText.text = "Track : " + track.ToString() + " / 4";
            distanceText.text = "Distance : " + ((int)distance).ToString() + "m";
            speedText.text = "Speed : " + PlayerMove.speed.ToString() + "m/s";
            if(timerCount % 50 == 0){
                timeText.text = "Time : " + (limit_time - timerCount/50).ToString() + "s";
            }
            switch(changeWeather.rand){ // cloud, rain, snow, sun
                case 0:
                    weatherText.text = "CLOUD :: 그림자";
                    break;
                case 1:
                    weatherText.text = "RAIN :: 시야 감소, 이동속도 감소";
                    break;
                case 2:
                    weatherText.text = "SNOW :: 시야 감소, 장애물 생성";
                    break;
                case 3:
                    weatherText.text = "SUN :: 탈진";
                    break;
            }
            if(PlayerCollision.notMove != 0)
                moveText.text = "Not move! : " + PlayerCollision.notMove + "s";
```

```
            else
                moveText.text = "";
        }
    }

    void getDistance(){
        if(player.transform.position.z < -250)
            z = -250;
        else
            z = player.transform.position.z;
        z = (z > 0) ? 250 + z : 250 + z;

        if(GameObject.Find("track 1")){
            distance = trackLength * 4 - z;
            track = 1;
        }
        else if(GameObject.Find("track 2")){
            distance = trackLength * 3 - z;
            track = 2;
        }
        else if(GameObject.Find("track 3")){
            distance = trackLength * 2 - z;
            track = 3;
        }
        else if(GameObject.Find("track 4")){
            distance = trackLength * 1 - z;
            track = 4;
        }
    }
}
```

ii. 게임 종료 화면

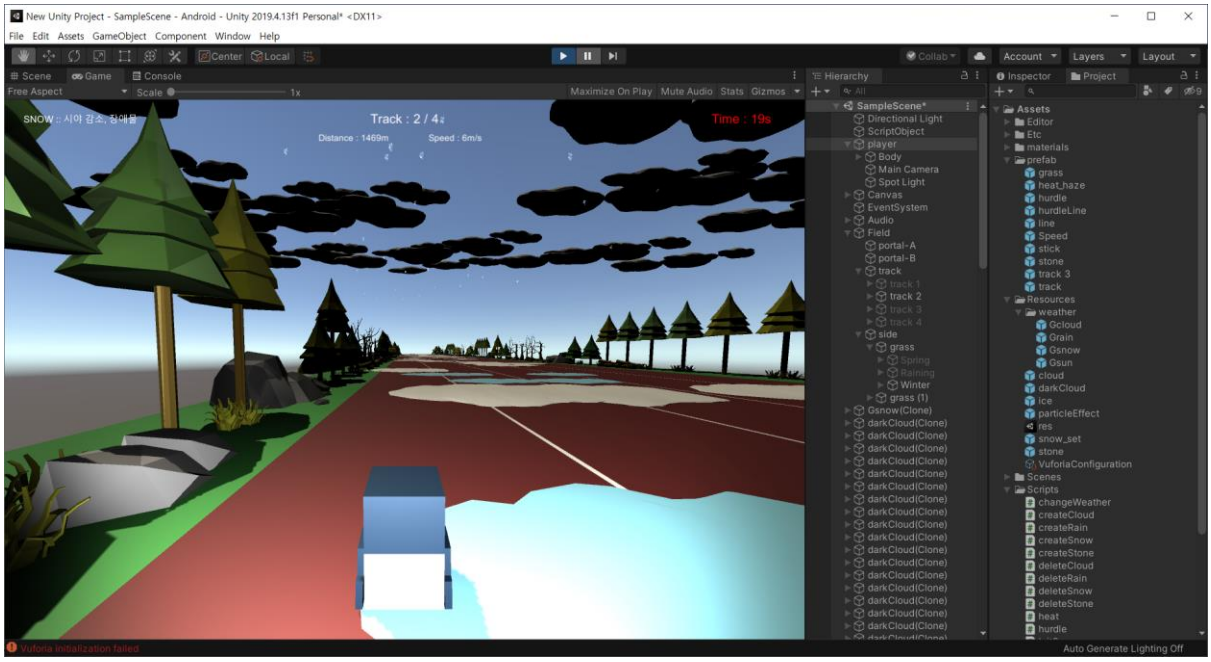


Figure 21 게임 종료까지 20초 이내

게임 종료까지 남은 시간이 20초 이하가 되면 화면 우측 상단에 시간을 표시하는 UI의 색을 red-white으로 바꿔간다.

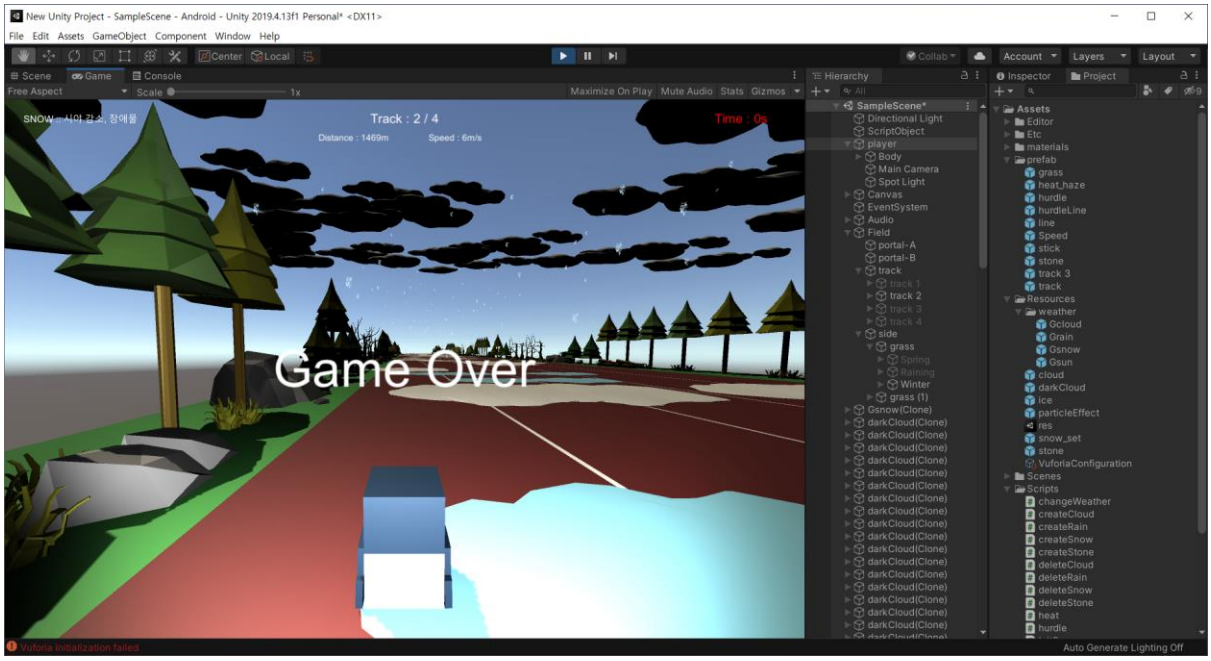


Figure 22 Game Over

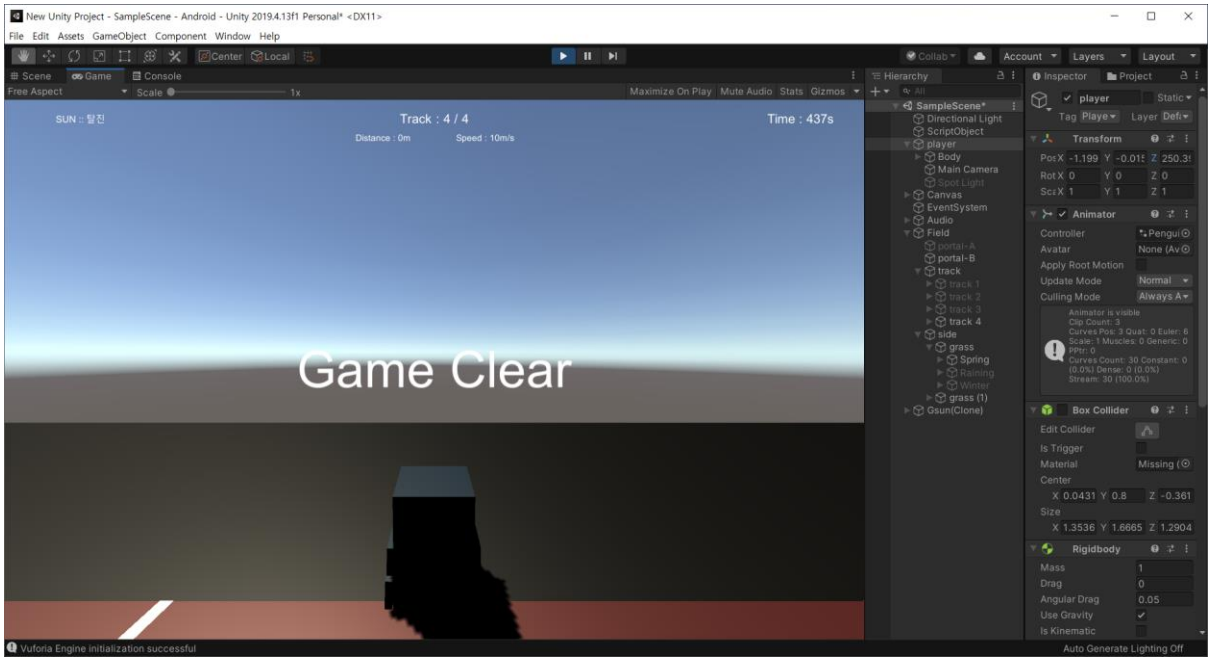


Figure 23 Game Clear

```
void FixedUpdate(){
    if(oneTime){
        if(distance <= 0){
            resultText.text = "Game Clear";
            oneTime = !oneTime;
        }
        else if(limit_time == ++timerCount/50){
            resultText.text = "Game Over";
            timeText.text = "Time : 0s";
            oneTime = !oneTime;
        }
        else{
            if(PlayerCollision.notMove != 0 && timerCount % 50 == 0)
                PlayerCollision.notMove --;
        }
    }
    if(timerCount/50 >= limit_time - 20){
        if(timerCount / 5 % 4 != 0)
            timeText.color =  Color.white;
        else
            timeText.color =  Color.red;
    }
}
```

iii. 날씨에 따른 Spot Light와 Grass

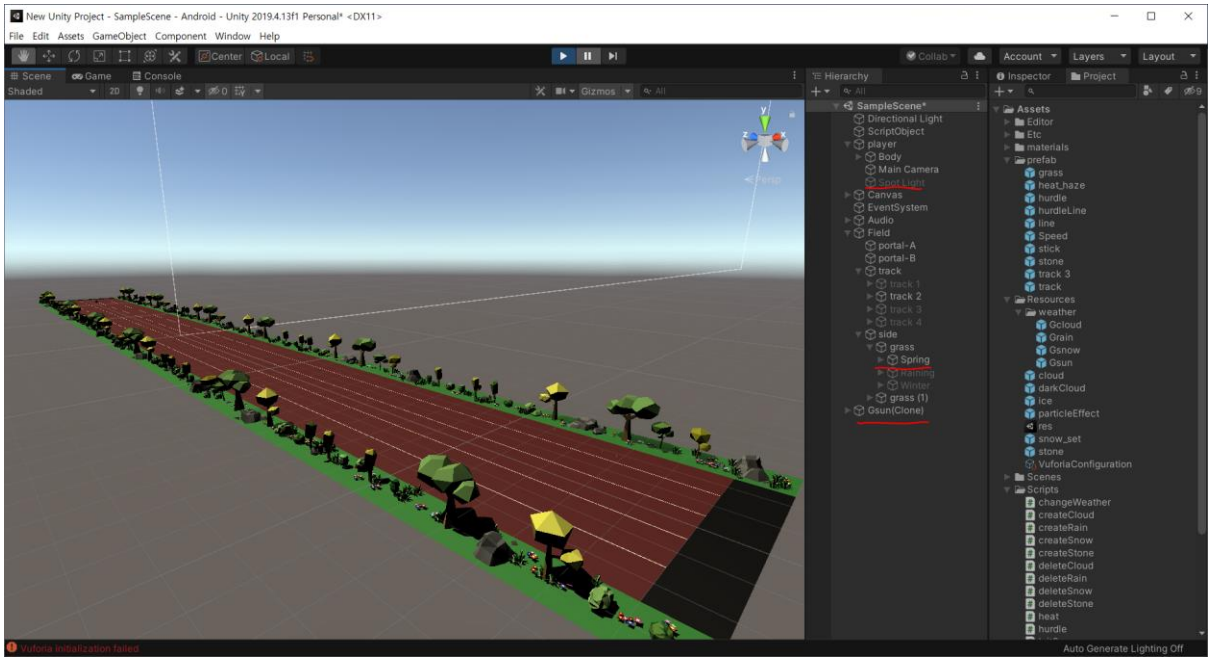


Figure 24 Gsun

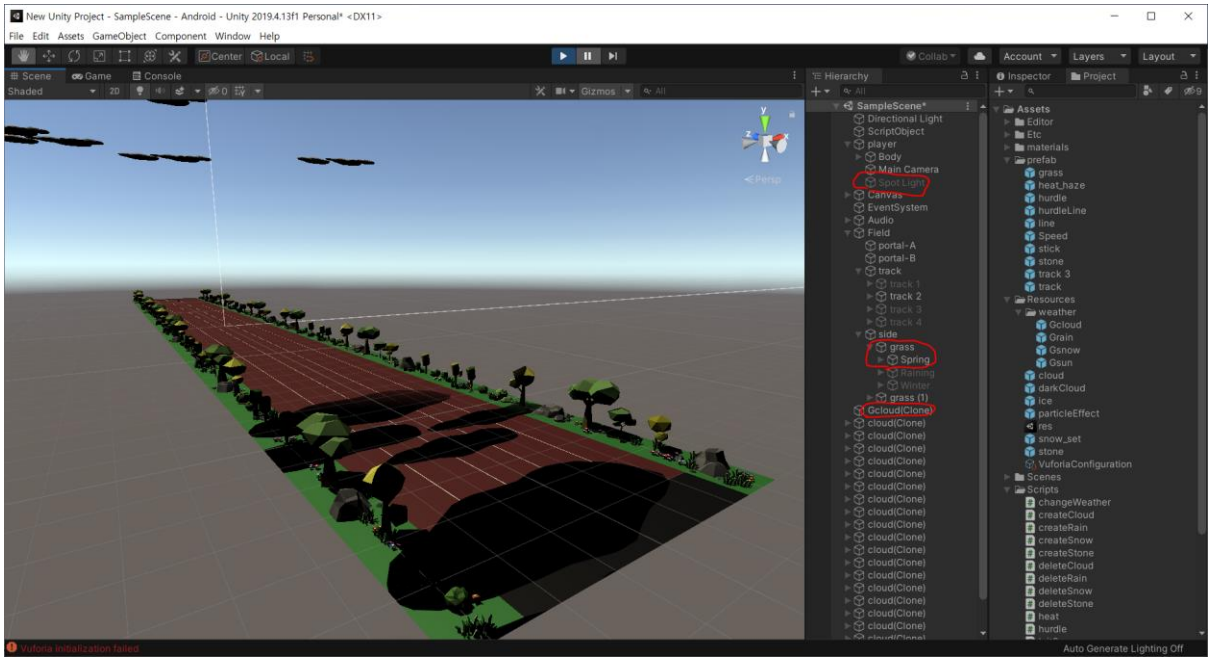


Figure 25 Gcloud

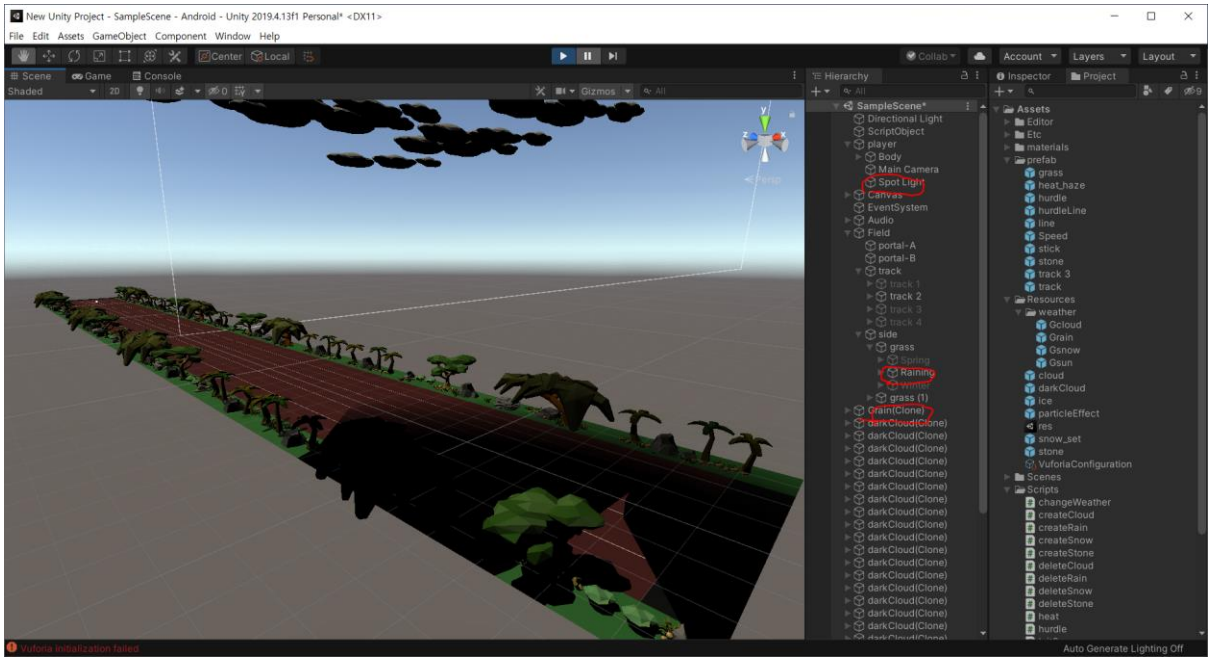


Figure 26 Grain

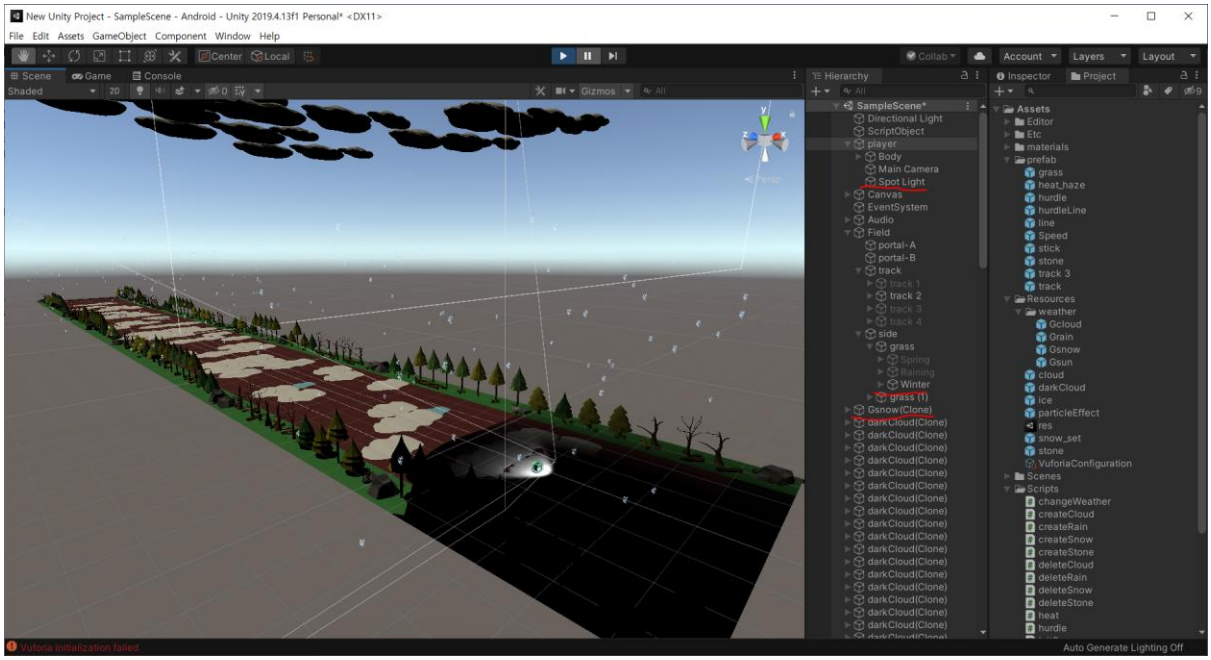


Figure 27 Gsnow

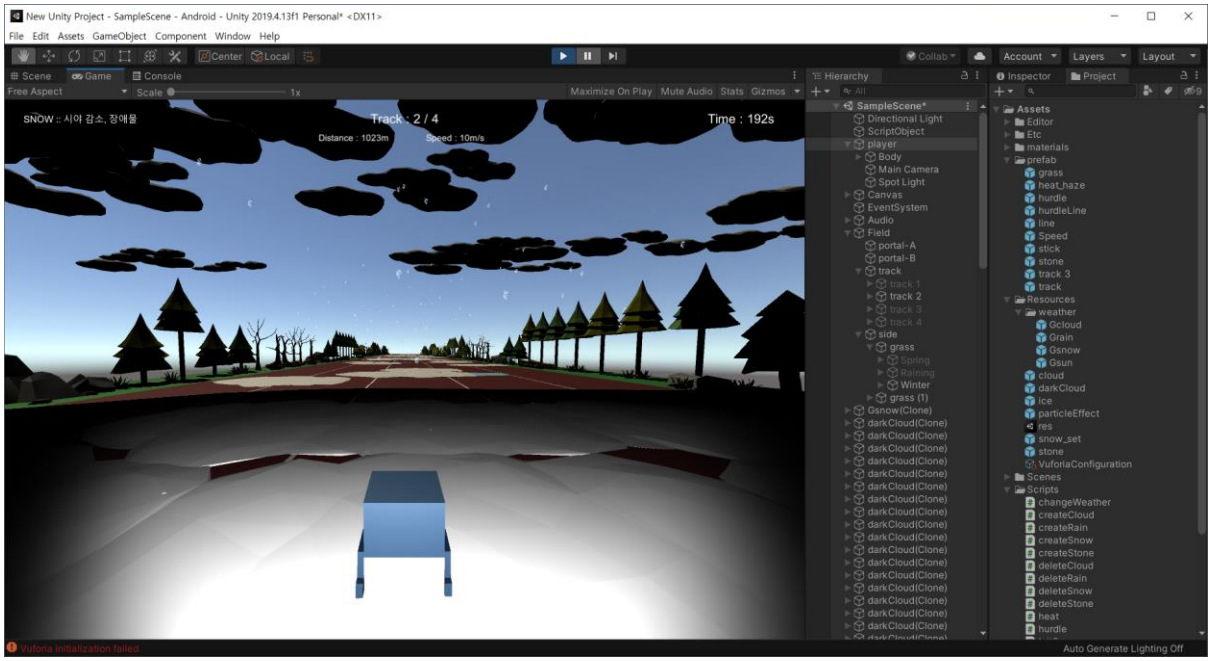


Figure 28 Spot Light

Gsun, Gcloud 때는 조명이 비활성화 상태이다가, Grain, Gsnow가 되면 조명이 활성화 상태로 바뀌어 제한된 시야를 확보한다.

Gsun, Gcloud 때는 spring 잔디가 활성화되고, Grain 때는 raining 잔디가, Gsnow 때는 winter 잔디가 활성화된다. 처음 게임을 시작할 때에는 spring으로 초기화한다.

```
// changeWeather.cs :: ScriptObject
    public GameObject spring, spring1, raining, raining1, winter, winter1;
    public GameObject sp;

    void change(){
        if(InitScene.oneTime){
            if(pre != rand){
                if(rand == 1){
                    raining.SetActive(true);
                    raining1.SetActive(true);
                }
                else{
                    raining.SetActive(false);
                    raining1.SetActive(false);
                }
                if(rand == 2){
                    winter.SetActive(true);
                    winter1.SetActive(true);
                }
                else{
                    winter.SetActive(false);
                    winter1.SetActive(false);
```

```
                }
                if(rand != 1 && rand != 2)
                    sp.SetActive(false);
                else
                    sp.SetActive(true);
                if(rand != 0 && rand != 3){
                    spring.SetActive(false);
                    spring1.SetActive(false);
                }
                else{
                    spring.SetActive(true);
                    spring1.SetActive(true);
                }
            }
        }
    }
```

iv. 음향

게임 시작을 위한 카운트 다운이 끝나면 bgm 오디오를 플레이한다. 플레이어의 y 좌표가 -1이하가 되면 플레이어가 track에서 떨어져 낙하하는 것이므로 bgm을 일시 중단하고, portal-C를 통해 다시 track위로 올라오면 중단된 bgm을 이어서 플레이한다.

```
// Start.cs :: ScriptObject
    GameObject.Find("player").GetComponent<PlayerMove>().bgmplay();

// PlayerMove.cs :: player
    public AudioSource bgm;

    void FixedUpdate(){
        if(InitScene.oneTime && PlayerCollision.notMove == 0) {
            if(transform.position.y < -1)
                bgm.Pause();
            else{
                bgm.UnPause();
                . . .
            }
        }
        if(!InitScene.oneTime)
            bgm.Stop();
        }
    }
```

플레이어가 hurdle, stick, portal, ice와 충돌하면 해당하는 오디오를 플레이한다.

플레이어가 board 위에 있으면 wind 오디오를 플레이한다. Board 위에서 점프할 때 음향이 여러 번 재생되지 않게 하기 위해 audio_wind 변수를 사용해서 1번만 재생되도록 관리한다.

플레이어가 snow_set 위에서 움직이고 있을 때에만 소리가 나야 한다. 만약 플레이어가 눈덩이 위에서 뛰거나 정지해 있으면 소리는 즉시 멈춰야 하고 플레이어가 여러 눈덩이에 연속으로 충돌할 경우 음향이 여러 번 겹쳐서 재생되지 않게 하기 위해 audio_snow 변수를 사용한다.

```
// PlayerCollision.cs :: player
    public AudioSource wind, hurdle, stick, snow_set, ice, portal;
    bool audio_snow, audio_wind;

    void OnCollisionEnter(Collision other){
        if(other.gameObject.tag == "hurdle")
            hurdle.Play();
        if(other.gameObject.tag == "stick")
            stick.Play();
        if(other.gameObject.name == "portal-A")
            portal.Play();
        if(other.gameObject.name == "board"){
            if(!audio_wind){
                wind.Play();
                audio_wind = true;
            }
        }
        else{
            wind.Stop();
            audio_wind = false;
        }
        if(other.gameObject.tag == "snow_set"){
            if(!PlayerMove.isJump && !audio_snow&& (PlayerMove.move1 || PlayerMove.move2)){
                snow_set.Play();
                audio_snow = true;
            }
        }
        else{
            snow_set.Stop();
            audio_snow = false;
        }
        if(other.gameObject.tag == "ice")
            ice.Play();
    }

    void OnCollisionStay(Collision other){
        if(other.gameObject.tag == "snow_set"){

            if(!(PlayerMove.move1 || PlayerMove.move2)){
                snow_set.Pause();
```

```
                audio_snow = false;
            }
            if(!audio_snow && (PlayerMove.move1 || PlayerMove.move2)){
                snow_set.Play();
                audio_snow = true;
            }
        }
    }

    void FixedUpdate(){
        if(audio_snow){
            if(PlayerMove.isJump){
                snow_set.Pause();
                audio_snow = false;
            }
        }
    }
```

Grain으로 인해 비가 내릴 때 천둥과 비 내리는 음향을 플레이한다.

```
// changeWeather.cs :: ScriptObject
    public AudioSource rain, thunder;

    void change(){
        if(InitScene.oneTime){
            if(pre != rand){
                if(rand == 1){
                    rain.Play();
                    thunder.Play();
                }
                else{
                    rain.Stop();
                    thunder.Stop();
                }
            }
        }
    }
```

F. 게임 시작/재시작 UI

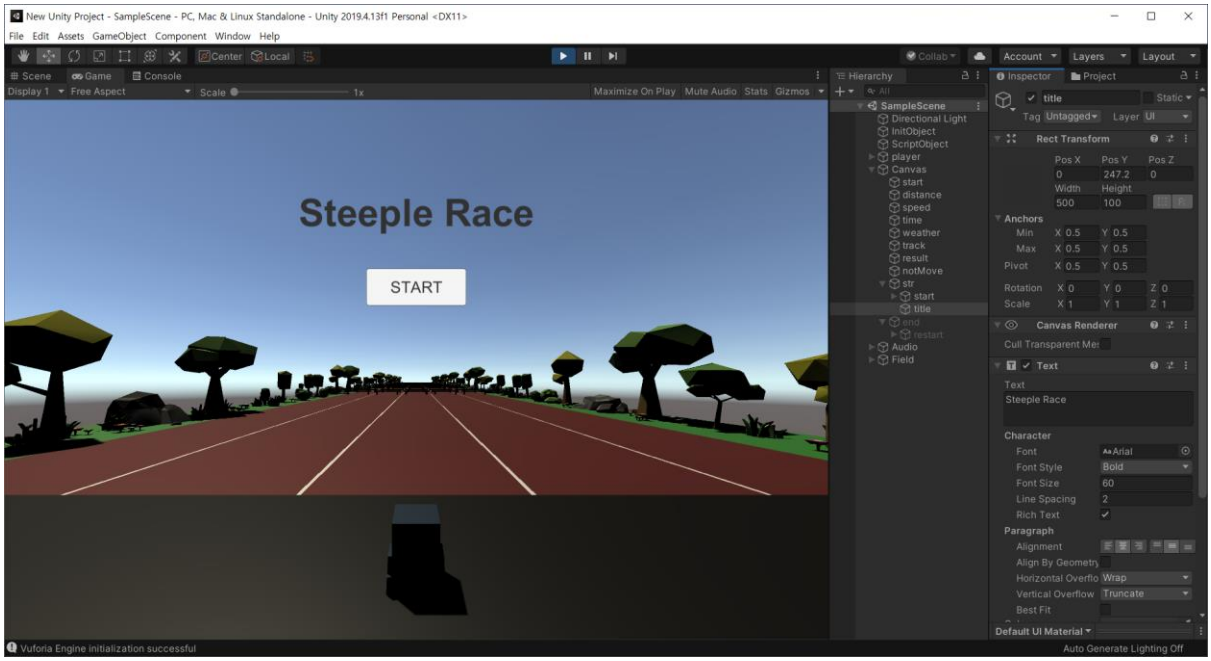


Figure 29 게임 시작 화면 – 시작 버튼

시작 버튼을 누르기 전까지 저 화면을 유지하다가, 시작 버튼을 누르면 그 때 카운트 다운이 시작된다.

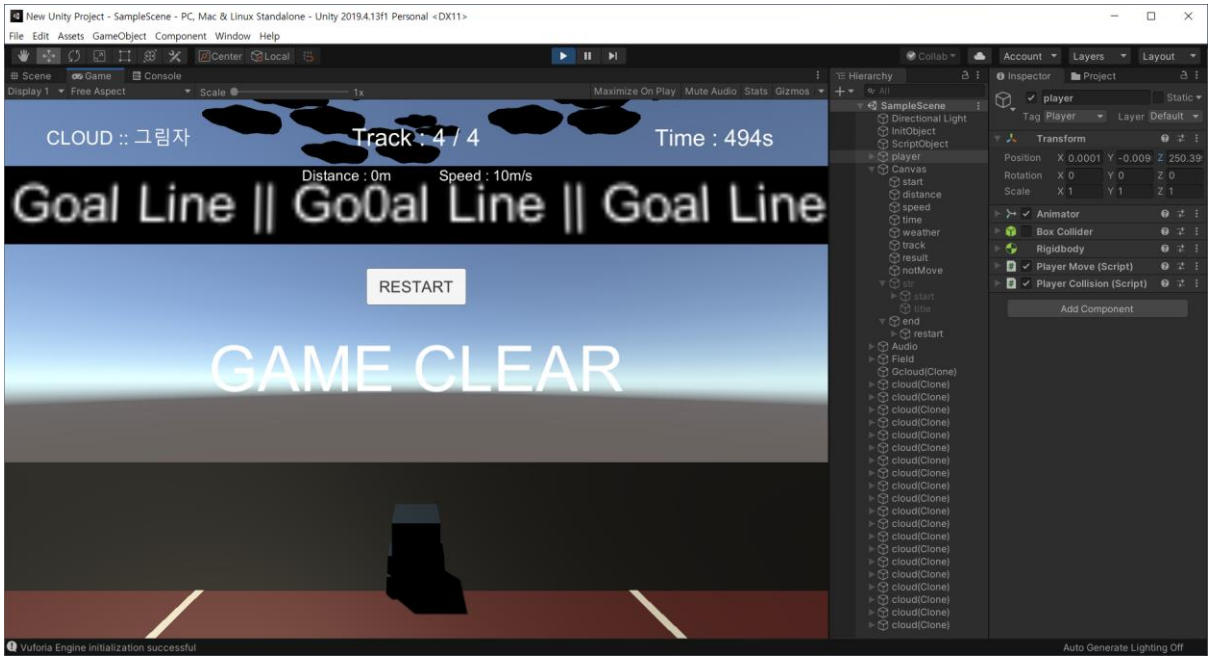


Figure 30 게임 종료 화면 – 재 시작 버튼

게임이 종료되면 재 시작 버튼이 나타나고, 버튼을 누르면 모든 설정이 초기화 되고 처음 시작 버튼이 있는 화면으로 넘어간다.

```
// Before.cs
    void Start(){
        init();
    }

    public void RestartButtonDown(){
        RestartButton = true;
    }
    public void StartButtonDown(){
        StartButton = true;
    }

    void FixedUpdate(){
        if(RestartButton){
            end.SetActive(false);
            str.SetActive(true);
            init();
            RestartButton = false;
        }
        if(StartButton){
            str.SetActive(false);
            gameStart();
            StartButton = false;
        }
    }

    void gameStart(){
        script.GetComponent<start>().enabled = true;

        // player setting
        player.transform.position = new Vector3(0, 3, -255);
        player.GetComponent<PlayerCollision>().enabled = true;

        // UI setting
        text[0].text = "Distance : 2000m";
        text[1].text = "Speed : 10m/s";
        text[2].text = "Time : 500s";
        text[3].text = "WEATHER EFFECT";
        text[4].text = "Track : 1 / 4";

        // start.cs setting
        PlayerCollision.notMove = 7;
        start.timer = 0;

        // weather setting
        changeWeather.start = true;
    }
```

```
void init(){
    // scriptObject setting
    script.GetComponent<start>().enabled = false;
    script.GetComponent<InitScene>().enabled = false;
    script.GetComponent<changeWeather>().enabled = false;

    // bgm setting
    bgm.loop = true;
    bgm.Stop();

    // InitScence.cs setting
    InitScene.oneTime = true;
    InitScene.limit_time = 500;
    InitScene.distance = InitScene.trackLength * 4;
    InitScene.timerCount = 0;

    // UI setting
    for(int i = 0; i < 8; i++)
        text[i].text = "";
    text[7].enabled = true;

    // player setting
    player.GetComponent<PlayerCollision>().enabled = false;
    player.transform.position = new Vector3(0, 0, -255);
    sp.SetActive(false);

    // player move setting
    PlayerMove.speed = PlayerCollision.initSpeed = 10;
    PlayerMove.jump = PlayerCollision.initJump = 80f;
    PlayerMove.isJump = false;
    PlayerMove.move1 = PlayerMove.move2 = false;

    // track setting
    track1.SetActive(true);
    track2.SetActive(false);
    track3.SetActive(false);
    track4.SetActive(false);
    portalA.SetActive(true);

    // grass setting
    grass[0].SetActive(true);
    grass[1].SetActive(true);
    for(int i = 2; i < 6; i++)
        grass[i].SetActive(false);

    // weather setting
    GameObject ob;
    if(ob = GameObject.FindWithTag("Gsun"))
```

```
            Destroy(ob);
        if(ob = GameObject.FindWithTag("Gcloud"))
            Destroy(ob);
        if(ob = GameObject.FindWithTag("Grain"))
            Destroy(ob);
        if(ob = GameObject.FindWithTag("Gsnow"))
            Destroy(ob);

        // obstacle-stone setting
        createStone.one = true;
    }
}
```