프로젝트 중간 보고서

-장애물 경주-

컴퓨터과학전공 1814965 김현주

0. 목차

1. 프로젝트 주제: <장애물 경주> 게임 제작

2. 프로젝트 진행 단계

   1) 프로젝트 계획서 내용을 토대로 구현할 기능 정의 (완료)

   2) 각 기능을 수행할 object와 코드 디자인 (완료)

   3) Object 간의 관계와 각 object에 적용될 property, script 매칭 (완료)

   4) 각 기능 개발 + 기능별 관계에서의 에러 수정 (진행 중)

3. 구현한 내용과 구현할 내용

   A. 구현한 내용

      i. 플레이어 모델과 움직임

      ii. 플레이어의 포탈을 통한 트랙 이동

iii. 각각 날씨에 따라 생성될 asset(구름, 비, 눈, 빙판 등)

iv. 일정 시간마다 날씨 자동 변환 + 플레이어에게 미치는 효과

B. 구현할 내용

i. track3의 판의 아래에서 사용될 포탈

ii. Track4 이후 goal line

iii. 장애물 stick, hurdle 모델, 생성

iv. 장애물 stone 생성, 소멸 코드

v. stick 움직임 코드

vi. game start, timer, distance, result 모델과 코드

vii. 음향 모델 추가, 적용

4. 프로젝트의 요구 기능

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.

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

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 3의 아래에 위치하여 player가 board에서 떨어지면 이 포탈을 통해 track 3의 처음 시작 위치(=portal-B)로 이동한다.

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

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

F. Collider

i. Player & hurdle: 부딪히는 소리가 나며 허들은 쓰러지고 player는 n초간 움직이지 못한다.

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

iii. Track & stone: 부딪히는 소리가 나며 돌멩이가 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.

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

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

iv. Game end: 제한 시간 내에 end point에 도착하면 성공 메시지를, 그렇지 못하면 실패 메시지를 출력하고, 그 이후에 player는 움직일 수 없다..

5. GameObject 구조와 Scripts :: ~~현재 구현한 내용~~ 표시

A. Hierarchy

~~i. Directional Light~~

ii. ~~ScriptObject: EmptyObject~~. MainScript.cs, ~~changeWeather,~~ Start.cs

~~iii. player: rigidbody, rotation freeze, PlayerMove.cs, PlayerCollision.cs~~

~~1. Main Camera~~

2. time: 3D text

3. distance: 3D text

4. start: 3D text. Active >> inactive

5. result1: Resources.

6. result2: Resources.

iv. ~~track 1: Active >> inactive~~. createStone.cs

~~1. track: prefab~~

2. hurdle(n): prefab, rigidbody

~~v. track 2: Inactive >> active~~

~~1. track: prefab~~

2. stone: Resources

~~vi. track 3: Inactive >> active~~

~~1. board~~

2. portal-C

~~vii. track 4: Inactive >> active~~

~~1. track: prefab~~

2. stick(n): prefab, rigidbody, moveStick.cs

3. end Point

A. GoalLine: rigidbody

~~viii. portal A. Active >> inactive~~

~~ix. portal B~~

~~x. Gsun: Resources~~

~~1. heat_haze: particle~~

~~xi. Gcloud: Resources~~

~~1. cloud: Resources~~

~~xii. Grain: Resources~~

~~1. darkCloud: Resources~~

~~2. rain: particle~~

~~xiii. Gsnow: Resources~~

~~1. darkCloud: Resources~~

~~2. snow: particle~~

~~3. snow_set: Resources~~

~~4. ice: Resources~~

B. Resources

i. result1: 3D text, "Game success"

ii. result2: 3D text, "Game Failed"

iii. ~~stone: prefab~~, rigidbody, deleteStone.cs

~~iv. portalEffect: Particle System~~

~~v. weather/Gsun: EmptyObject~~

~~vi. weather/Gcloud: EmptyObject, createCloud.cs~~

~~vii. weather/Grain: EmptyObject, createRain.cs~~

~~viii. weather/Gsnow: EmptyObject, createSnow.cs~~

~~ix. cloud: prefab, deleteCloud.cs, moveCloud.cs~~

~~x. darkCloud: prefab, deleteRain.cs, moveCloud.cs~~

~~xi. snow_set: prefab, deleteSnow.cs~~

~~xii. ice: prefab, deleteSnow.cs~~

C. Scripts

~~i. PlayerMove.cs: Player 이동~~

ii. MainScript.cs: timer/distance 출력, game result 로드

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

iv. createStone.cs: stone 자동 생성

v. deleteStone.cs: stone 소멸, sound

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

6. 구현 내용 실행 화면

A. 처음 시작 화면

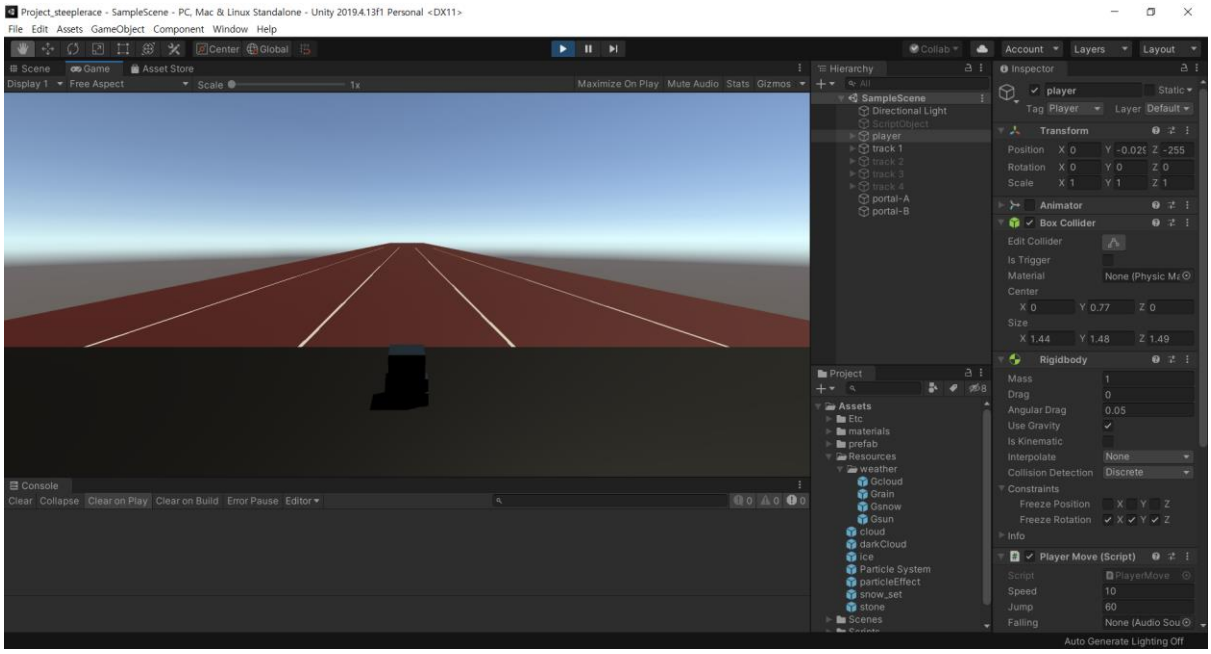

Figure 1 시작 화면

B. 플레이어 움직임

i. 플레이어는 위/왼/오른쪽 화살표로 +z, -x, +x 축으로 이동할 수 있다.

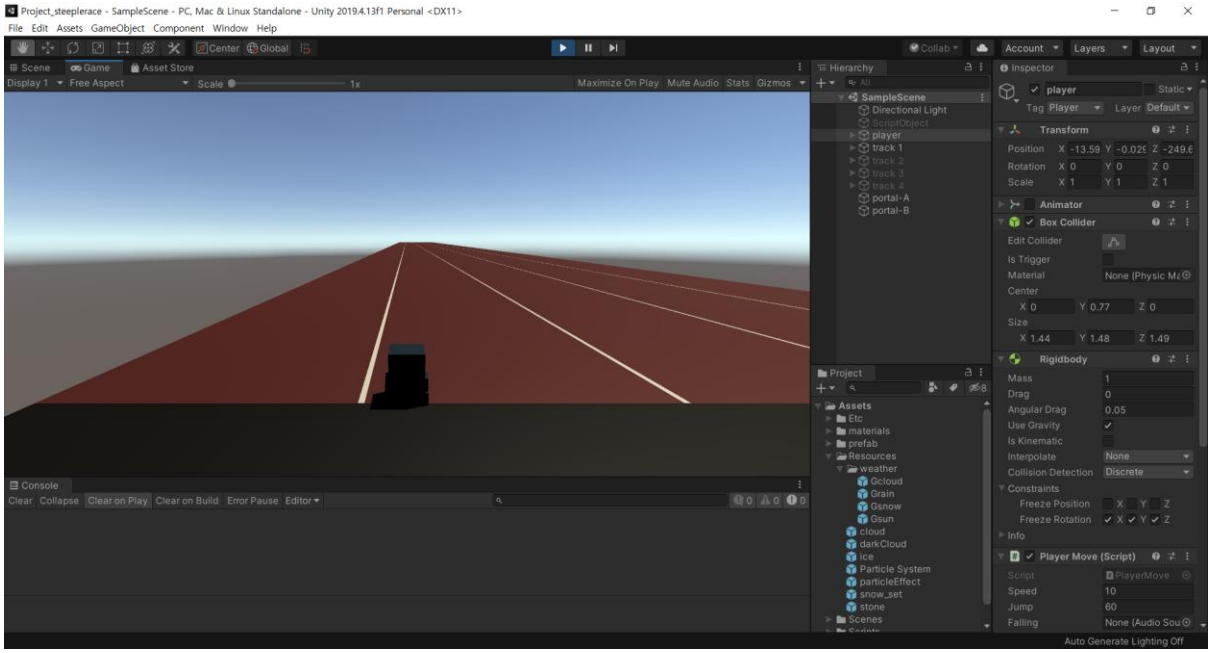

Figure 2 플레이어 좌우, 앞 이동

```
// PlayerMove.cs
if(Input.GetKey(KeyCode.UpArrow))
	transform.Translate(Vector3.forward * speed * Time.deltaTime);
if(Input.GetKey(KeyCode.LeftArrow))
	transform.Translate(Vector3.left * speed * Time.deltaTime);

if(Input.GetKey(KeyCode.RightArrow))
	transform.Translate(Vector3.right * speed * Time.deltaTime);
```

ii. Space로 점프할 수 있다.

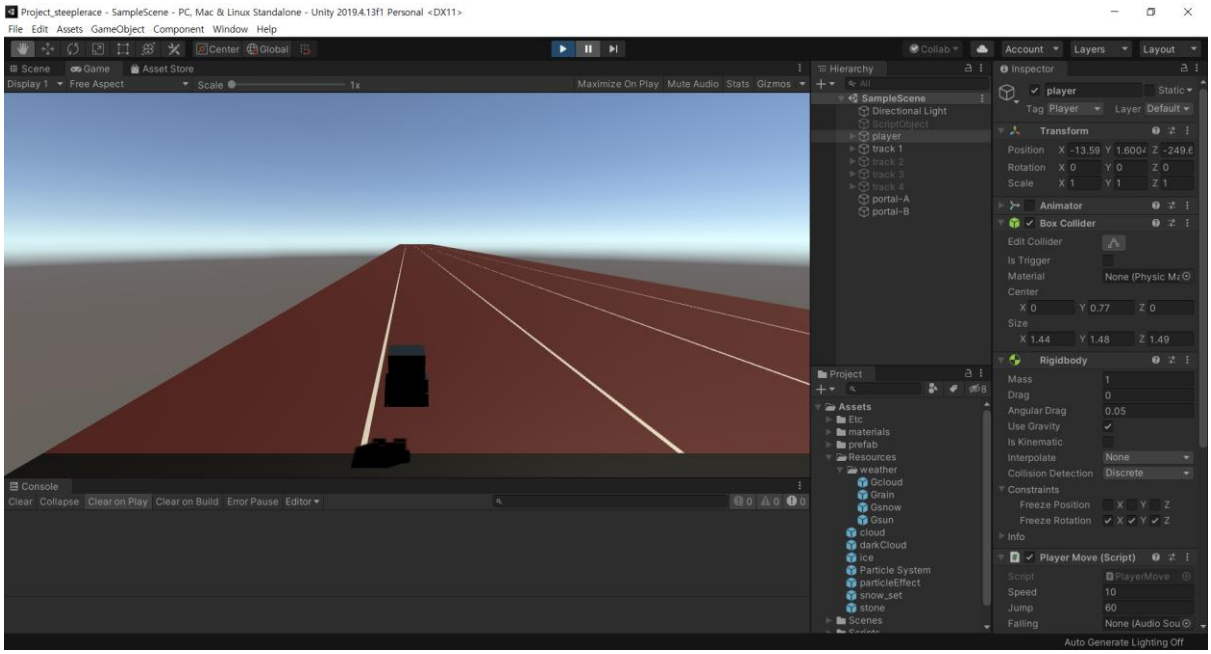

Figure 3 플레이어 점프

```
// PlayerMove.cs
if(Input.GetKey(KeyCode.Space))
	GetComponent<Rigidbody>().AddForce(Vector3.up * jump);
```

C. 각 트랙, 트랙 간 이동

i. Track1, Track2, Track4는 장애물을 제외하고는 똑같다.

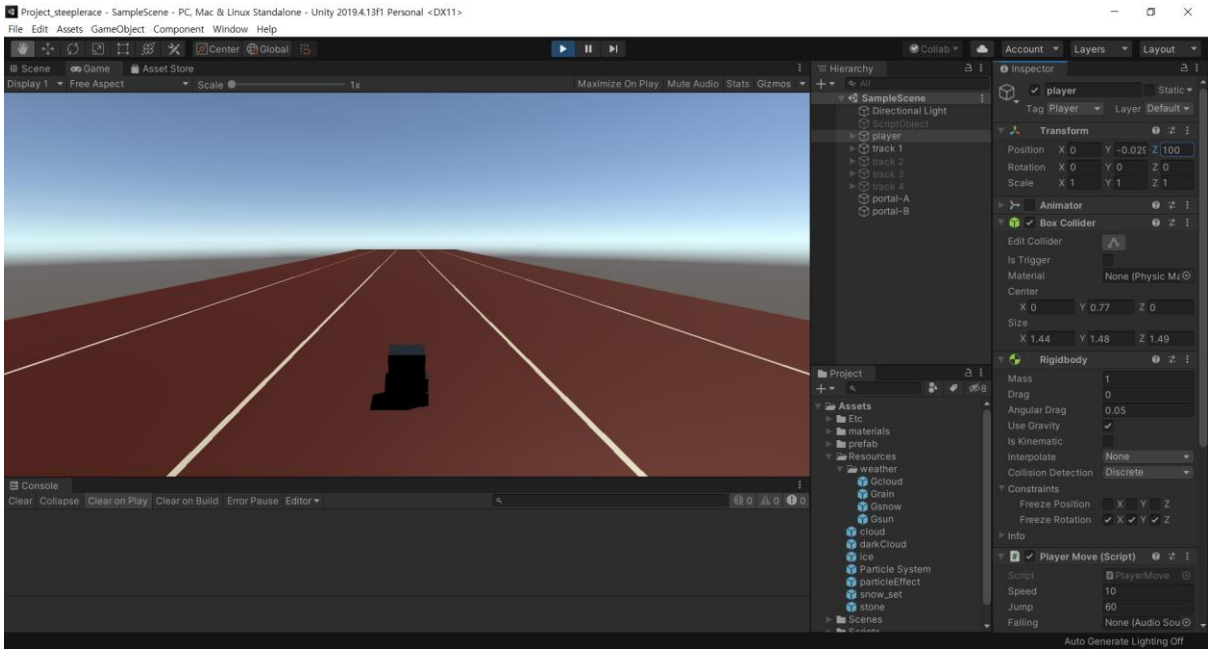

Figure 4 Track1, Track2, Track4

ii. Track3은 긴 판이 땅을 대신한다. 판 위에서 플레이어의 이동속도와 점프 높이가 낮아지고, x 좌표가 임의로 변한다.

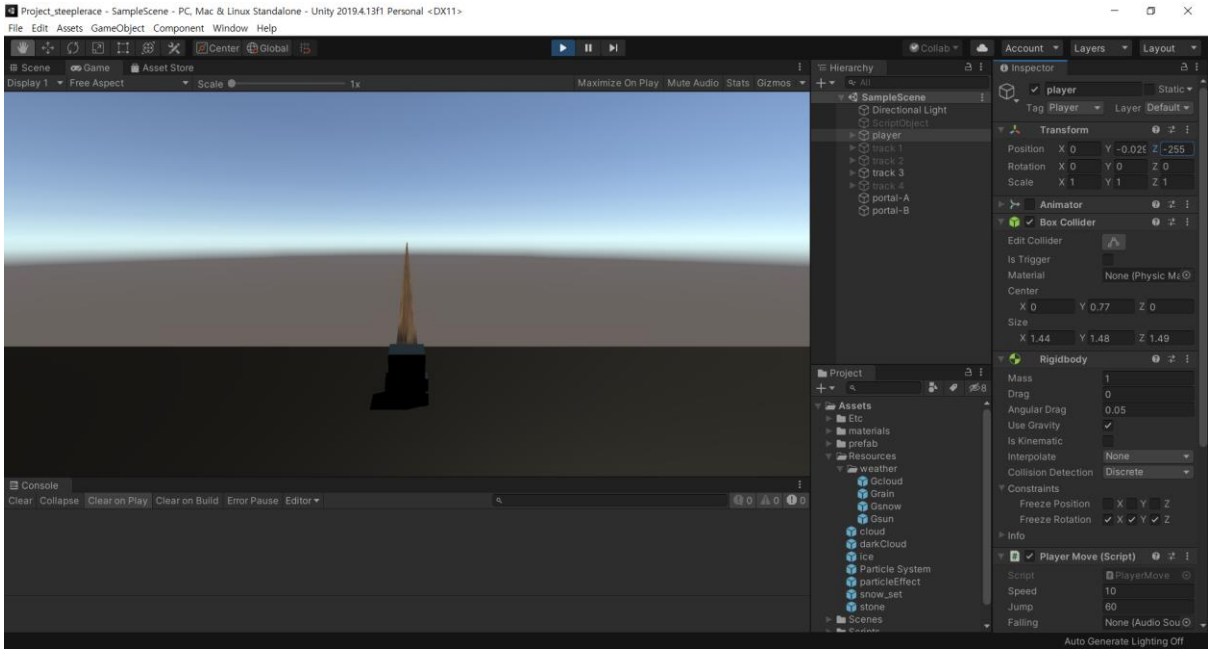

Figure 5 Track3

```
// PlayerCollision.cs
void OnCollisionEnter(Collision other){
        if(other.gameObject.name == "board"){
            GetComponent<PlayerMove>().jump = 30f;
            GetComponent<PlayerMove>().speed = initSpeed - 3;

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

iii. 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 그 track을 비활성화 시키고, 다음 track을 활성화 시킨다.

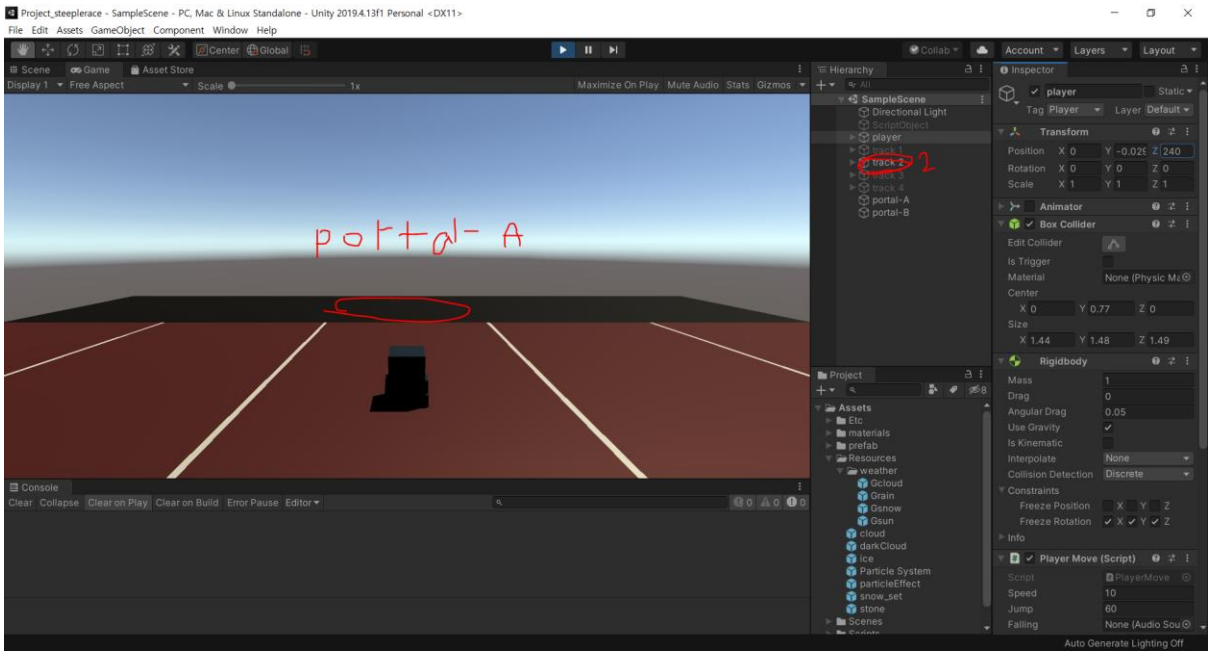

Figure 6 Track1, Track2, Track3의 끝 부분

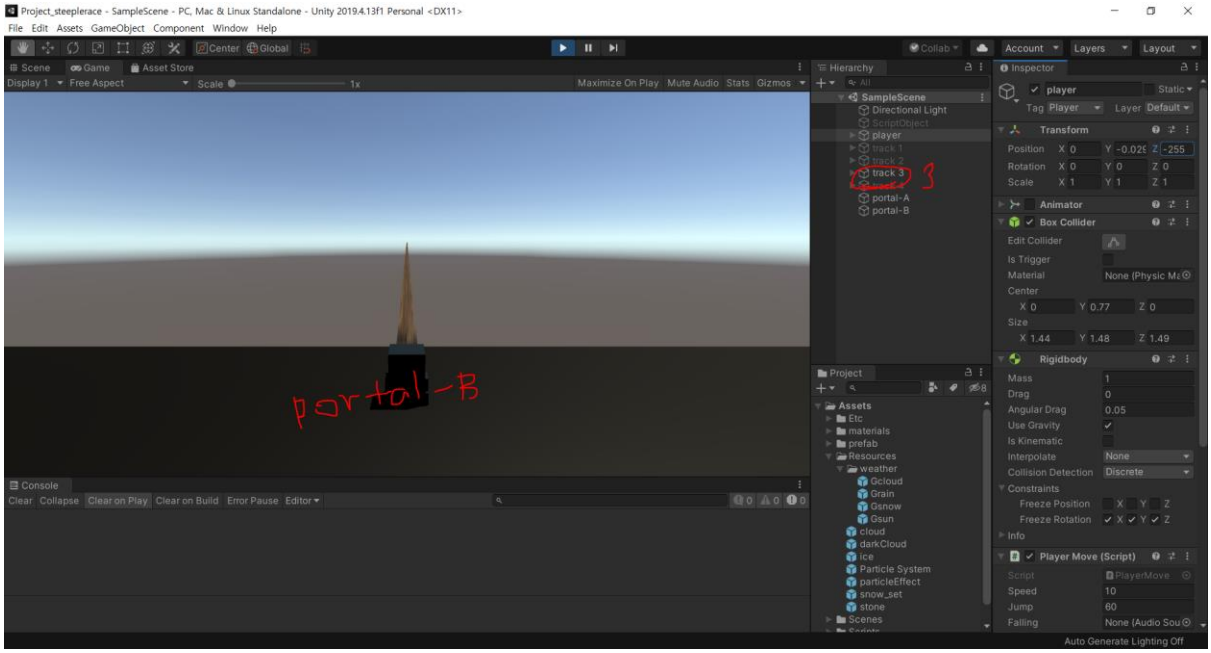

Figure 7 Track2, Track3, Track4의 시작 부분

```
// PlayerCollision.cs
void OnCollisionEnter(Collision other){
        . . .
        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>().positio
n;
                //portal.Play();
        }
        . . .
}
```

D. 각 날씨 효과

i. Sun: 지열로 인한 아지랑이가 생성되고 플레이어는 15초 이상 달리면 3초간 움직이

지 못한다. 3초 후 다시 움직일 수 있는 것은 MainScript.cs에서 구현할 내용이라 아직은 다시 움직일 수 없다.

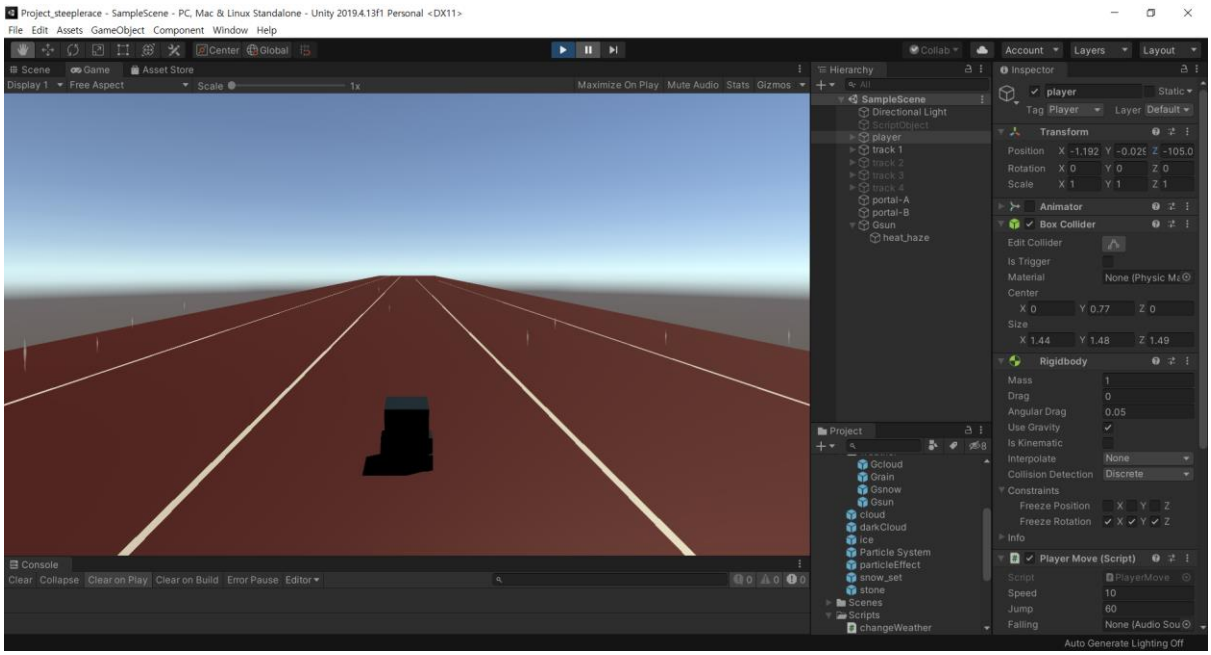

Figure 8 sun – 아지랑이

```
// PlayerMove.cs
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(sun > 15){
	gameObject.GetComponent<PlayerCollision>().notMove = 3;
	sun = 0;
}
```

ii. Cloud: 구름이 생성되고, 플레이어에게는 제약이 없다.

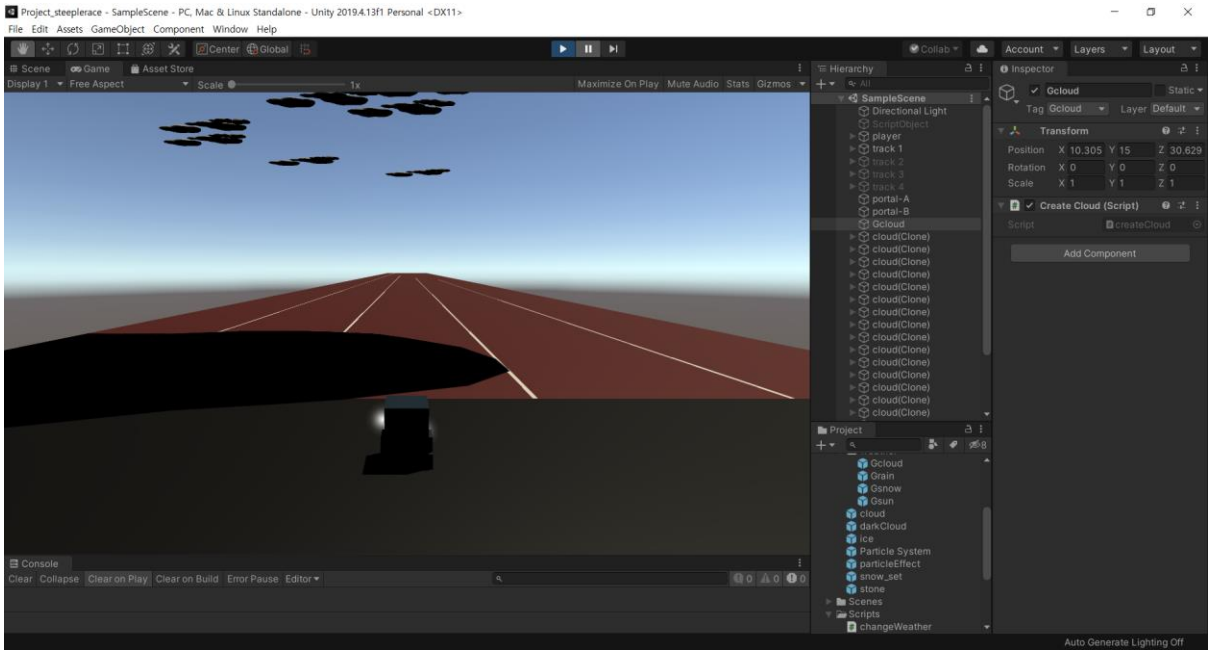

Figure 9 cloud - 구름

```
// createCloud.cs
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.ident
ity);
}

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

iii. Rain: 먹구름과 비가 생성되고, 플레이어의 이동 속도가 감소한다.

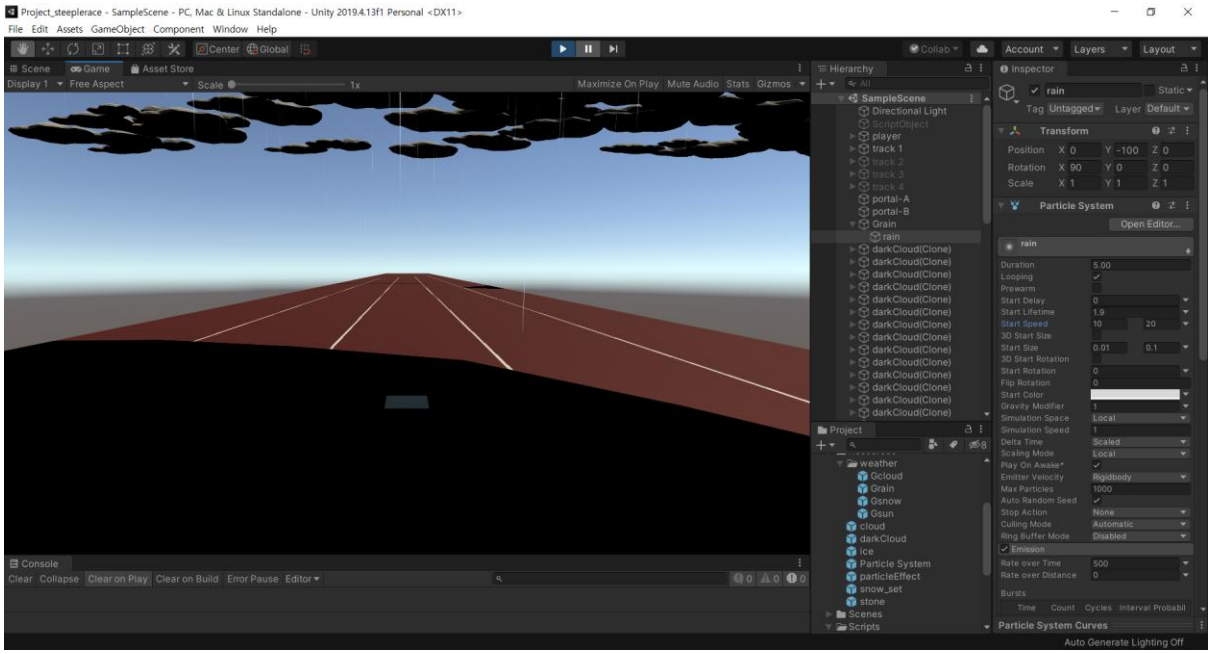

Figure 10 rain - 먹구름, 비

```
// createRain.cs
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.i
dentity);
}

// deleteRain.cs
if(!GameObject.FindWithTag("Gcloud") && !GameObject.FindWithTag("Gsnow"))

	Destroy(gameObject);

// changeWeather.cs
player.GetComponent<PlayerMove>().speed -= 4;
player.GetComponent<PlayerCollision>().initSpeed -= 4;
```

iv. Snow: 먹구름과 눈, 바닥에 빙판과 눈덩이가 생성된다. 빙판 위에서 플레이어의 이동속도와 점프 높이가 낮아지고, 눈덩이 위에서 플레이어의 이동속도가 낮아지고 x좌표가 임의로 변한다.

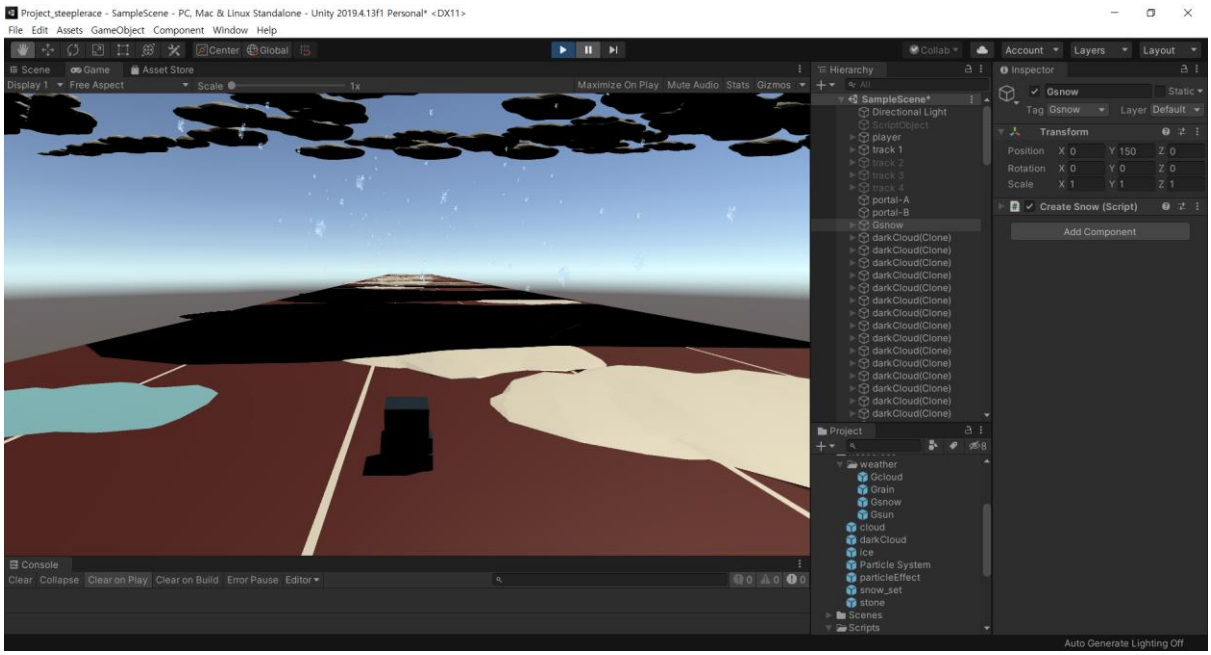

Figure 11 snow - 먹구름, 눈, 빙판, 눈덩이

```
// createSnow.cs
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);
}

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);
}
```

```
// PlayerCollision.cs
if(other.gameObject.tag == "ice"){
        GetComponent<PlayerMove>().jump = 30f;
        GetComponent<PlayerMove>().speed = initSpeed - 4;
}
else if(other.gameObject.tag == "snow_set"){
        GetComponent<PlayerMove>().speed = initSpeed - 4;
        transform.position = new Vector3(transform.position.x + Random.Range(-
5, 5) * 0.01f, transform.position.y, transform.position.z);
}
```

E. 날씨 자동 변환

i. 처음 시작하고 1초 뒤부터 30초 간격으로 날씨가 자동으로 변환된다.

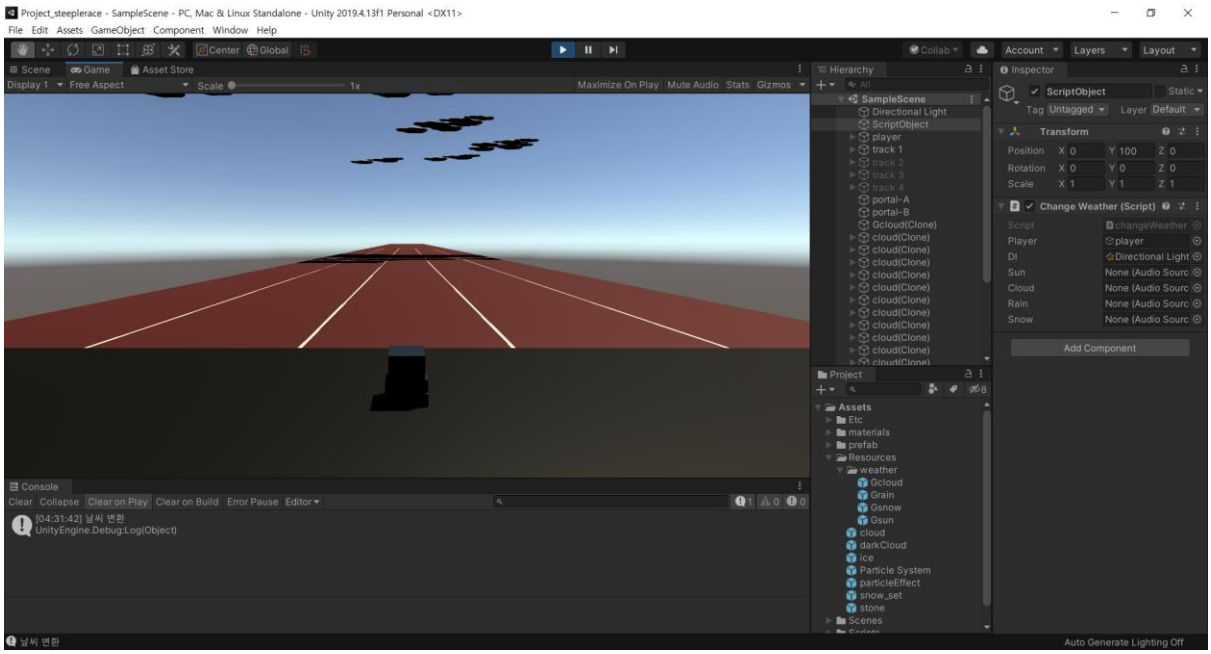

Figure 12 날씨 변환 - 처음 시작: cloud

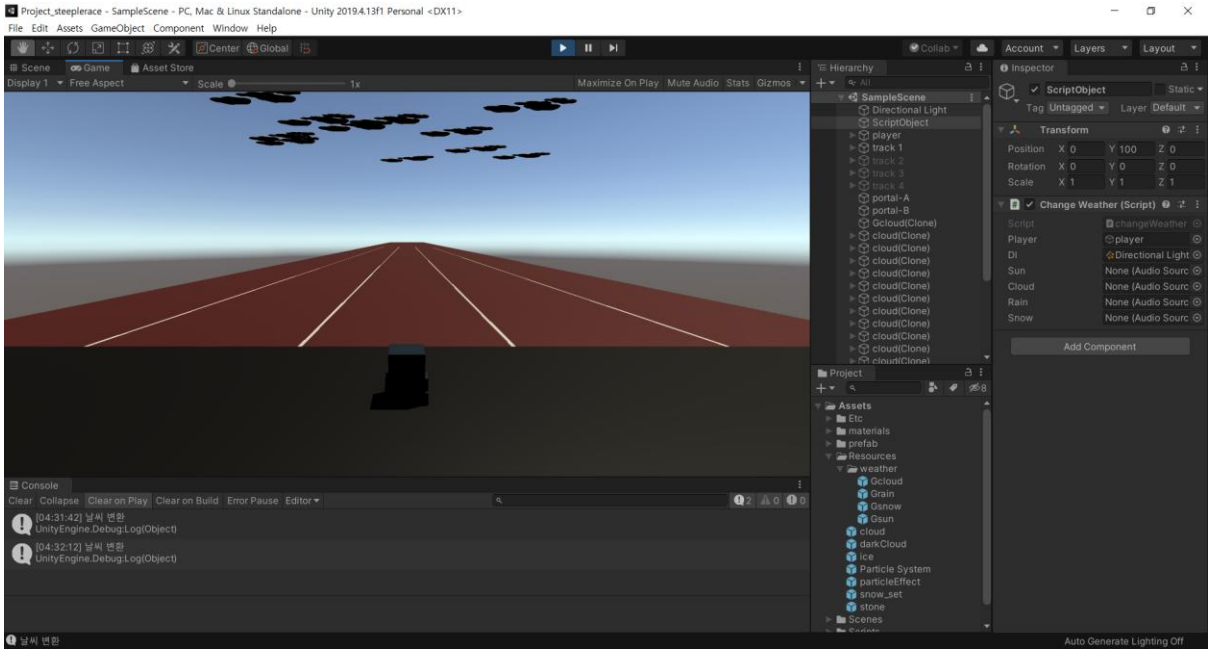

Figure 13 날씨 변환 - 두번째: cloud

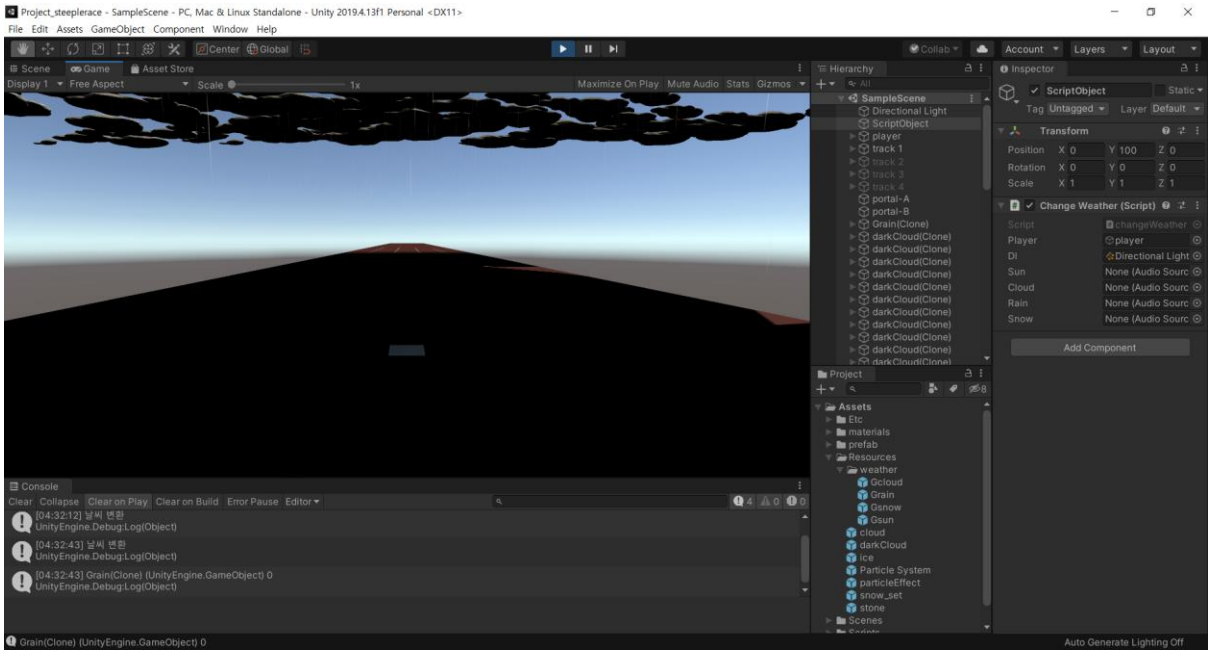

Figure 14 날씨 변환 - 세번째: rain

```
// changeWeather.cs
void Start(){
        . . .
       InvokeRepeating("change", 1f, 30f);
        InvokeRepeating("delete", 30.1f, 30f);
    }
void change(){
        Debug.Log("날씨 변환");
        if(GameObject.FindWithTag("result") == null){
```

```
            if(pre != rand)
                switch(rand){
                    case 3: // Gsun
                        dl.GetComponent<Light>().intensity = 1.2f;
                        instance1 = (GameObject) Instantiate(instance[rand] as
 GameObject, new Vector3(0, 0, 0), Quaternion.identity);
                        //sun.Play();
                        break;
                    case 0: // Gcloud
                        dl.GetComponent<Light>().intensity = 1f;
                        instance1 = (GameObject) Instantiate(instance[rand] as
 GameObject, pos, Quaternion.identity);
                        //cloud.Play();
                        break;
                    case 1: // Grain
                        dl.GetComponent<Light>().intensity = 0.9f;
                        instance1 = (GameObject) Instantiate(instance[rand] as
 GameObject, pos, Quaternion.identity);
                        player.GetComponent<PlayerMove>().speed -= 4;
                        player.GetComponent<PlayerMove>().initSpeed -= 4;
                        //rain.Play();
                        break;
                    case 2: // Gsnow
                        dl.GetComponent<Light>().intensity = 0.9f;
                        instance1 = (GameObject) Instantiate(instance[rand] as
 GameObject, pos, Quaternion.identity);
                        //snow.Play();
                        break;
                }
        }
}

void delete(){
        if(GameObject.FindWithTag("result") == null){
            pre = rand;
            rand = Random.Range(0,4);
            if(pre != rand){
                Destroy(instance1);
                if(pre == 1){
                    player.GetComponent<PlayerMove>().speed += 4;
                    player.GetComponent<PlayerMove>().initSpeed += 4;
                }
            }
        }
}
```