

PORTFOLIO

김일환(GIM IL HWAN)

Email : dodjungvv@naver.com

Phone Number : 010 - 4142 - 1442

2. CubeBreaker



GIMILHWAN

1. CubeBreaker 소개

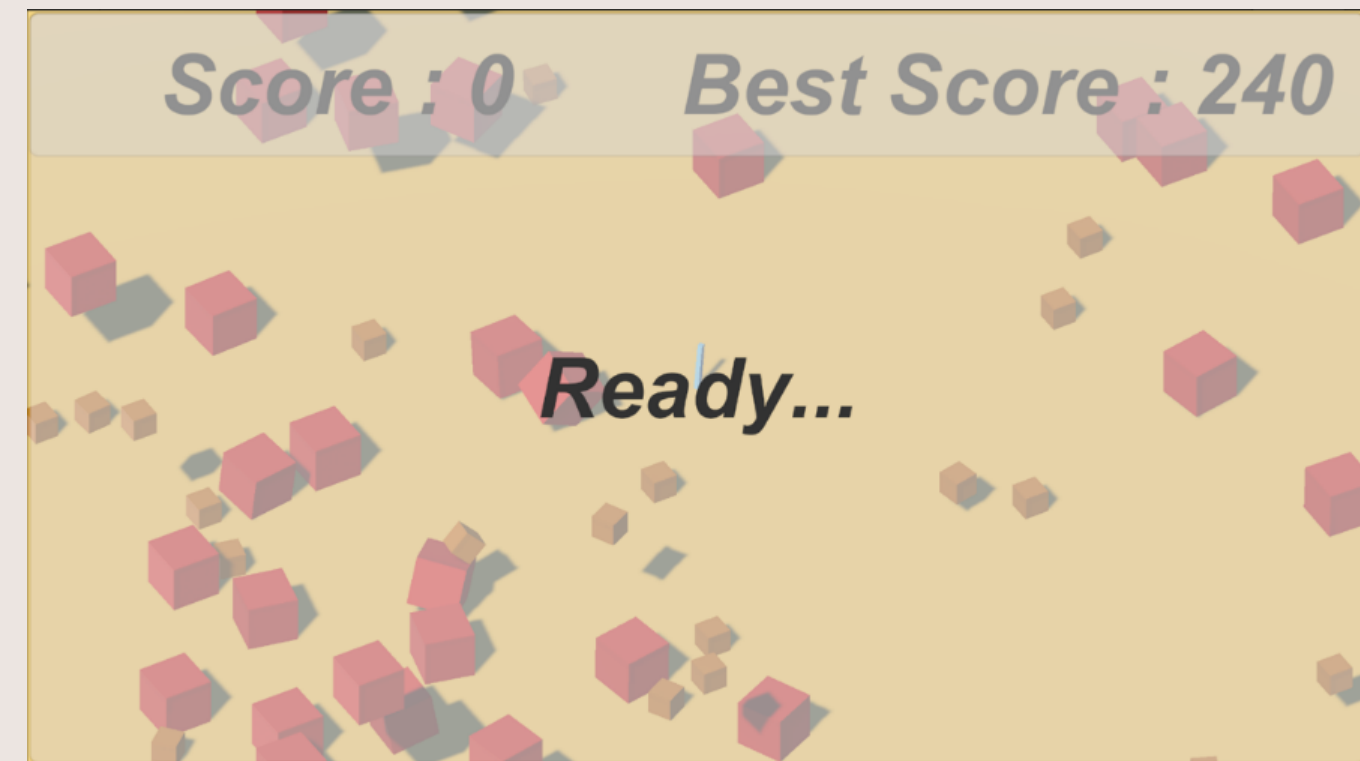
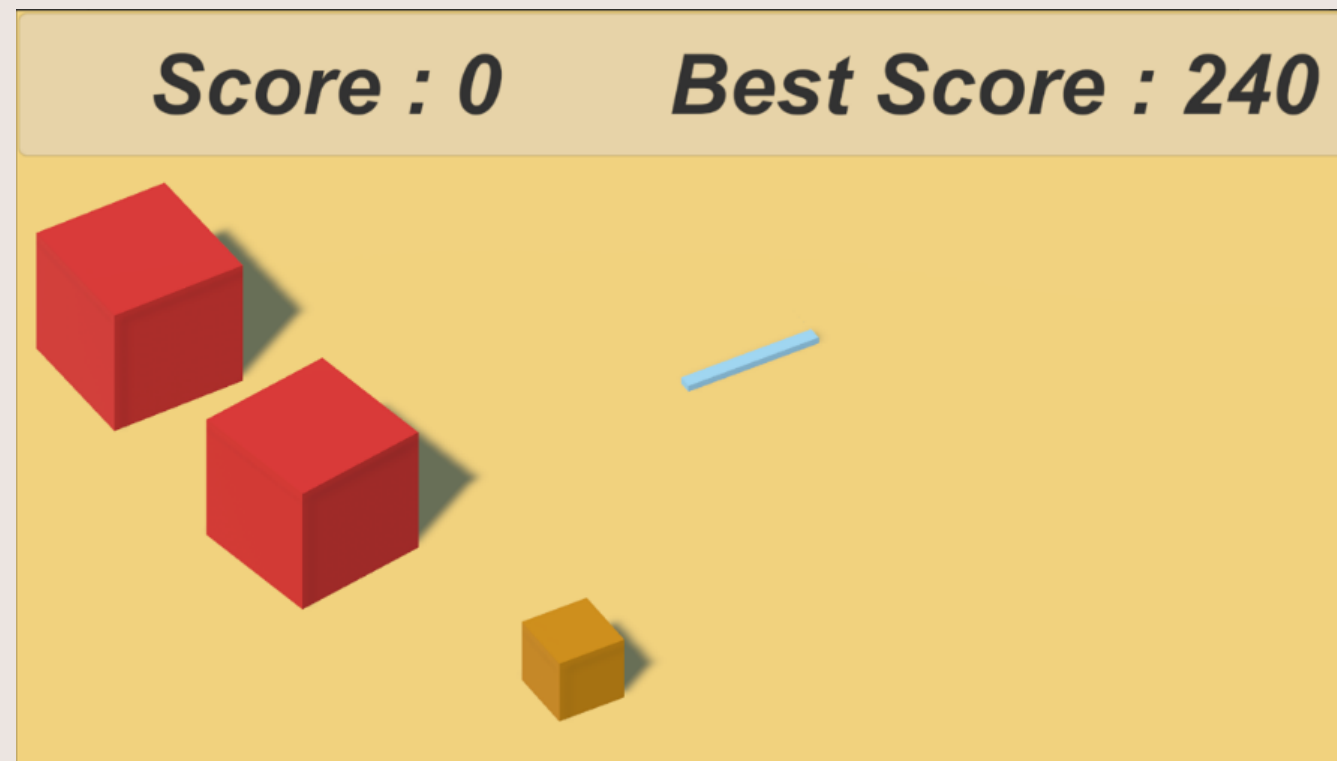
2. Game 화면및 기능 소개

3. 중요 Point및 소스

4. 차후 계획

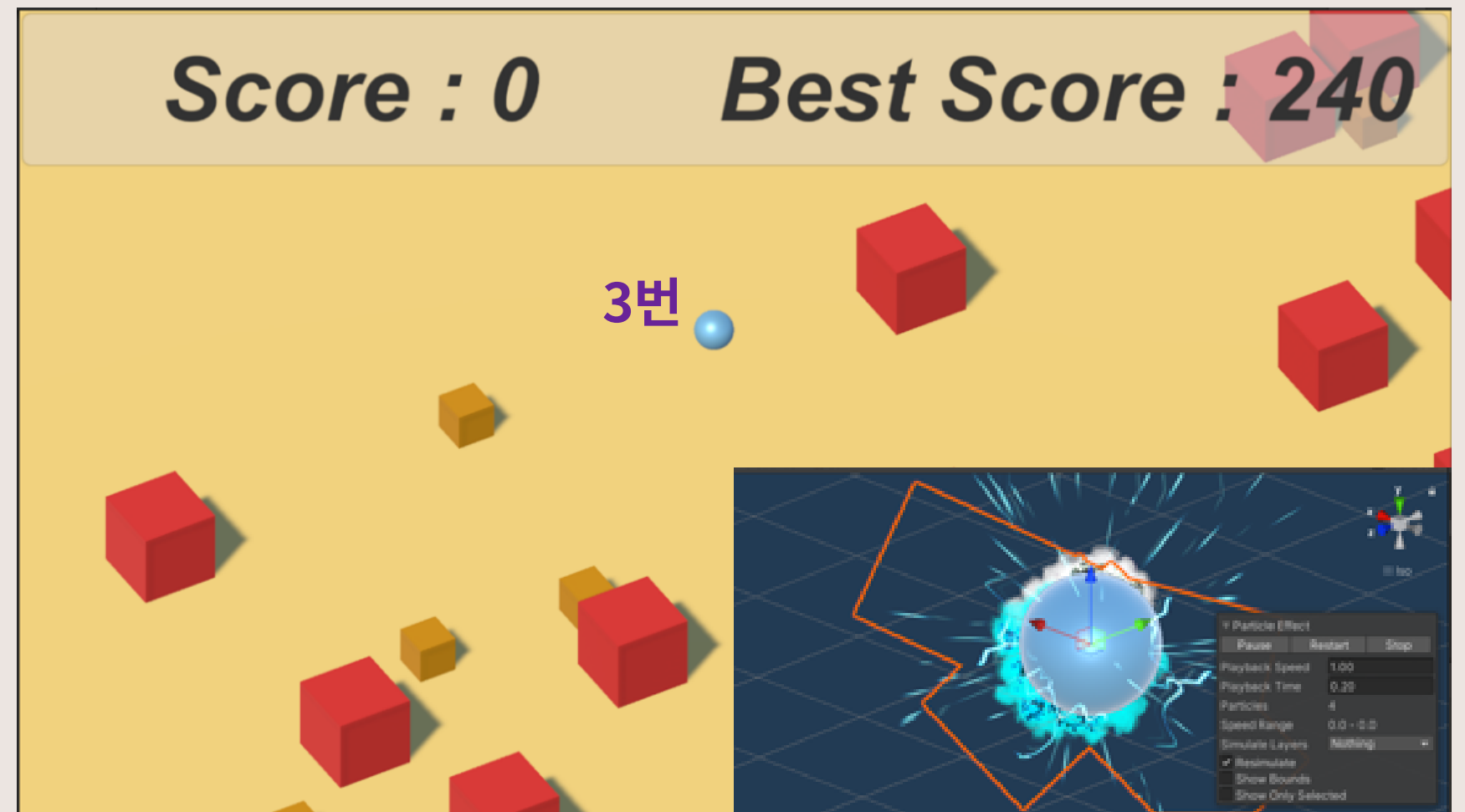
1. CubeBreaker

프로젝트에 대한 간단 설명 및 주요 Point



1. 중앙에 파랑색 포신을 회전시킨 후 파랑 폭탄을 발사 할 수 있다.
2. 발사한 폭탄이 지면에 떨어지거나 박스에 충돌하면 주변에 박스가 파괴 된다.
3. 부서진 박스만큼 Score를 획득 한다.

1. Game Control



1.
 - 키보드의 왼쪽 ctrl 키 또는 마우스 좌클릭 을 사용하여 Y축, X축 기준으로 각각 1회 회전 한다
2.
 - 중앙에 포신을 2회 회전 후 다시 Ctrl키 또는 마우스 좌클릭을 홀드해서 파워(Slider)를 증가 시킬 수 있다.
 - 일정 파워에서 홀드를 해제 하면 포신에서 파랑 폭탄이 발사 된 다.
3.
 - 폭탄은 포신의 방향으로 발사 후 카메라는 폭탄을 따라간 후 폭탄이 충돌하면 임팩트와 함께 살아진다.

2. Game Panel



1.
- 게임 첫 시작시 또는 게임이 다시 시작 될 때 Control을 더이상 할 수 없도록 하고 Ready Text를 통해 시작을 알려 준 후 Score 를 '0' 으로 초기화 한다.
2.
- 게임이 끝난 직 후 화면에 현재 Score및 Best Score를 보여 준다
3.
- 게임이 끝난 상황을 알려주기 위해 화면에 Text를 'Wait For Next Round' 로 변경하여 다음 게임을 준비 중인 것을 알려 준다.

1. GameManager

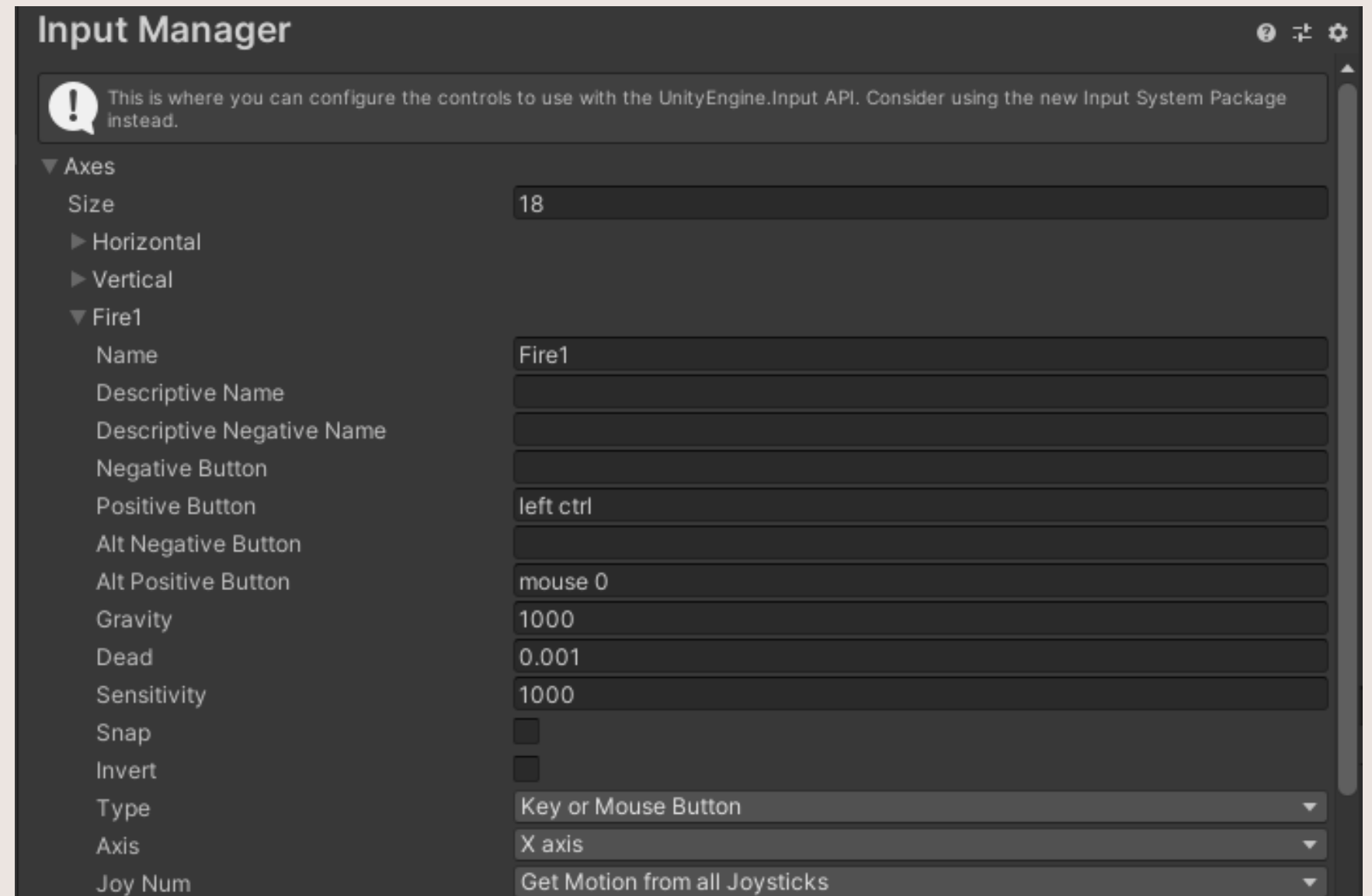
```
77 public void Reset()
78 {
79     score = 0;
80     UpdateUI();
81
82     //라운드 다시 시작을 위한 코드
83     StartCoroutine("RoundRoutine");
84 }
85
86 IEnumerator RoundRoutine() //상황에 마제 Back 단에서 처리
87 {
88     //Ready
89     onReset.Invoke();
90
91     readyPannel.SetActive(true);
92     cam.SetTarget(ShooterRotator.transform, CamFollow.State.Idle);
93     ShooterRotator.enabled = false;
94
95     isRoundActive = false;
96
97     messageText.text = "Ready...";
98
99     yield return new WaitForSeconds(3f);
```

```
100
101     //Play
102     isRoundActive = true;
103     readyPannel.SetActive(false);
104     ShooterRotator.enabled = true;
105
106     cam.SetTarget(ShooterRotator.transform, CamFollow.State.Ready);
107
108     while(isRoundActive == true)
109     {
110         yield return null;
111     }
112
113     //End
114     readyPannel.SetActive(true);
115     ShooterRotator.enabled = false;
116
117     messageText.text = "Wait For Next Round...";
118
119     yield return new WaitForSeconds(3f);
120     Reset();
121 }
```

- 77 라인 Reset() 함수를 사용하여 게임의 상태를 초기화 시킨다. (현재 Score를 '0' 으로 지정 후 Panel에 UI 를 Update 시켜 준다.)
- 91라인에서 readyPannel을 활성화 후 92 라인에서 카메라의 위치를 ShooterRotator로 지정 후 현 상태를 Idle 대기 상태로 변경 및 93 라인에서 ShooterRotator를 사용할 수 없도록 한다.
- 99 라인에서 대기시간 3초를 부여 후 102 ~ 104 라인에서 Game이 진행 가능한 상태로 변경 한다.
- 114 ~ 120 에서 다시 Panel의 정보및 ShooterRotator의 상태를 변경 후 3초 대기시간 뒤 게임의 화면을 Reset() 함수를 사용하여 게임 상태를 초기화 한다.

2. ShooterRotator

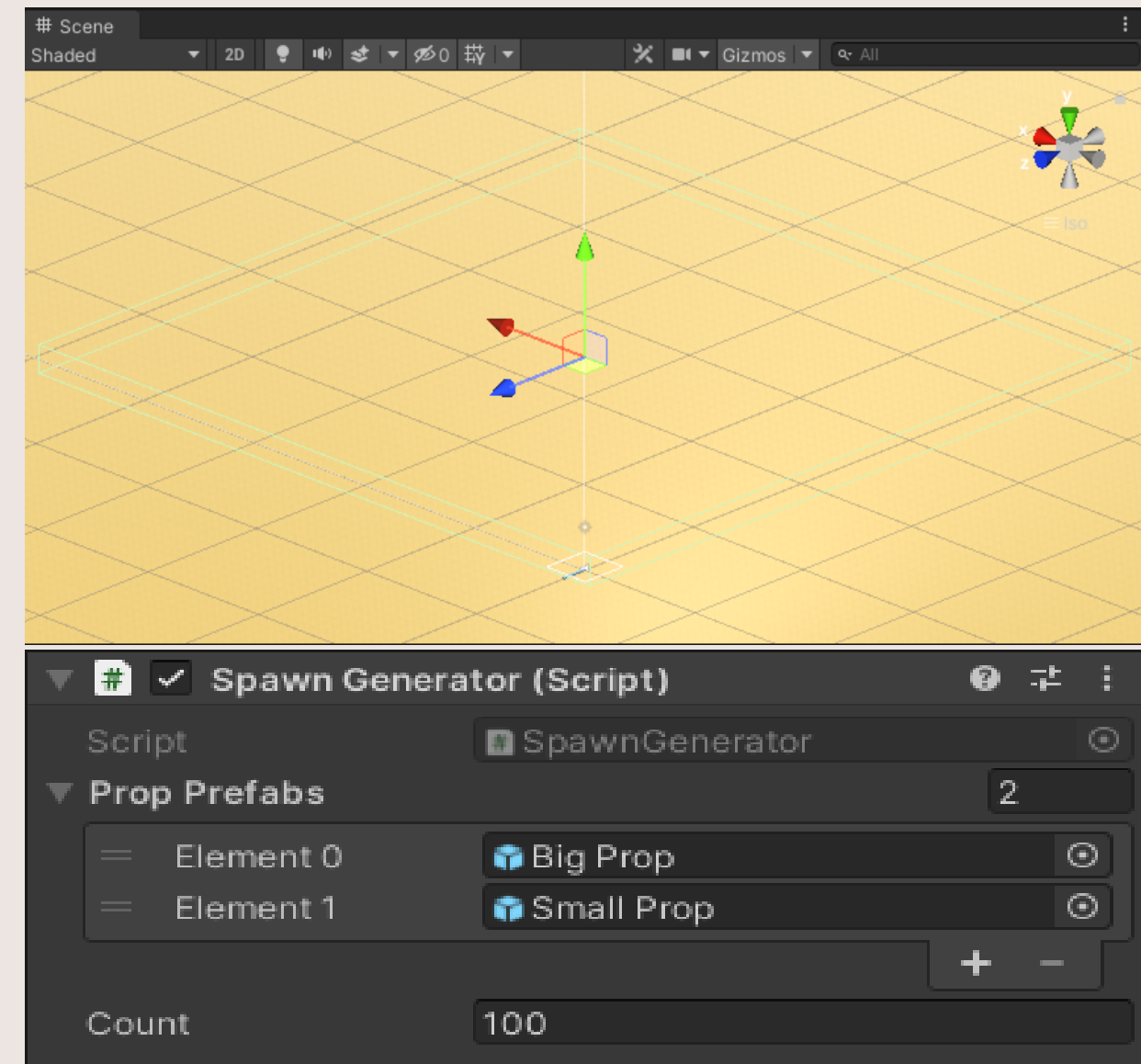
```
22 void Update()
23 {
24     switch(state){
25     case RotateState.Idle:
26         if (Input.GetButtonDown("Fire1"))
27         {
28             state = RotateState.Horizontal;
29         }
30         break;
31
32     case RotateState.Horizontal:
33         if (Input.GetButton("Fire1"))
34         {
35             transform.Rotate(new Vector3(0, horizontalRotateSpeed * Time.deltaTime, 0));
36         }
37         else if (Input.GetButtonUp("Fire1"))
38         {
39             state = RotateState.Vertical;
40         }
41         break;
42
43     case RotateState.Vertical:
44         if (Input.GetButton("Fire1"))
45         {
46             transform.Rotate(new Vector3(-verticalRotateSpeed * Time.deltaTime, 0, 0));
47         }
48         else if (Input.GetButtonUp("Fire1"))
49         {
50             state = RotateState.Ready;
51             ballShooter.enabled = true;
52         }
53         break;
54     }
55 }
```



- Update() 함수 에서 ShooterRotator 의 state 에 따라 움직임을 전환 한다.
- 26, 33, 37, 44, 48 라인에 Input "Fire1" 은 왼쪽 이미지에 InputManager 를 통해 마우스 왼쪽 버튼및 왼쪽 ctrl 키를 지정하여 사용함.(이를 사용하여 컨트롤러를 사용한 플레이를 구현 가능.)
- 25 라인 Idle상태 에서 Horizontal 상태로 변경하고 32 라인 Horizontal 상태에서 수평 방향으로 회전 후 vertical 상태로 변경한다. 마지막으로 43 라인에서 수직방향으로 변경 후 Ready 상태로 변경한다.

3. SpawnGenerator

```
12 public int count = 100;
13
14 private List<GameObject> props = new List<GameObject>();
15
16 // Start is called before the first frame update
17 void Start()
18 {
19     area = GetComponent<BoxCollider>();
20
21     for (int i = 0; i < count; i++)
22     {
23         Spawn();
24         //생성용 함수
25     }
26
27     area.enabled = false; //불필요한 충돌을 방지 하기 위해
28 }
29
30 private void Spawn()
31 {
32     int selection = Random.Range(0, propPrefabs.Length);
33
34     GameObject selectedPrefab = propPrefabs[selection];
35
36     Vector3 spawnPos = GetRandomPosition();
37
38     GameObject instance = Instantiate(selectedPrefab, spawnPos, Quaternion.identity);
39
40     props.Add(instance);
41 }
```



- 17 라인에 Start() 함수 실행시 왼쪽 Game화면에 BoxCollider 를 area로 등록 후 21 라인 반복문에서 count 수만큼 Spawn() 함수를 사용한다.
- Spawn() 함수의 32 ~ 36 라인에서 왼쪽 아래 이미지의 등록된 PropPrefabs에서 랜덤하게 받아 온 후 지정된 범위안에서 랜덤한 위치를 받아 온다.
- 38 라인에서 GameObject를 등록 후 40 라인에서 instance GameObject를 화면에 생성한다.

4. Prop

```
5 public class Prop : MonoBehaviour
6 {
7     public int score = 20 ;
8     public ParticleSystem explosionPartical;
9     public float hp = 10;
10
11     public void TakeDamage(float damage)
12     {
13         hp -= damage;
14
15         if(hp <= 0)
16         {
17             ParticleSystem instance = Instantiate(explosionPartical, transform.position, transform.rotation);
18             //(생성할 Object, 생성위치, 회전
19
20             AudioSource explotionAudio = instance.GetComponent<AudioSource>();
21
22             //explosionPartical.transform.parent = null; //부모 gameObject 를 지운다.
23             //tranform 은 부모와 자식 Object 간에 관계를 활용 할때 사용한다.
24             //explosionPartical.Play();
25
26             explotionAudio.Play();
27             GameManager.instance.AddScore(score);
28
29             Destroy(instance.gameObject, instance.duration);
30             gameObject.SetActive(false);
31             //Prop를 파괴 하는 것이 아닌 보이지 않도록 했다가 다시 보여 주기 위해서 사용
32         }
33     }
34 }
```

- Prop Class 에서 각 Prop 마다 Score(점수) 및 hp 를 지정해준다.
- 11 라인 TakeDamage() 함수의 damage 를 입력 받은 후 hp 에 적용한다.
- 15 라인 if 에서 hp 가 '0' 이하로 감소하면 Object의 위치및 회전량을 입력 받아 AudioSource 를 생성한다.
- 26, 27 라인에서 Audio를 재생 후 총 점수를 더 한다. 29 라인에서 duration를 파괴 후 gameObject는 파괴가 아닌 보이지 않도록 한다.(Game 재시작시 계속 다시 생성하는 부하 를 줄이기 위해서.)

5. Ball

```
28 private void OnTriggerEnter(Collider other)//물체의 충돌을 감지하고 other으로 충돌 한 물체가 무엇인지 알 수 있다.
29 {
30     Collider[] colliders = Physics.OverlapSphere(transform.position, explosionRadius, whatIsProp);
31     //(가상원을 만들 위치, 반지름, 필터)
32
33     for(int i = 0; i < colliders.Length; i++)
34     {
35         Rigidbody targetRigidbody = colliders[i].GetComponent<Rigidbody>();
36
37         targetRigidbody.AddExplosionForce(explosionForce, transform.position,
38             explosionRadius);
39
40         Prop targetProp = colliders[i].GetComponent<Prop>();
41
42         float damage = CalulateDamage(colliders[i].transform.position);
43
44         targetProp.TakeDamage(damage);
45     }
46
47     explotionParticle.transform.parent = null; //부모 gameObject 를 지운다.
48     //tranform 은 부모와 자식 Object 간에 관계를 활용 할때 사용한다.
49
50     explotionParticle.Play();
51     explotionAudio.Play();
52     GameManager.instance.OnBallDestory();
53
54     Destroy(explotionParticle.gameObject, explotionParticle.duration);
55     // duration 을 사용하여 해당 Object 의 running time을 알 수 있다.
56
57     Destroy(gameObject);
58
59 }
```

```
60 private float CalulateDamage(Vector3 targetposition)
61 {
62     Vector3 explrotonToTarget = targetposition - transform.position;
63
64     float distance = explrotonToTarget.magnitude;
65
66     float edgetoCenterDistance = explosionRadius - distance;
67
68     float percentage = edgetoCenterDistance / explosionRadius;
69
70     float damage = maxDamage * percentage;
71
72     damage = Mathf.Max(0, damage);
73     // damage 가 0 미만으로 감소 하는 경우 hp 가 회복 되는 것을 방지 하기 위해서
74     // 0 미만 인 경우 해당 값을 0으로 변경채
75
76     return damage;
77 }
```

- 28 라인 OnTriggerEnter() 함수에서 Ball GameObject 가 충돌한 Object를 확인 할 수 있다.
- 30 라인 에서 Ball 이 충돌한 위치에서 지정된 반지름의 크기만큼 원을 그리고 해당 원안에서 whatIsProp 필터를 사용하여 Prop Object만 가지고 온다.
- 37 ~ 44 라인에서 확인 된 Prop Object 에 TakeDamage() 함수를 통해 Damage를 부여 한다.
- 60 라인에 CalulateDamage() 함수를 통해 Ball의 충돌 시점부터 Prop 까지의 거리를 구해 거리에 따른 damage를 return 한다.

6. BallShooter

```
55     powerSlider.value = minForce;
56
57     if(currentForce >= maxForce && !fired)
58     {
59         currentForce = maxForce;
60         Fire();
61     }
62     else if (Input.GetButtonDown("Fire1"))
63     {
64         fired = false;
65         //true 를 사용하면 연사 가능함!
66
67         currentForce = minForce;
68         shootingAudio.clip = chargingClip;
69         shootingAudio.Play();
70     }
71     else if (Input.GetButton("Fire1") && !fired)
72     {
73         currentForce = currentForce + chargeSpeed * Time.deltaTime;
74
75         powerSlider.value = currentForce;
76     }
77     else if(Input.GetButtonUp("Fire1") && !fired)
78     {
79         Fire();
80         //발사처 us
81     }
```

```
83     public void Fire()
84     {
85         fired = true;
86
87         Rigidbody ballInstance = Instantiate(ball, firePos.position, firePos.rotation);
88
89         ballInstance.velocity = currentForce * firePos.forward; //해당 Object 에 앞쪽 방향
90
91         shootingAudio.clip = fireClip;
92         shootingAudio.Play();
93
94         currentForce = minForce;
95
96         cam.SetTarget(ballInstance.transform, CamFollow.State.Tracking);
97     }
```

- 57 ~ 61 라인 if 문에서 currentForce (Slider Gage) 값 을 초과한 경우 자동으로 Ball을 발사 하기 위한 Fire() 함수를 실행한다.
- 62 ~ 70 라인에서 currentForce 의 최소 값을 정하고 shootingAudio 를 재생한다.
- 71 ~ 76 라인에서 Slider 의 Gage 값을 증가 시키고 증가하는 동안 shootingAudio를 재생 시킨다.
- 77 ~ 81 라인에서 왼쪽 이미지의 83 라인의 Fire() 함수를 실행한다.

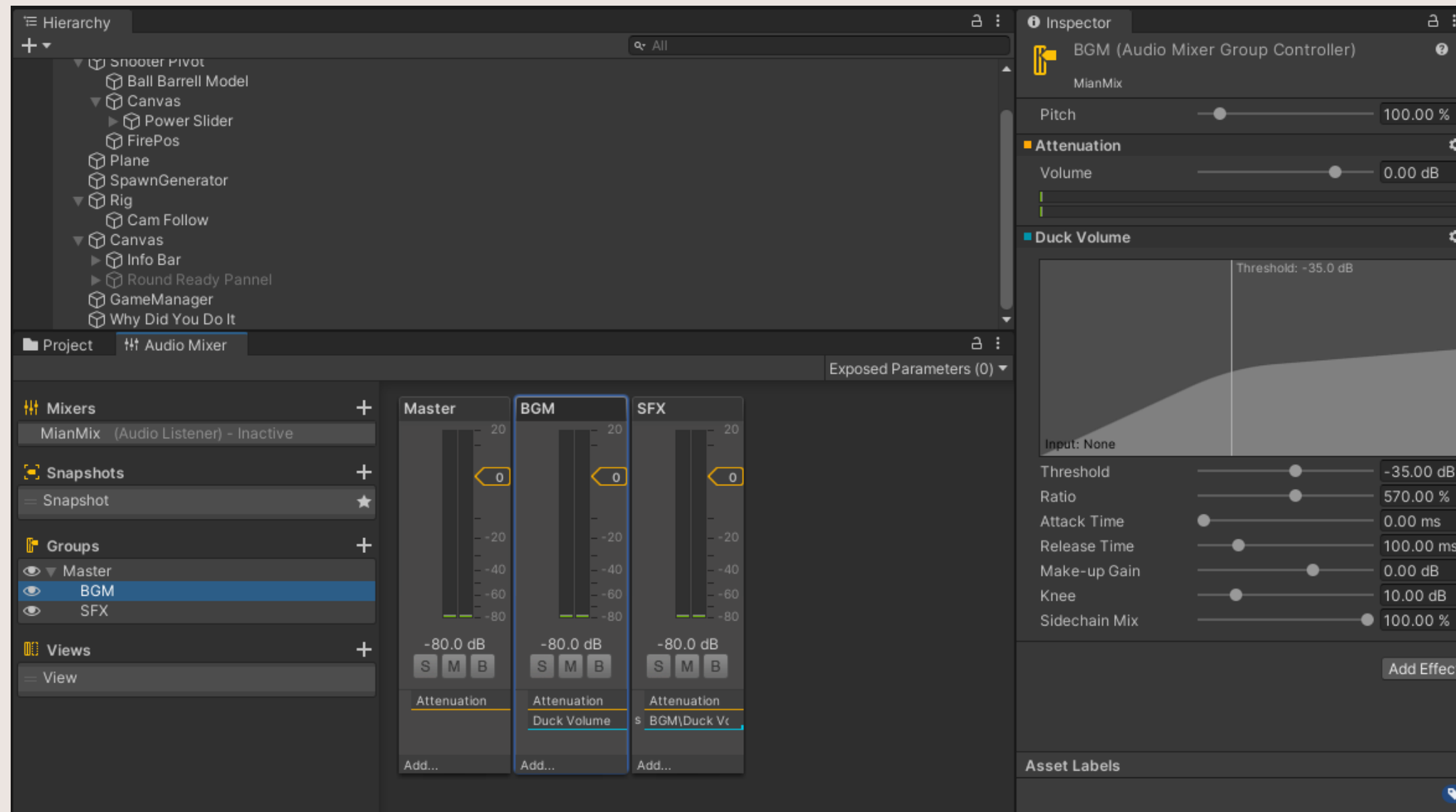
7. BallShooter

```
7  public enum State
8  {
9      Idle, Ready, Tracking
10 }
11 private State state
12 {
13     set
14     {
15         switch (value)
16         {
17             case State.Idle:
18                 targetZoomSize = roundReadyZoomSize;
19                 break;
20             case State.Ready:
21                 targetZoomSize = readysotZoomSize;
22                 break;
23             case State.Tracking:
24                 targetZoomSize = trackingZoomSiae;
25                 break;
26         }
27     }
28 }
```

```
54 private void Move()
55 {
56     targetPosition = target.transform.position;
57
58     Vector3 smoothPostion = Vector3.SmoothDamp(transform.position, targetPosition,
59         ref lastmovingVelocity, smoothTime);
60
61     transform.position = smoothPostion;
62 }
63
64 private void Zoom()
65 {
66     float smoothZoomSize = Mathf.SmoothDamp(cam.orthographicSize, targetZoomSize,
67         ref lastZommSize, smoothTime);
68
69     cam.orthographicSize = smoothZoomSize;
70 }
```

- 7 라인에 enum 을 이용한 State 정의 (Case 문과 상성이 좋다.)
- 11 ~ 28 라인을 통해 각 State 에 따른 Zoom Size 를 결정한다.
- 54 라인 Move() 함수의 56 라인을 통해 카메라가 따라갈 Target을 지정하고 58, 59 라인에서 카메라의 위치로부터 Target까지 smoothTime에 따라 천천히 따라간다.
- 64 라인 Zoom() 함수의 66, 67 라인에서 카메라(cam) 의 Zoom 사이즈를 결정하는데 마지막 Zoom 사이즈 값에서 smoothTime에 따라 천천히 변경 된다.

8. AudioMix



- Audio Mixer 의 Duck Volume 기능을 사용하여 SFX 소리가 일정 크기 이상으로 커지면 BGM 의 소리를 작게 수정한다.
- Game 에서 스킬 사용이나 특정 상황에서 배경음악 소리가 작아 저야하는 경우 많이 사용된다.

- 앞으로 나아갈 길...



1. 모바일 환경 해서 플레이 가능하도록 모바일용 UI 추가.
2. 모바일 환경 개발 후 이번 게임에서 MySql 연동 후 DB에 Score저장 기능 추가.
3. Game Menu 추가. (Menu 기능 구상 중...)

마지막으로...

이번 프로젝트에서 MySql 연동 후 Data관리 까지를 중점으로 진행할 예정 입니다.
각 프로젝트 마다 목표를 정하고 퇴근 후 또는 주말에 조금씩 조금씩 구현 중입니다!
아직도 한참 부족하지만 눈 앞에 목표를 하나씩 해결 해서 언젠가는..!

끝 까지 읽어 주셔서 감사합니다!