캡스톤 디자인

2022년 46조

20163165 최민혁
20163168 최원준

2D 종스크롤 게임

목차

#1, 개발 내용

#2, 주요 기술

#3, 중간발표 피드백

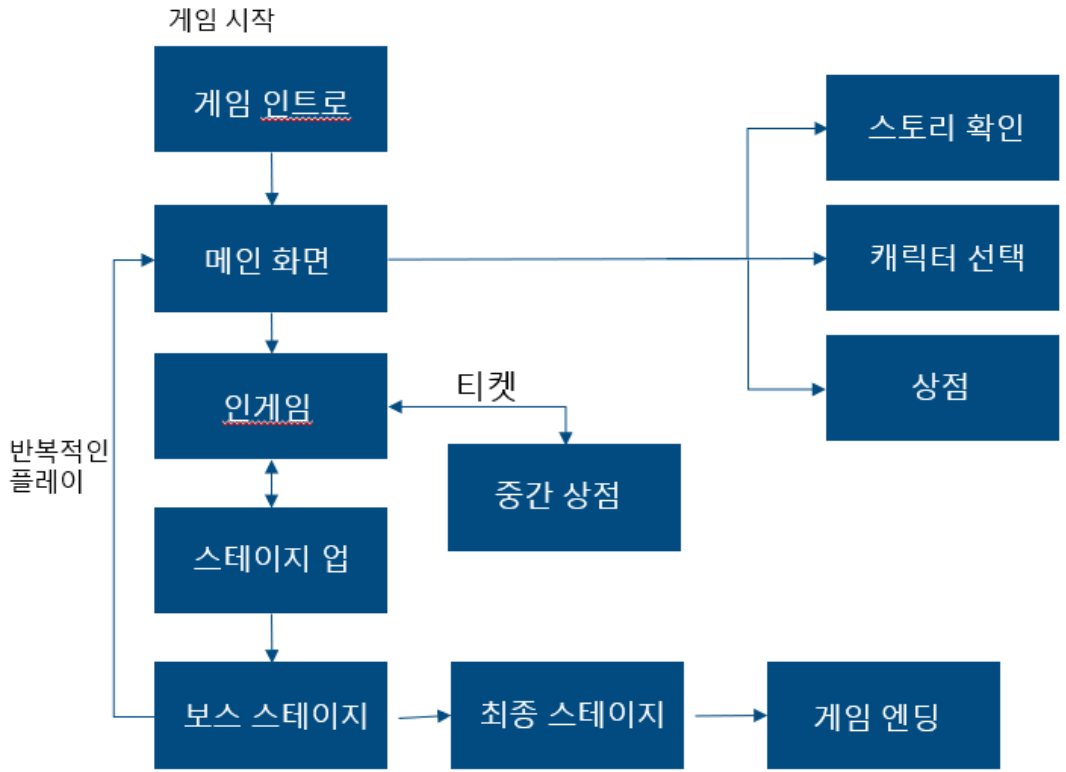
#4, 앞으로의 계획

Part 1,

개발 내용

Part 1

게임 개발 내용



게임 전체 구성

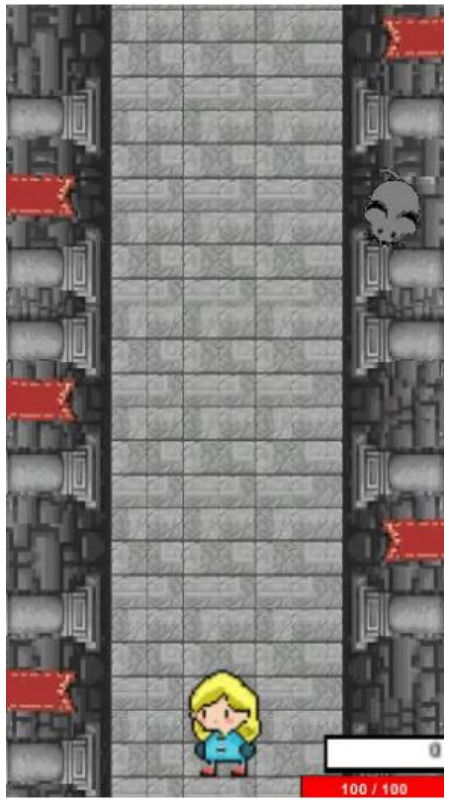
타이틀

로비화면

인게임

Part 1

게임 전체 구성

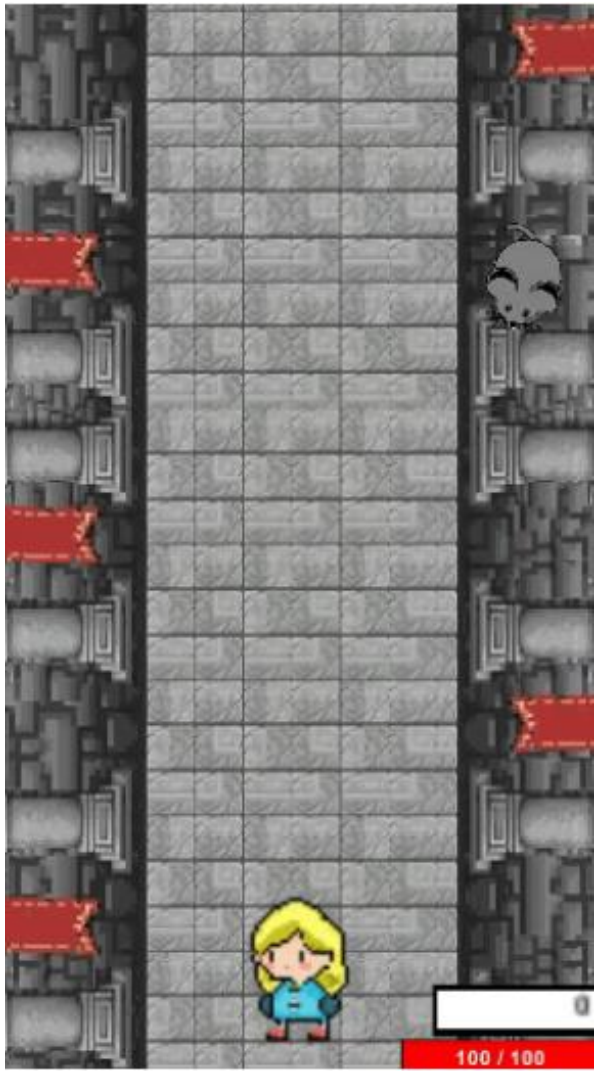


Part 1

게임 로비 화면

Part 1

게임 로비 화면



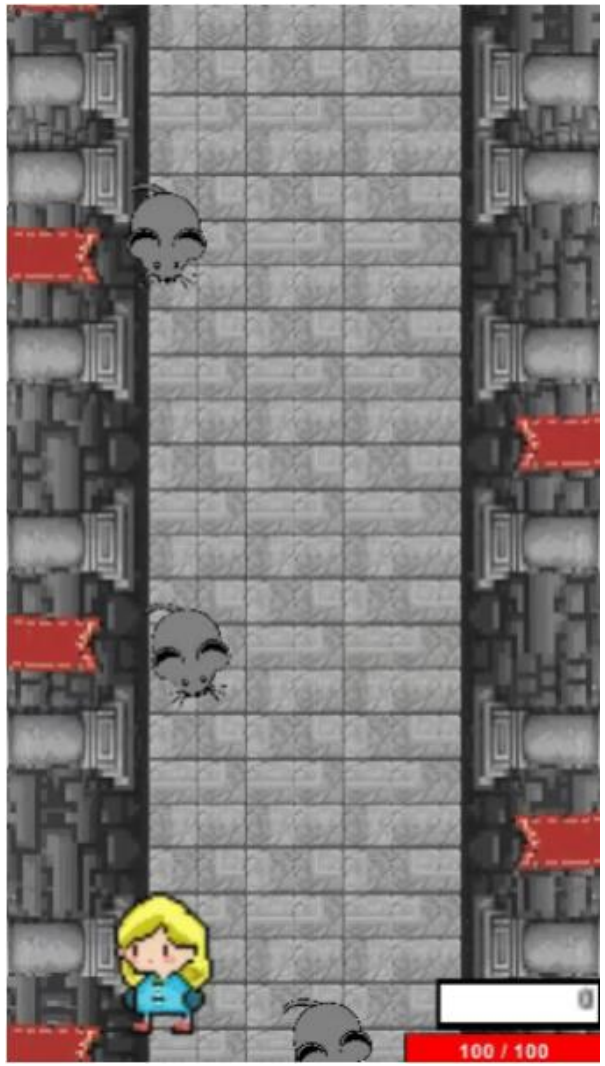
적 패턴


```
void Update()
{
    if (enemyName == "enemy001")
        moveControl();
    else if (enemyName == "enemy003")
        fly();
    else simpleMove();

}
```

```
void simpleMove()
{
    float distanceY = Speed * Time.deltaTime;
    this.gameObject.transform.Translate(0, -1 * distanceY, 0);
}
```

Part 1

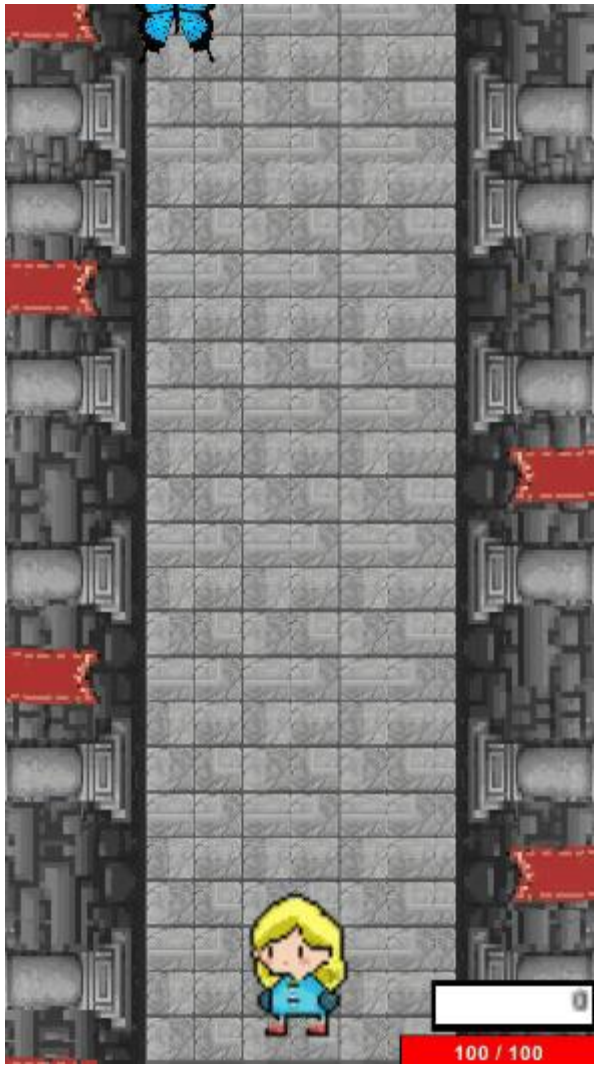
적 패턴




```
private void Start()
{
    if (enemyName == "enemy001")
    {
        dirVec = Player.instance.transform.position - transform.position;
        float angle = Mathf.Atan2(dirVec.y, dirVec.x) * Mathf.Rad2Deg;
        transform.rotation = Quaternion.AngleAxis(angle + 90, Vector3.forward);
    }
}
```

```
void moveControl()
{
    transform.position = transform.position + dirVec.normalized * Speed * Time.deltaTime;
}
```

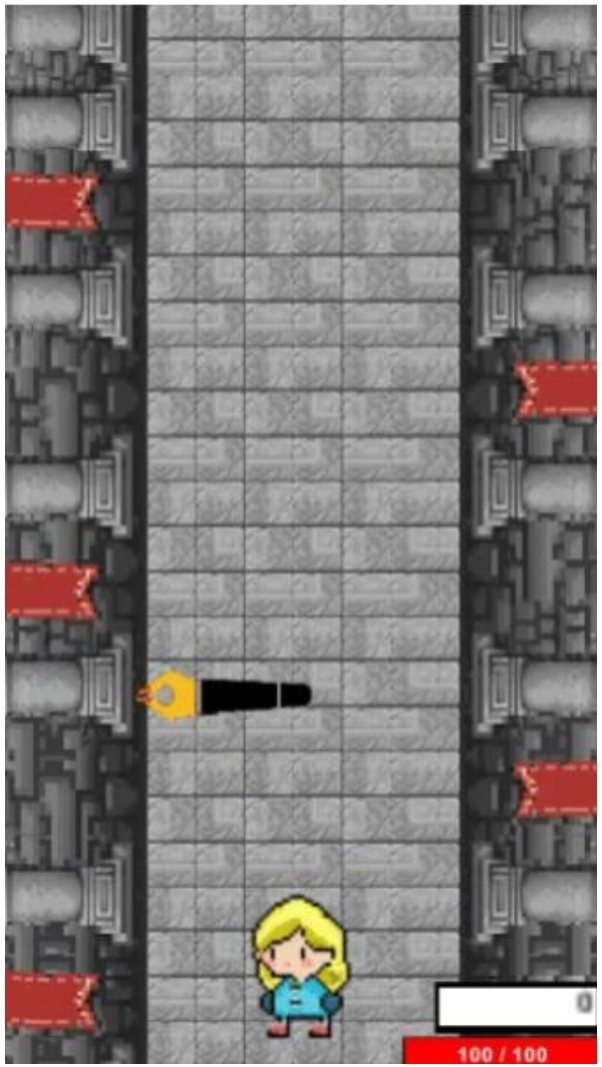
적 패턴

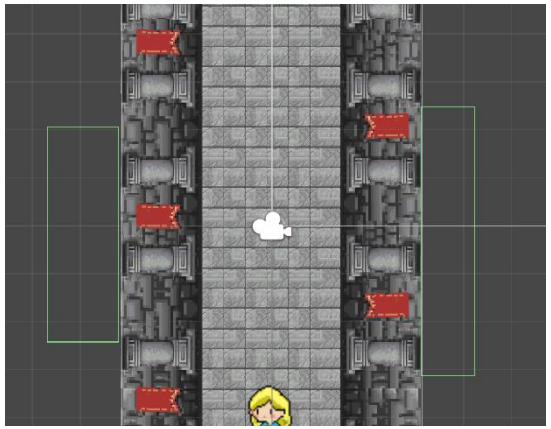



```
void fly()
{
    transform.Rotate(Player.instance.transform.position - transform.position);//팔랑대기
    transform.position = transform.position + (Player.instance.transform.position - transform.position).normalized * Speed * Time.deltaTime; // 쫓아오기
```

Part 1

적 패턴

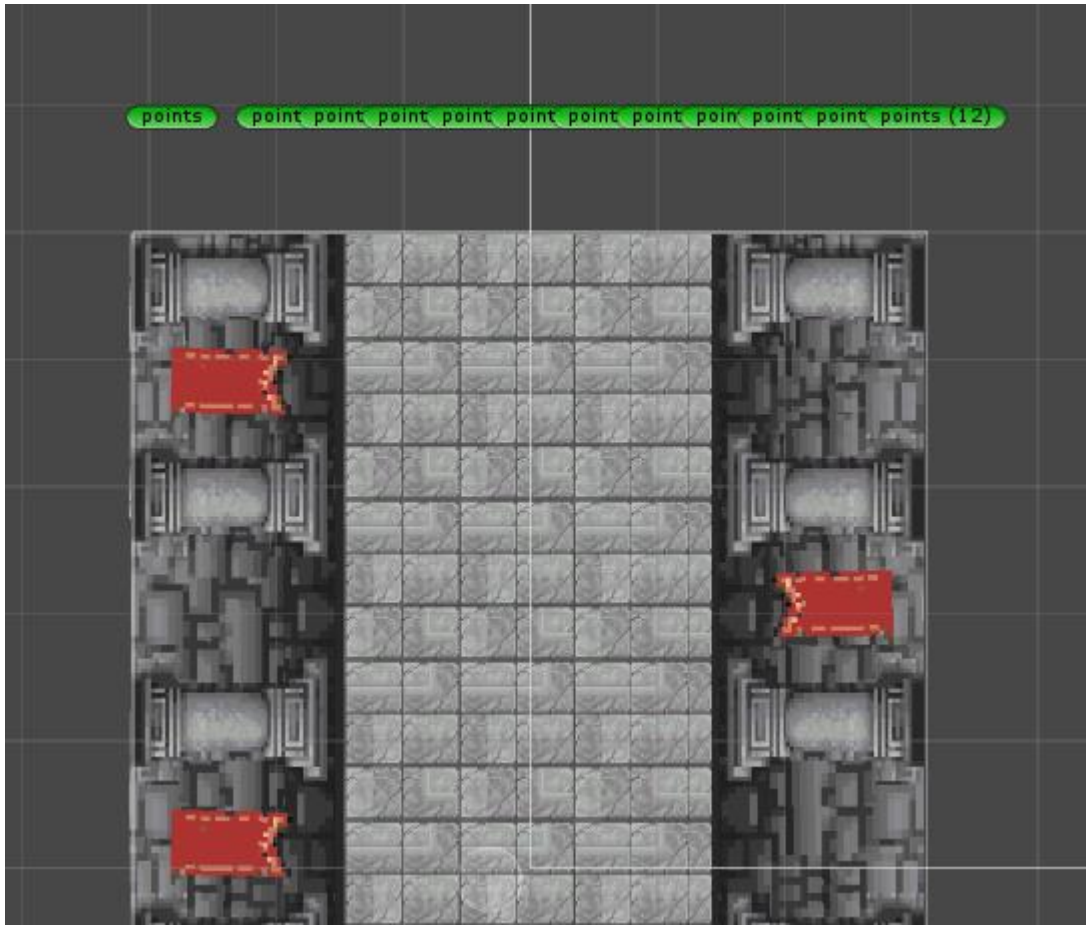




```
private IEnumerator Spawn()
{
    while(true)
    {
        yield return new WaitForSeconds(delay);

        Vector3 spawnPos = get_RandomPosition();
        if(dir == 1)
        {
            GameObject instance = objectManager.MakeObj("penRight");
            instance.transform.position = spawnPos;
        }
        else
        {
            GameObject instance = objectManager.MakeObj("penLeft");
            instance.transform.position = spawnPos;
        }


    }
```

Part 2

스테이지 파일



Part 2

스테이지 파일


```
void ReadSpawnFile()
{
    //변수 초기화
    spawnList.Clear();
    spawnIndex = 0;
    spawnEnd = false;
    //리스폰 파일 읽기
    TextAsset textFile = Resources.Load("stage1") as TextAsset;
    StringReader stringReader = new StringReader(textFile.text);

    while (stringReader != null)
    {
        string line = stringReader.ReadLine();

        if (line == null)
            break;
        //리스폰 데이터 생성
        Spawn spawnData = gameObject.AddComponent<Spawn>();
        spawnData.delay = float.Parse(line.Split(',')[0]);
        spawnData.type = line.Split(',')[1];
        spawnData.point = int.Parse(line.Split(',')[2]);
        spawnList.Add(spawnData);
    }
    //텍스트파일 닫기
    stringReader.Close();
    nextSpawnDelay = spawnList[0].delay;
}
```

Part 2

플레이어 피격


```
void OnDamaged(Vector2 targetPos)
{
    gameObject.layer = 10;

    spriteRenderer.sprite = char_sprite[1];  //이미지 바꿈

    spriteRenderer.color = new Color(1, 1, 1, 0.6f);

    int dirc = transform.position.x - targetPos.x > 0 ? 1 : -1;
    StartCoroutine(KnockBack(dirc));

    Invoke("OffDamaged", 0.4f);

}

void OffDamaged()
{
    gameObject.layer = 7;
    spriteRenderer.sprite = char_sprite[0];
    spriteRenderer.color = new Color(1, 1, 1, 1);
}
```

Part 2

플레이어 죽음


```
// 게임종료 체크 (플레이어 체력으로)
if(Player.health <= 0 )
{
    Time.timeScale = 0.0f;
    GameOver();
}
```

```
public void GameOver()
{
    dead.SetActive(true); //애니메이션 활성
    dead.transform.position = Player.instance.transform.position;  //위치잡아줌
    Player.instance.spriteRenderer.color = new Color(1, 1, 1, 0f);

    gameOverImg.SetActive(true);
    StartCoroutine(goGameOverScene());
}
```

Part 2

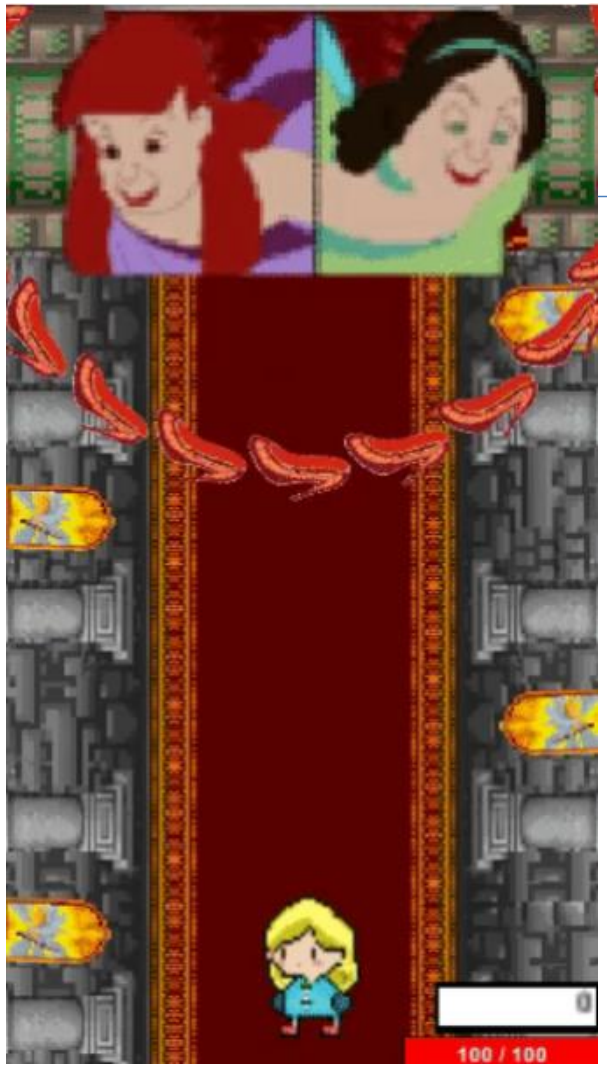
중간상점

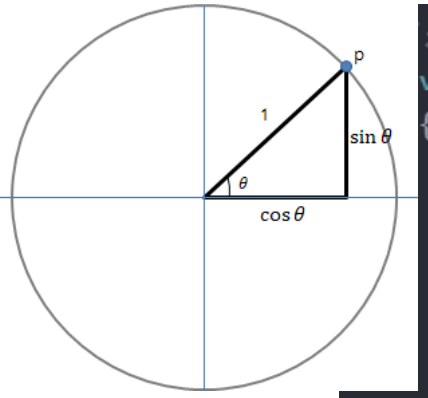

```
if (health <= 0)
{
    int ran = Random.Range(0, 100);//퍼센테이지로 표기 0~100 무언가
    if (ran < ticketDrop)
    {

        GameObject itemTicket = objectManager.MakeObj("itemTicket");
        if (itemTicket != null)
        {
            itemTicket.transform.position = transform.position;
        }
        else
        {
            GameObject itemCoin = objectManager.MakeObj("itemCoin");
            itemCoin.transform.position = transform.position;
        }
    }
```

```
if (col.gameObject.tag == "Ticket")//티켓 습득 시 상점 호출
{
    GameManager manager = GameObject.Find("GameManager").GetComponent<GameManager>();
    manager.shopSet.SetActive(true);
}
```

Part 2

원형 패턴

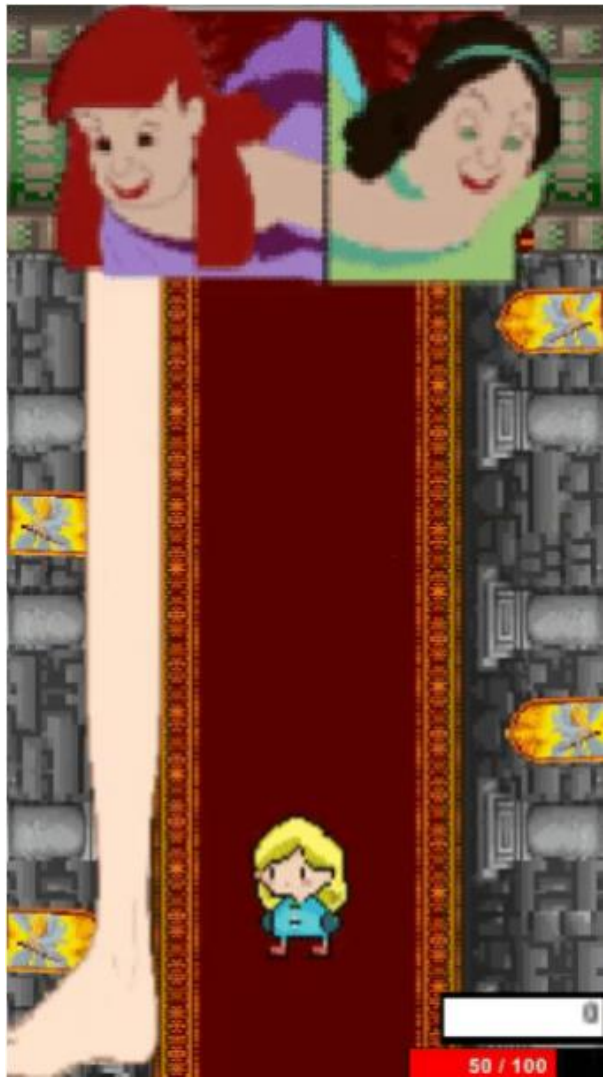


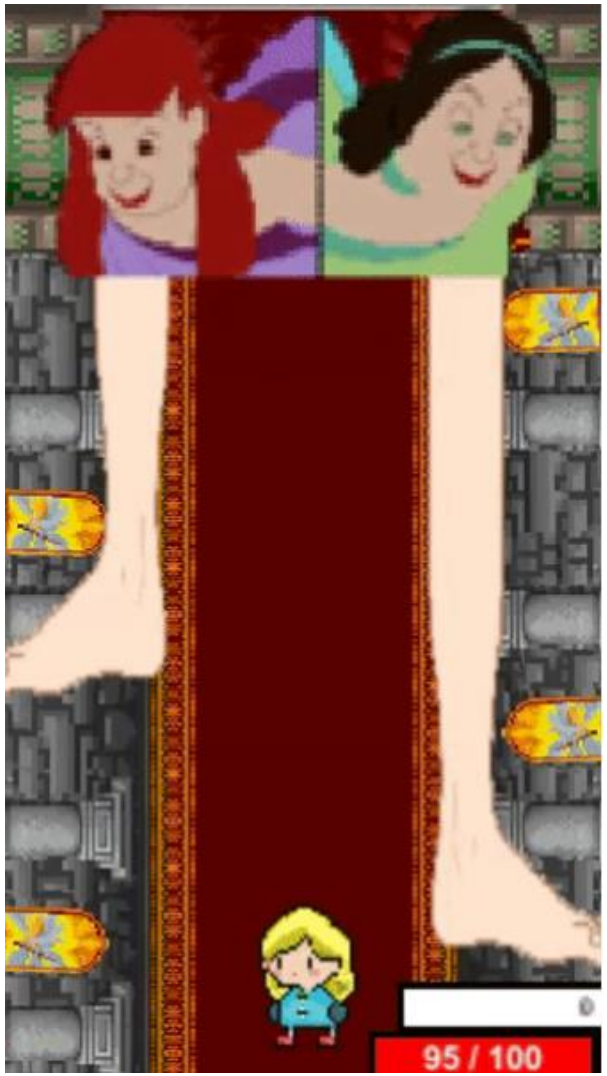



```
원 패턴
void FireRight()
{
    int bulletNum = 20;
    for(int index = 0; index < bulletNum; index++)
    {
        Debug.Log(index);
        GameObject bullet = objectManager.MakeObj("bulletBossSisters");
        //Debug.Log("보스신발 생성");
        bullet.transform.position = transform.position;
        bullet.transform.rotation = Quaternion.identity;

        Rigidbody2D rigid = bullet.GetComponent<Rigidbody2D>();
        Vector2 dirVec = new Vector2(Mathf.Cos(Mathf.PI * 2 * index / bulletNum),
                                     Mathf.Sin(Mathf.PI * 2 * index / bulletNum));
        rigid.AddForce(dirVec.normalized * 10, ForceMode2D.Impulse);

        Vector3 rotVec = Vector3.forward * 360 * index / bulletNum + Vector3.forward * 90;
        bullet.transform.Rotate(rotVec);
    }
```

Part 2

발 패턴





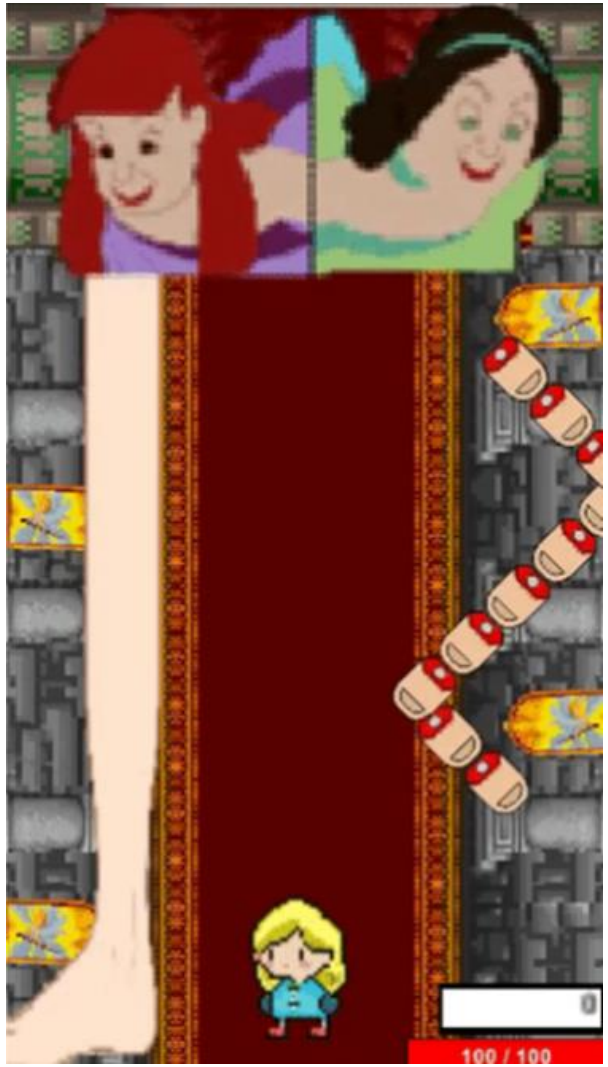
Part 2

발 패턴

```
//발 패턴 (왼쪽)
  void kickLeft()
  {

      Debug.Log("왼쪽 발 패턴");
      Instantiate(boss_foot_L);

  }

  //발 패턴 (오른쪽)
  void kickRight()
  {
      Debug.Log("오른쪽 발 패턴");
      Instantiate(boss_foot_R);

  }
```

Part 2

손가락




```
//s자 패턴
void finger_S(int dir)
{
    float x_dir ;
    if(dir == 0){x_dir = Random.Range(0f, 1.5f); } //0이면 오른쪽
    else{ x_dir = Random.Range(-3f, -1f); }
    StartCoroutine(finger_spawn(x_dir));
}

 IEnumerator finger_spawn(float x_dir)
{
    for(int i = 0; i < 10 ; i ++)
    {
        GameObject finger = objectManager.MakeObj("bossFinger");
        finger.transform.position = new Vector3(x_dir, 2.5f, 0);
        yield return new WaitForSeconds(0.2f);
    }

}
```

Part 2

보스 페이즈

```
void Think()
{

    if(health >= (maxHP/4)*3)
    {
        phase_One();
    }
    else if(health >= (maxHP/4)*2)
    {
        phase_Two();
    }
    else
    {
        phase_Three();
    }
```

```
void phase_One()
{
    Debug.Log("phase_One 실행중");
    FireRight();
    Invoke("Think", 3f);
}
```

```
void phase_Two()
{
    int pattern = Random.Range(0, 5);
    Debug.Log("패턴 넘버 : " + pattern);
    switch(pattern)
    {
        case 0 :
        kickLeft();
        break;

        case 1 :
        kickRight();
        break;

        default :
        FireRight();
        break;

    }
    Invoke("Think", 3f);
}
```

Part 2

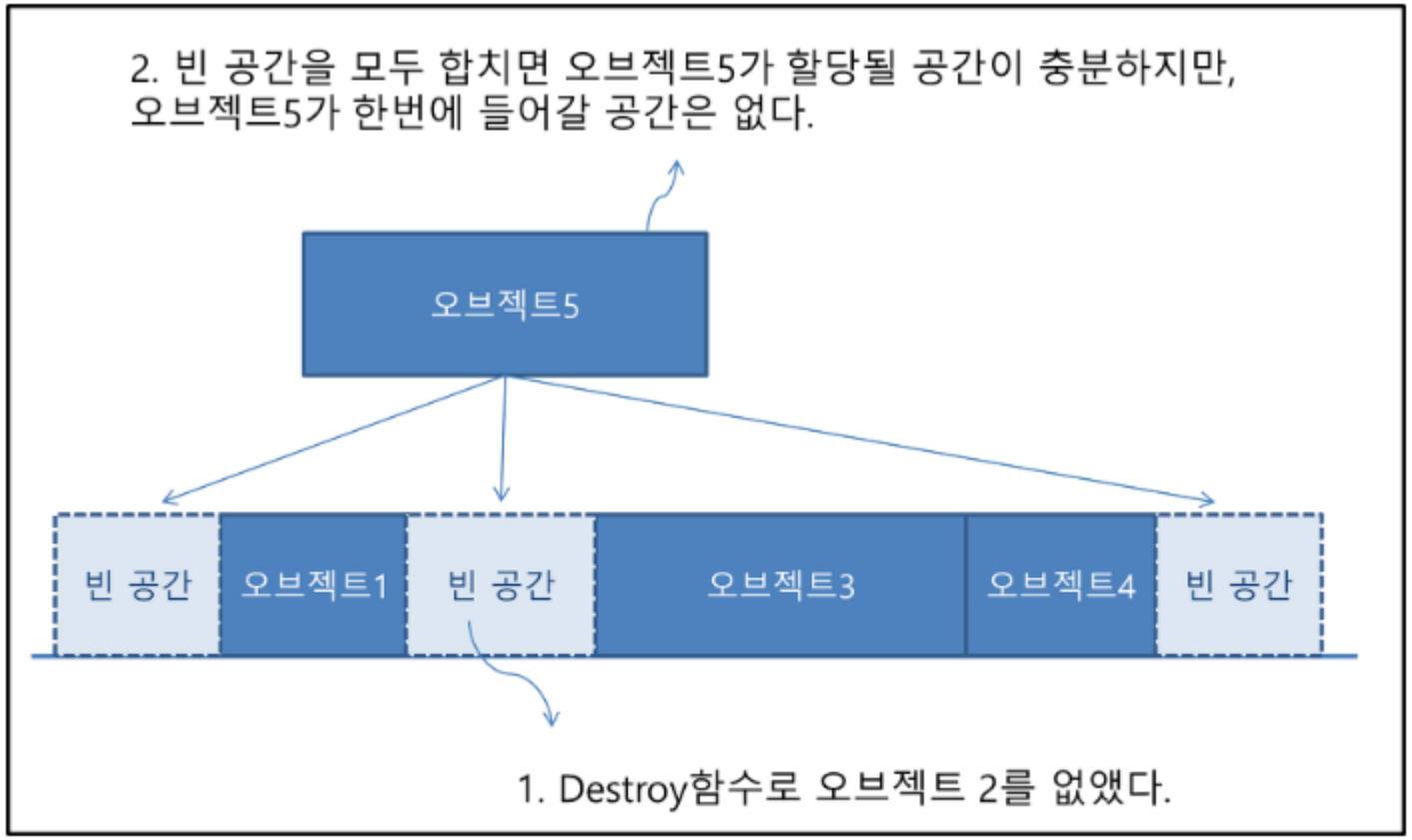
보스

```
void phase_Three()
{
    int pattern = Random.Range(0, 5);

    switch(pattern)
    {

        case 0 :
        foot_finger();
        break;

        case 1 :
        megalodon();
        break;


        default :
        FireRight();
        break;

    }

    Invoke("Think", 3f);
}
```

Part 2,

주요구현 부분

메모리 단편화(Memory fragmentation) 문제

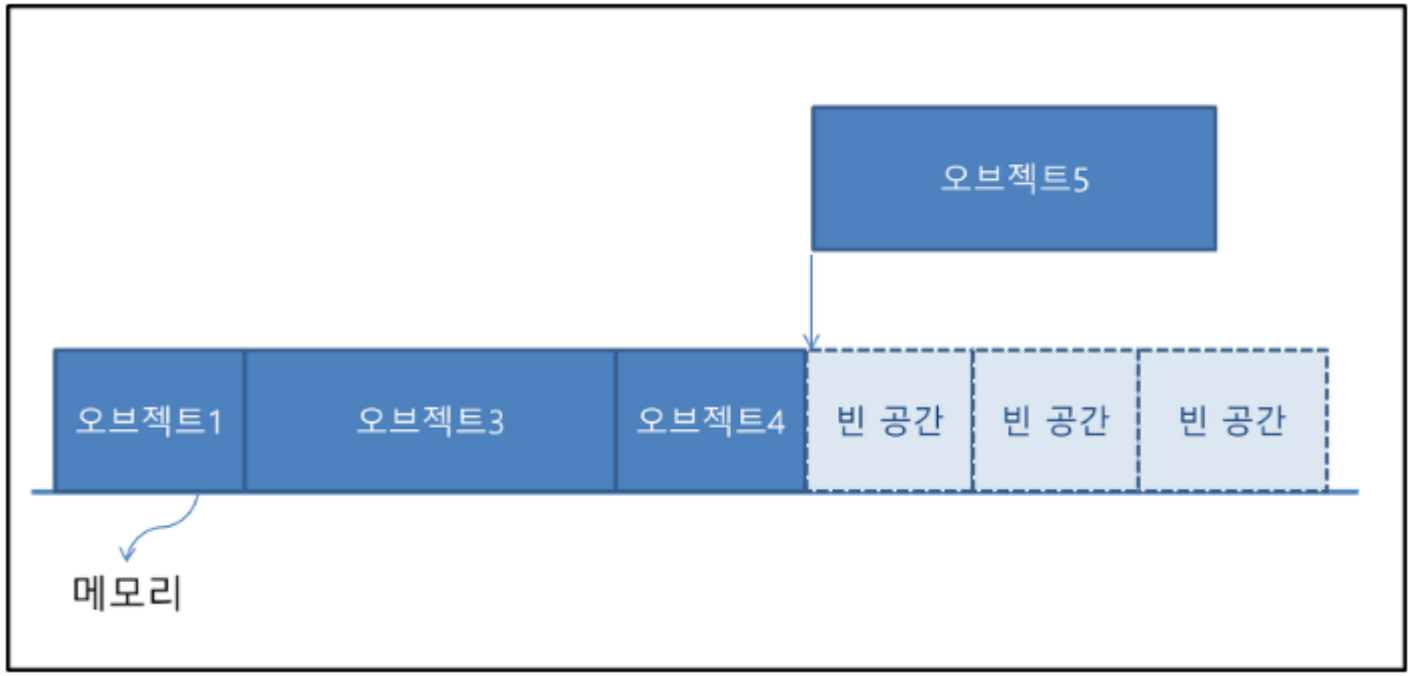


외부 메모리 단편화 상황

메모리 단편화(Memory fragmentation) 문제

- C# 에서는 가비지 컬렉터가 메모리 단편화를 Compaction (압축) 해서 해결

- 하지만 유니티에서는 지원 X



Part 2

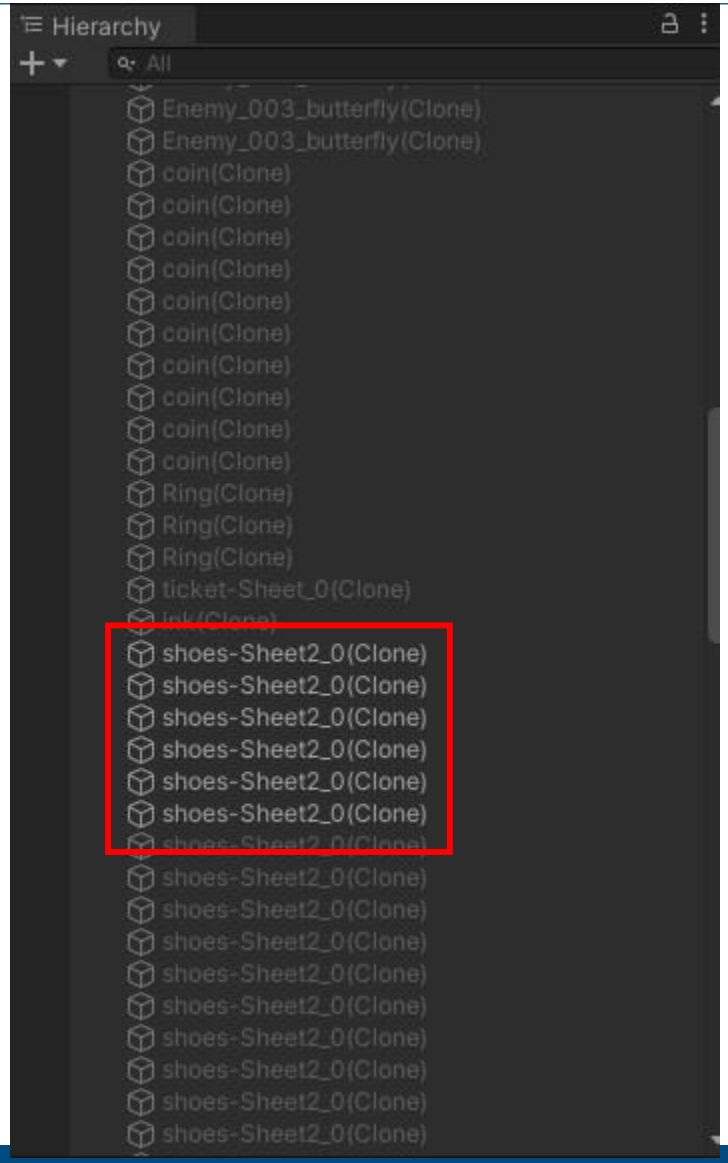
메모리 풀링

```
void Awake()
{
    enemy001 = new GameObject[20];
    enemy002 = new GameObject[10];
    enemy003 = new GameObject[10];
    bossSisters = new GameObject[1];

    itemCoin = new GameObject[10];
    itemRing = new GameObject[3];
    itemTicket = new GameObject[1];
    itemInk = new GameObject[1];

    bulletPlayer = new GameObject[20];
    bulletBossSisters = new GameObject[50];

    penRight = new GameObject[5];
    penLeft = new GameObject[5];
    bossFinger = new GameObject[15];


    Genarate();
}
```

```
void Genarate()
{
    for (int index = 0; index < enemy001.Length; index++)
    {
        enemy001[index] = Instantiate(enemy001Prefab);
        enemy001[index].SetActive(false);
    }

    for (int index = 0; index < enemy002.Length; index++)
    {
        enemy002[index] = Instantiate(enemy002Prefab);
        enemy002[index].SetActive(false);
    }

    for (int index = 0; index < enemy003.Length; index++)
    {
        enemy003[index] = Instantiate(enemy003Prefab);
        enemy003[index].SetActive(false);
    }
```

Part 2

메모리 풀링



Part 2

씬 이동 문제

```
private void Awake()
{
    DontDestroyOnLoad(gameObject);

}
```

Part 2

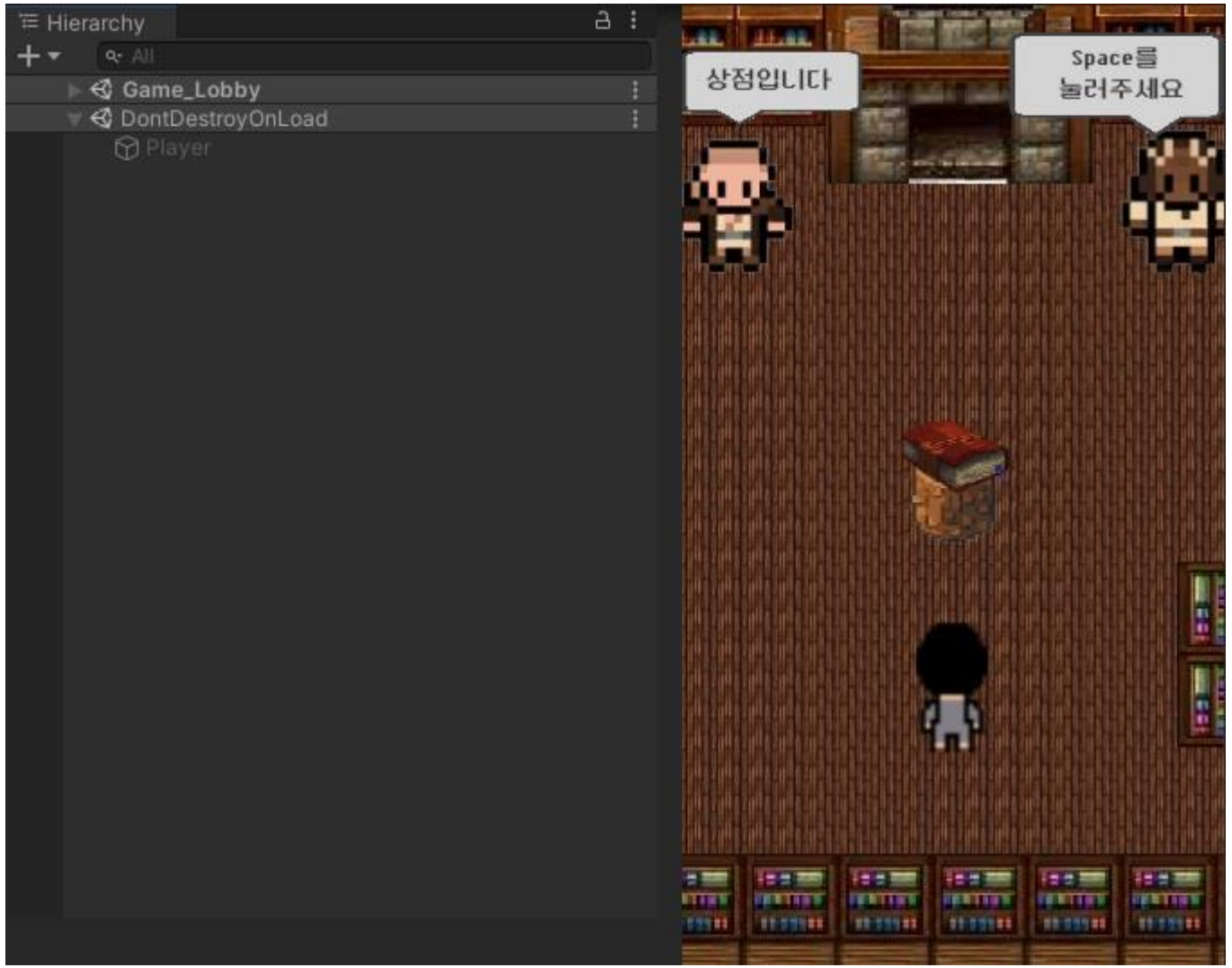
싱글턴

```
public class Player : MonoBehaviour
{
    // 싱글턴
    public static Player instance = null;

    public static float health = 100;
    public float Speed = 3f;
    public float dmg;
    public float dmg_F;
    public int maxPower;
    public int power;
    public float maxShotDelay;
    public float curShotDelay;
    public int score;
    public int ink;
    public int ring;
```

```
if (instance == null)
{
    instance = this;
    DontDestroyOnLoad(gameObject);
}
else
{
    if (instance != this)
      Destroy(this.gameObject);
}
```

Part 2

싱글턴

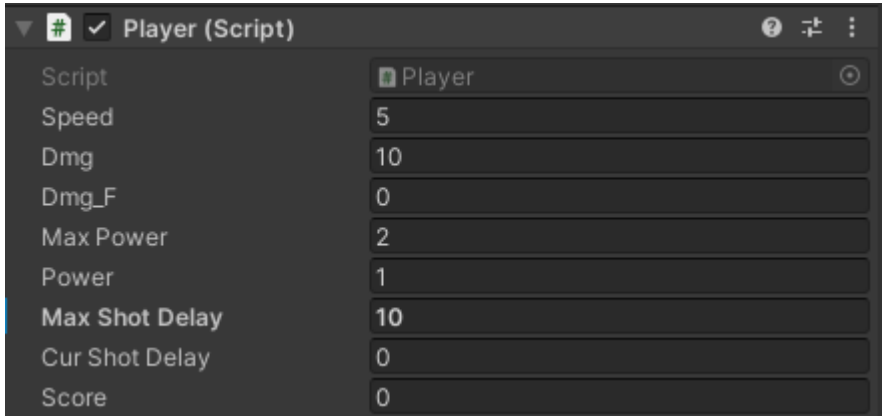


Part 2

데이터 저장

데이터 저장

흔히 사용하는 데이터 저장
DB, Json, Xml



유니티에서 제공하는
PlayerPrefs

데이터 저장

PlayerPrefs

Static Functions

DeleteAll	preference에서 모든 key와 값들을 제거합니다. 사용 시 경고가 뜹니다.
DeleteKey	키와 대응하는 값을 삭제합니다.
GetFloat	Preference 파일에 존재하는 /key/에 대응하는 값을 반환합니다.
GetInt	Preference 파일에 존재하는 /key/에 대응하는 값을 반환합니다.
GetString	Preference 파일에 존재하는 /key/에 대응하는 값을 반환합니다.
HasKey	키가 존재하는지 확인합니다.
Save	수정된 모든 preferences를 디스크에 씁니다.
SetFloat	/key/로 식별된 Preference의 값을 설정합니다.
SetInt	/key/로 식별된 Preference의 값을 설정합니다.
SetString	/key/로 식별된 Preference의 값을 설정합니다.

데이터 저장

Ex) 상점의 파워업 함수

```
public void UpgradePower()
{
    //Debug.Log("ǔ▯▯▯▯ ▯θ°▯▯▯?");
    if (!PlayerPrefs.HasKey("Power")) {
        PlayerPrefs.SetInt("Power", 0);}
    if (!PlayerPrefs.HasKey("Price1"))
        PlayerPrefs.SetInt("Price1", 1);
    if (PlayerPrefs.HasKey("Ink"))
    {

        if (PlayerPrefs.GetInt("Ink") >= PlayerPrefs.GetInt("Price1"))
        {
            PlayerPrefs.SetInt("Ink", PlayerPrefs.GetInt("Ink") - PlayerPrefs.GetInt("Price1"));
            PlayerPrefs.SetInt("Price1", PlayerPrefs.GetInt("Price1") + 1); // ▯▯▯▯ ▯▯▯▯
            Debug.Log("▯▯▯▯ " + PlayerPrefs.GetInt("Price1") + "▯▯ũ▯▯ ▯▯▯▯");
            PlayerPrefs.SetInt("Power", PlayerPrefs.GetInt("Power") + 1); //  ▯▯▯□▯▯▯ ▯▯▯▯
            Debug.Log("▯L▯ " + PlayerPrefs.GetInt("Power") + "▯▯ ▯▯▯▯");



        }

    }
```

Part 3,

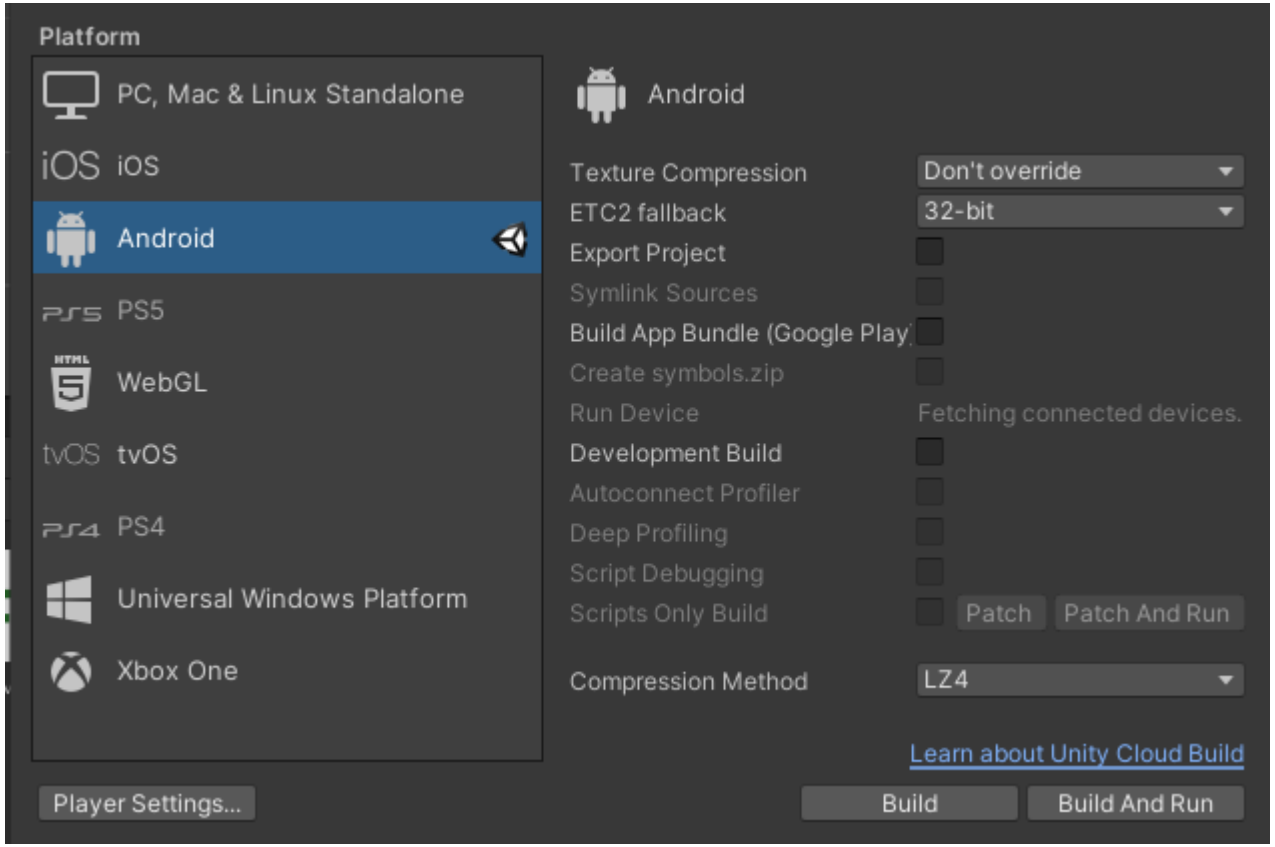
중간평가 피드백

Part 3

중간평가 피드백

- 모바일로도 개발을 꼭 했으면 좋겠다.
  ->안드로이드

- 유니티 자체적 변환가능

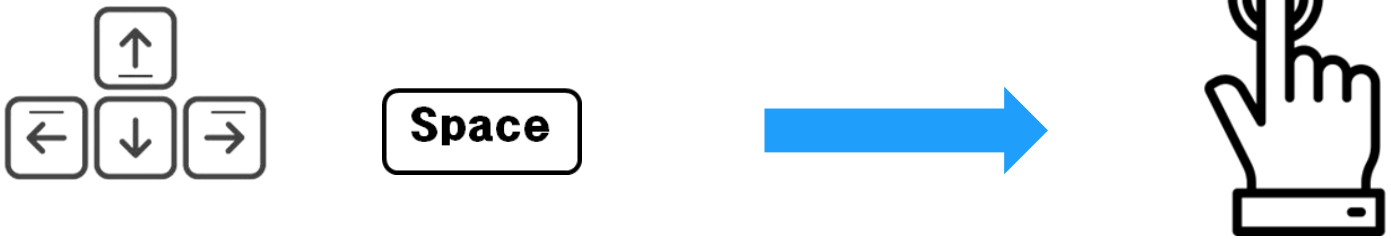


Part 3

중간평가 피드백

- 스마트폰에 맞춘 화면비율

- 방향키에서 스마트폰 터치로



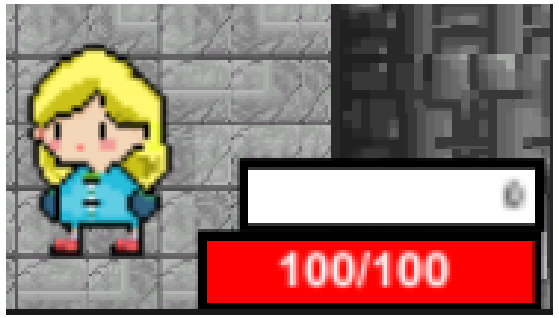
Part 3

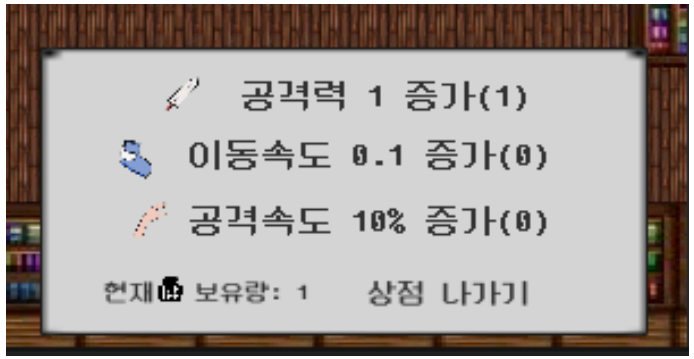
화면비율

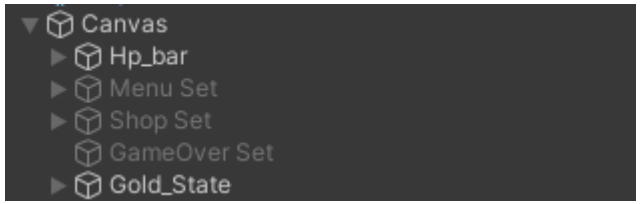
1,024,000	800 × 1280	10:16	갤럭시탭10.1 갤노트1	G패드10.1		넥서스7
1,646,400	840 × 1960	9:21	갤럭시폴드(커버)			
2,073,600	1080 × 1920	9:16	갤럭시S4 갤럭시S5 갤노트3	옵티머스G프로 G2	아이폰6+ 아이폰6S+ 아이폰7+ 아이폰8+	넥서스5 Pixel
2,304,000	1200 × 1920	10:16		G패드8.3		넥서스7(2013)
2,740,500	1125 × 2436	9:19.5			아이폰X	
3,145,728	1536 × 2048	3:4			아이패드미니레티나 아이패드에어2	
3,305,472	1536 × 2152	3:4.2	갤럭시폴드(메인)			
3,686,400	1440 × 2560	9:16	갤럭시S5광대역LTE-A 갤럭시S6 갤럭시S7 갤노트4 갤노트5 갤노트7	G3 G4 G5 V10 V20		Pixel XL
4,096,000	1600 × 2560	10:16	갤럭시탭S 갤럭시노트10.1(2014) 갤럭시노트프로12.2			넥서스10
4,147,200	1440 × 2880	9:18		G6 V30		
4,262,400	1440 × 2960	9:18.5	갤럭시S8 갤럭시S8+ 갤노트8			

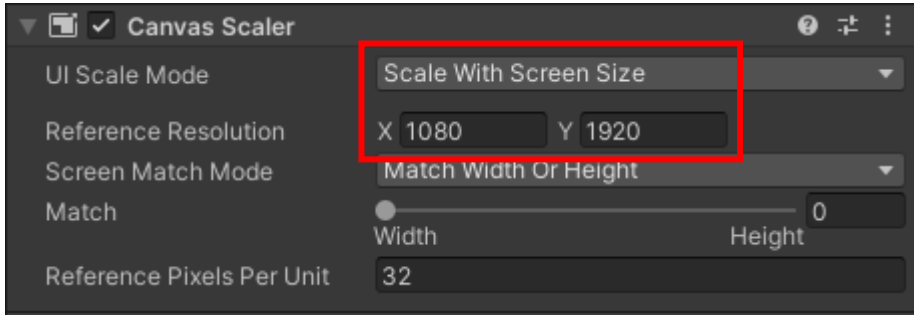
화면비율

- UI







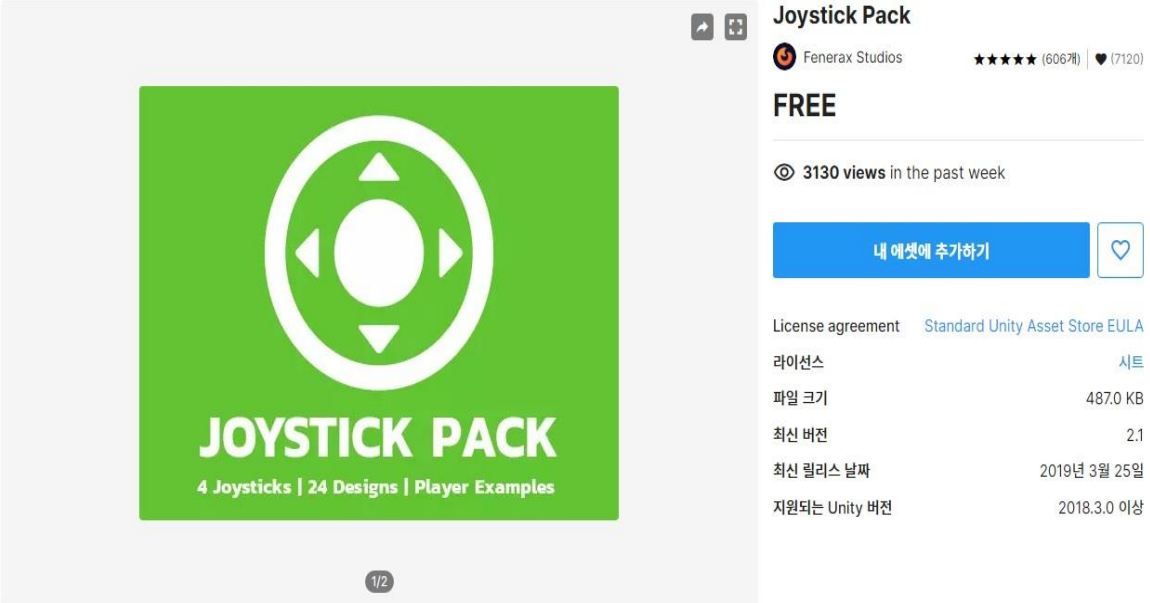


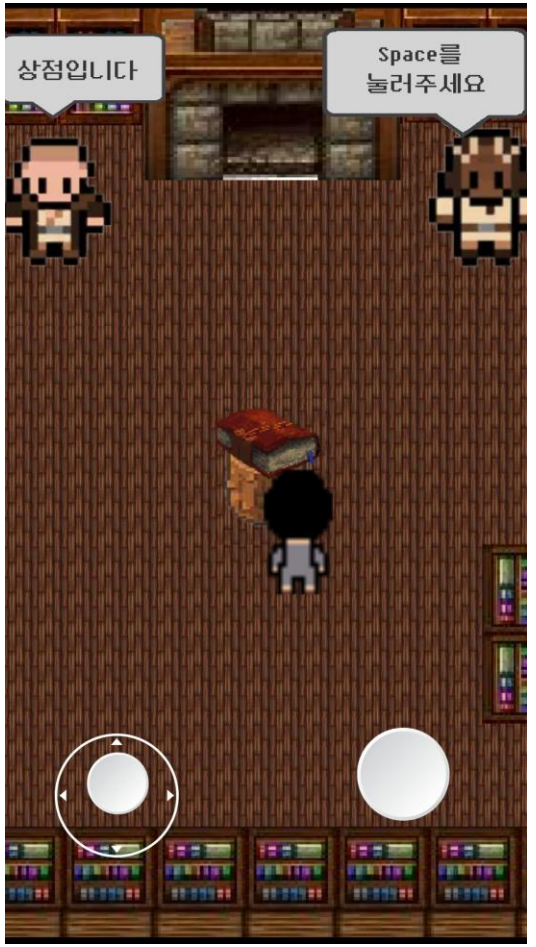
카메라

```
 void Start()
{
    Camera camera = GetComponent<Camera>();
    Rect rect = camera.rect;
    float scaleheight = ((float)Screen.width / Screen.height) / ((float)9 / 16); // (가로 / 세로)
    float scalewidth = 1f / scaleheight;
    if (scaleheight < 1)
    {
        rect.height = scaleheight;
        rect.y = (1f - scaleheight) / 2f;
    }
    else
    {
        rect.width = scalewidth;
        rect.x = (1f - scalewidth) / 2f;
    }
    camera.rect = rect;
}
```

터치 움직임

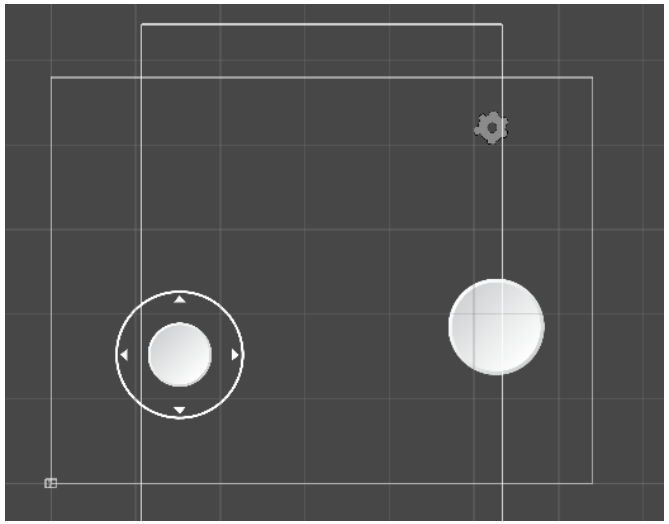
조이스틱

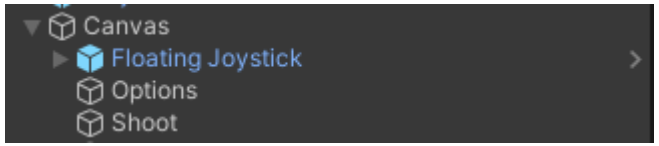




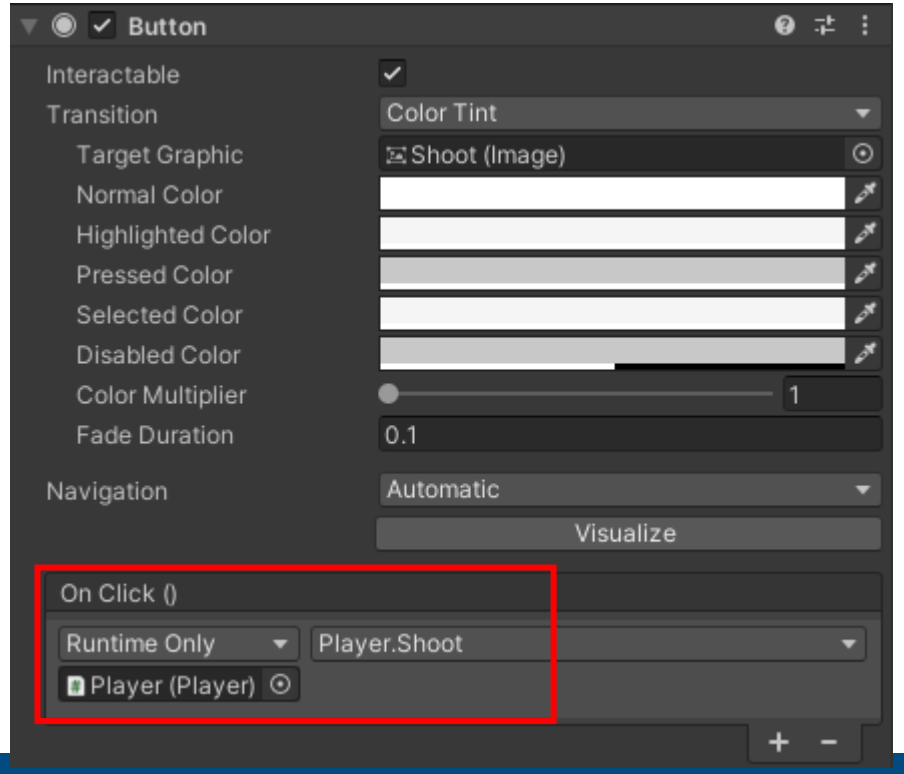
Part 3

터치 움직임






```
private void Move(){
    float x = joy.Horizontal;
    float y = joy.Vertical;
    Vector2 moveVec = new Vector2(x, y);
    Rigidbody2D rigid = GetComponent<Rigidbody2D>();
    rigid.MovePosition(rigid.position + moveVec * Speed * Time.deltaTime);
  }
```

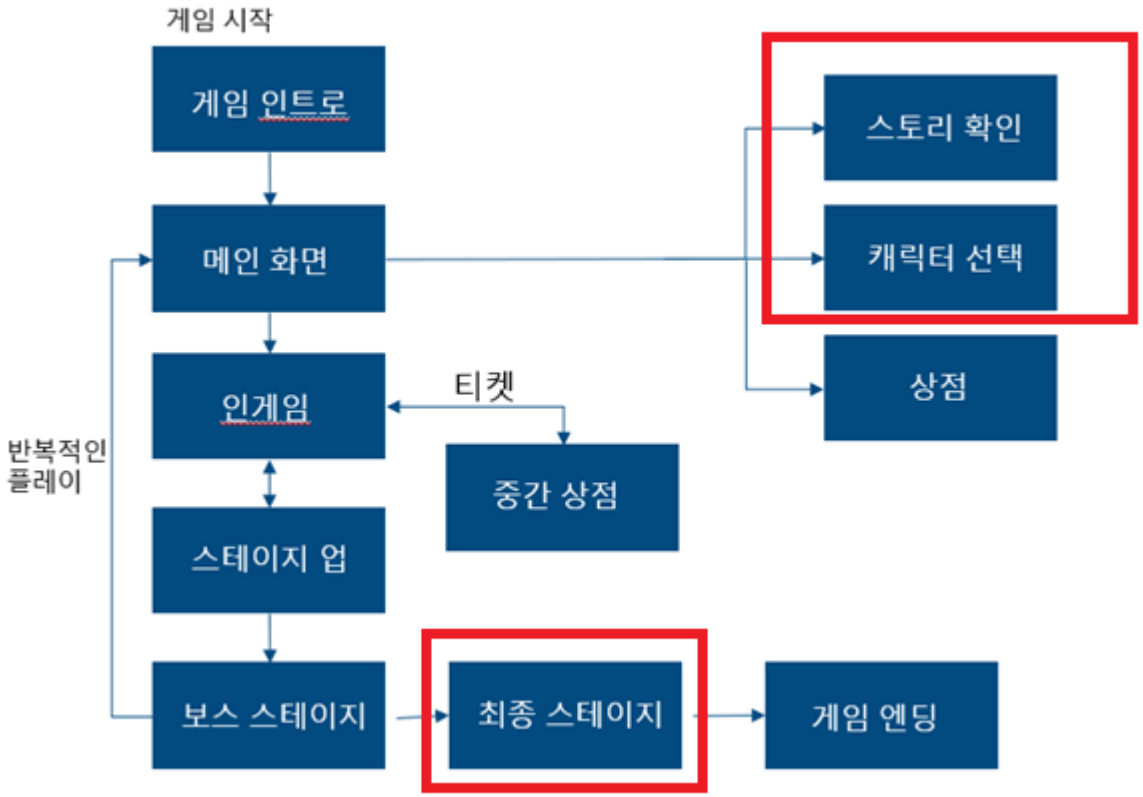


Part 4,

앞으로의 계획

Part 4

앞으로의 개발계획



(예시)

앞으로의 개발계획

보스 패턴 부분

Q & A

경청해주셔서 감사합니다.