게임
기획서

목차

1. 게임 정보

1. 게임 정보

1) 유사 게임

▲ STRIKERS 1945

적과 보스의 공격을 피해 물리치고 스테이지를 깨 나가는 슈팅 게임

1. 게임 정보

2) 장르

슈팅 게임

3) 대상

적과 보스를 무자비하게 공격해 죽이면서 쌓인 스트레스를 해소하고 싶은 성인 층

4) 플랫폼

PC 웹

5) 개발 환경

UNITY 2019.4.21f1

2. 게임 주제

2. 게임 주제

1) 제목

AI Space War 2065

2) 컨셉

행성 쟁취를 위해 우주에서 벌어진 AI들의 우주 전쟁

3) 시나리오

2065년, 우주에서 지구를 닮은 새로운 행성 '동왕성'이 발견되고,
각 국에서 선두로 이 행성을 차지하기 위해 AI를 보낸다.
행성으로 가는 도중 마주친 경쟁 AI들과의 전투가 시작되는데,,
행성을 쟁탈하기 위한 AI 우주 전쟁! 그 험난한 여정이 시작된다...

4) 특징

다양한 난이도의 스테이지를 추가하여 스테이지 별로 다른 재미 부여
보스 맵의 보스를 물리치며 스트레스 해소 및 성취 욕구 상승

3. 게임 소재

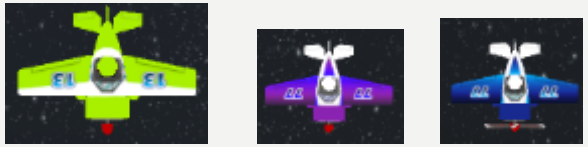
3. 게임 소재

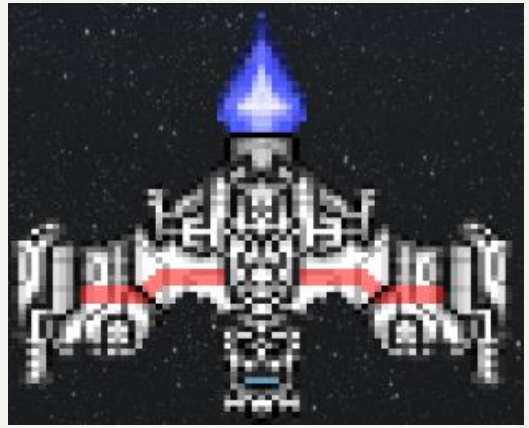
1) 캐릭터 – Kore AI

직접 플레이 할 수 있는 캐릭터로, 다양한 공격을 한다.
대한민국의 행성 차지를 위해 보내진 AI 대표 Kore가 타고 있다.

2) 적 – 경쟁국의 AI들

행성을 쟁탈하기 위해 캐릭터와 싸우며 무자비하게 공격한다.





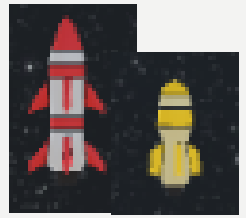
3. 게임 소재

3) 아이템

포션
플레이어가 포션을 먹으면
체력이 증가합니다.

스타
플레이어가 스타를 먹으면
일정 시간 동안 체력이 줄지 않는
무적 상태가 됩니다.

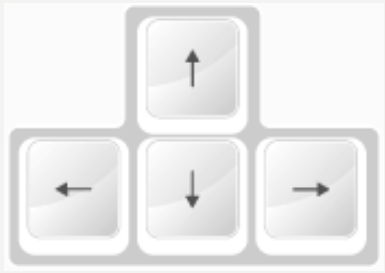
연료
플레이어가 연료를 먹으면
일반 총알보다 공격력이 2배 높은 총알이
일정 개수 만큼 생성됩니다.



4. UI 소개

4. UI 소개

1) 게임 동작 방식



방향키, WASD

캐릭터가 상하좌우로 이동합니다.

마우스 좌 클릭

캐릭터의 공격 및 화면 내 버튼들을 조작합니다.

4. UI 소개

2) 그래픽 컨셉

우주선, 전투기와 미사일, 불꽃, 전기 꽃 등 우주 그래픽 컨셉의 종 스크롤 슈팅게임

4. UI 소개

3) UI 소개

배경음

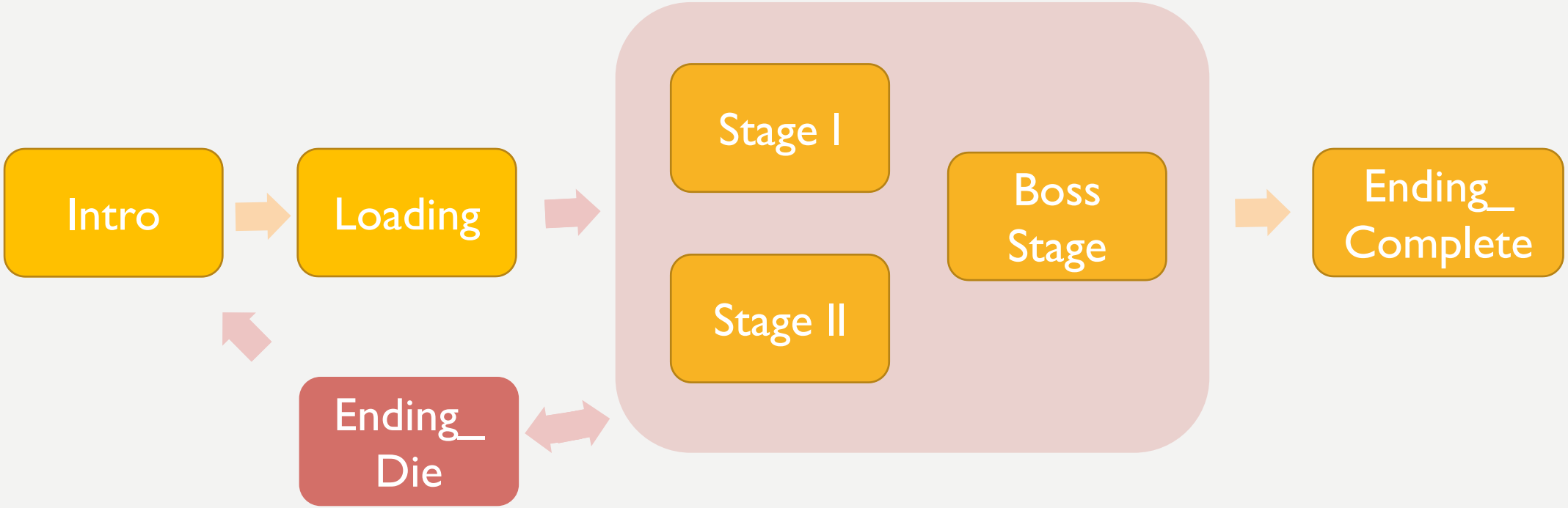
대기 중에는 잔잔하고 몽환적인, 전투 중에는 난이도 별로 다른 분위기의 노래

효과음

아이템 획득할 때, 적과 충돌 했을 때 등 다양한 상호작용에
Audio Source에 효과음을 등록하여 재생

5. 게임 프로세스

5. 게임 프로세스


Intro
Loading
Stage I
Stage II
Boss Stage
Ending_ Complete
Ending_ Die

6. 개발 순서 및 일정

6. 개발 순서 및 일정

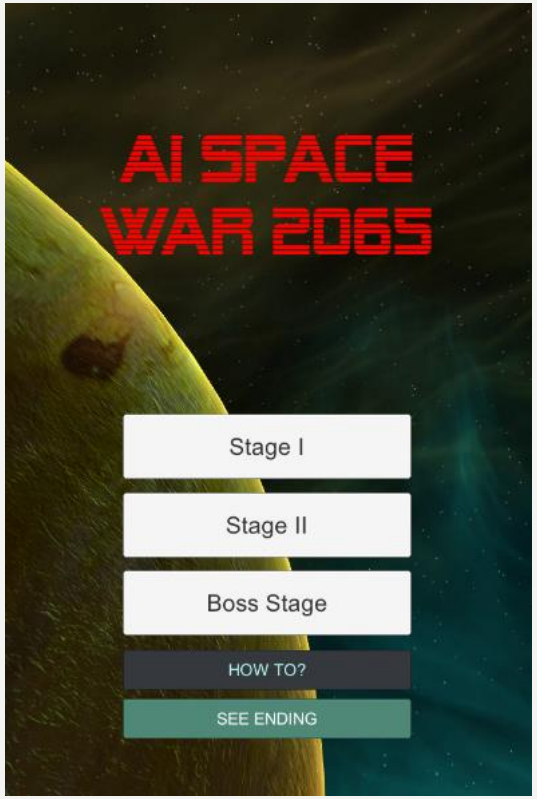
기간 : 2021.04.12 ~ 2021.04.26 (2주)

ID	활동명	시작일	종료일	기간(일)	진행 상황
1	게임의 주제 및 소재, UI 기획	4.12	4.13	2	
2	Main / Loading Scene 생성	4.14	4.15	2	
3	Stage I 구현	4.16	4.17	2	
4	Stage II 구현	4.18	4.20	3	
5	Boss Stage 구현	4.21	4.24	4	
6	Ending Scene 생성, Sound 추가	4.25	4.26	2	

7. 추가 기능

1. 씬 추가

AI SPACE WAR' S SCENE



Intro scene

원하는 씬으로 갈 수 있는
게임의 첫 화면

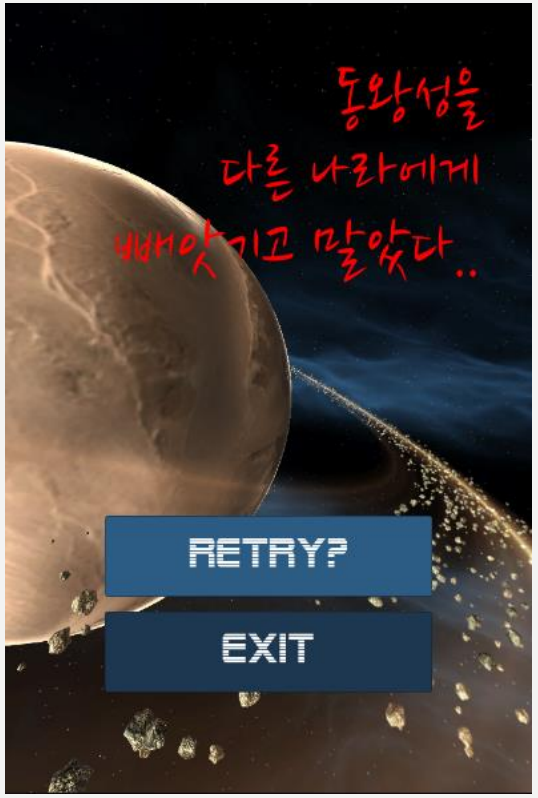
Loading scene

게임 스테이지로 가기 전
준비하는 로딩 씬

AI SPACE WAR' S SCENE

Ending_Complete scene

보스 스테이지를 완료하면 호출되는
게임의 최종 엔딩 씬



Ending_Die scene

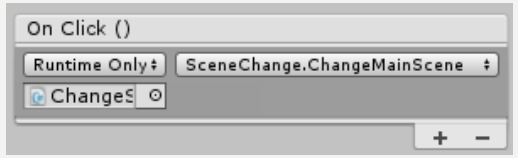
스테이지 중간에 캐릭터가 죽으면
호출되는 Die 엔딩 씬

AI SPACE WAR' S SCENE

씬 전환 방법

1) Application.LoadLevel(“Loading");
// 씬 이름

2) 버튼의 On Click() 이벤트에 함수 이름 등록



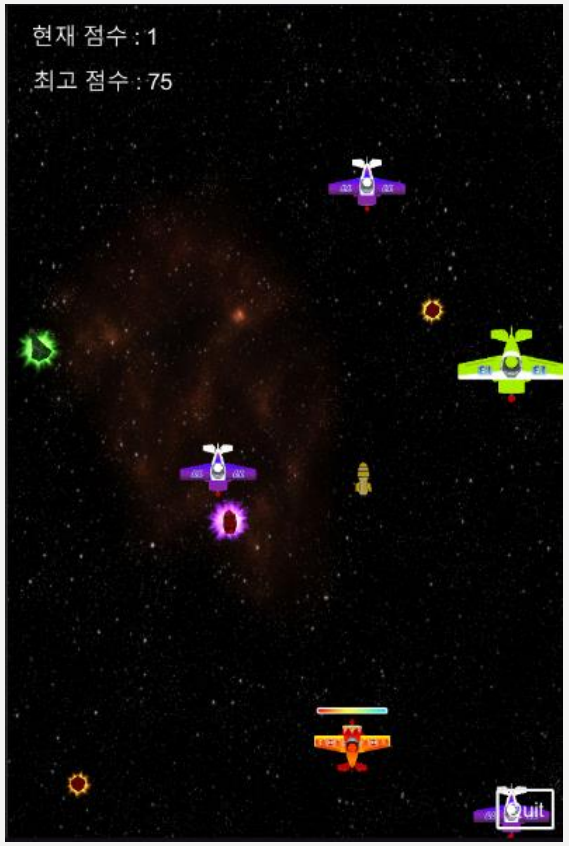

```
using UnityEngine.SceneManagement;

Unity 스크립트|참조 0개
public class SceneChange : MonoBehaviour
{
    참조 0개
    public void ChangeIntroScene()
    {
        //SoundManager.instance.BtnClickSound();
        SceneManager.LoadScene("Intro");
    }
    참조 0개
    public void ChangeStage1Scene()
    {
        SceneManager.LoadScene("Stage1");
    }
    참조 0개
    public void ChangeStage2Scene()
    {
        SceneManager.LoadScene("Stage2");
    }
    참조 0개
    public void ChangeBossStageScene()
    {
        SceneManager.LoadScene("BossStage");
    }
```

2. 스테이지 추가

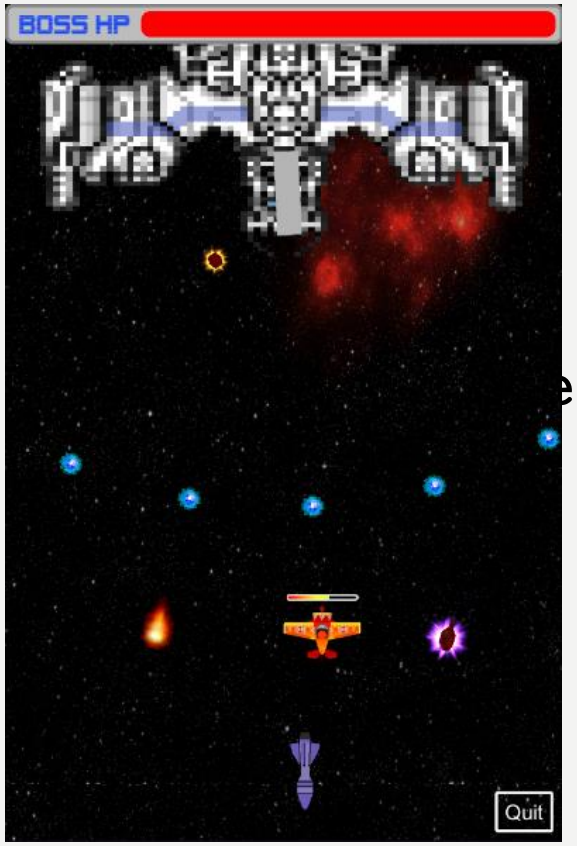
AI SPACE WAR' S STAGE

난이도별로 총 3개의 스테이지 추가

Stage1 scene



Stage2 scene



BossStage scene

3. 플레이어 제어

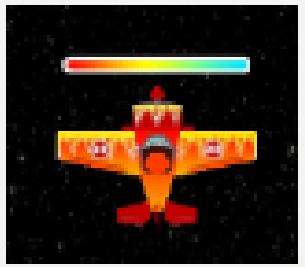
3. 플레이어 제어

```
// Update is called once per frame
Unity 메시지 | 참조 0개
void Update()
{
    Vector3 pos = Camera.main.WorldToViewportPoint(transform.position);

    if (pos.x < 0f) pos.x = 0f;  // 왼쪽 화면 밖으로나감
    if (pos.x > 1f) pos.x = 1f;  // 오른쪽 화면 밖으로 나감
    if (pos.y < 0f) pos.y = 0f;  // 아래쪽 화면 밖으로 나감
    if (pos.y > 1f) pos.y = 1f;  // 위쪽 화면 밖으로 나감

    transform.position = Camera.main.ViewportToWorldPoint(pos);

    float h = Input.GetAxis("Horizontal");
    float v = Input.GetAxis("Vertical");
    Vector3 dir = new Vector3(h, v, 0);
    transform.position += dir * speed * Time.deltaTime;
    // P = P0 + vt ( 미래위치 = 현재위치 + 속도X시간 )

    // HPBar 이동
    HPBar.transform.position = Camera.main.WorldToScreenPoint(transform.position + new Vector3(0, 0.35f, 0));
}
```

PlayerMove.cs

1. 활동 범위를 게임 화면 안으로 제한
Player가 화면 밖으로 넘어가지 않게 하기위해 Position을 World 공간에서 Viewport 공간으로 변경 후 MainCamera의 화면 밖으로 벗어나지 못하게 하고 다시 Viewport 공간을 World 공간으로 변경하여 현재 위치에 넣는다.

2. 플레이어의 체력 추가
World 공간을 Screen 공간으로 변경 후 플레이어의 머리 위 (+y축)에 위치 시키고 플레이어의 위치를 따라 다닌다.



3. 아이템

3) 아이템

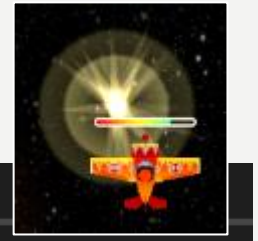
포션

플레이어가 포션을 먹으면
이펙트와 효과음이 발생하고
체력이 20 증가합니다.

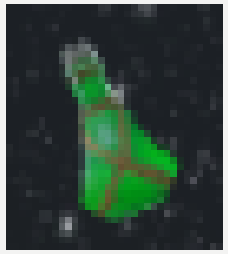
```
private void OnCollisionEnter(Collision collision)
{
    if (collision.gameObject.name.Contains("Player"))
    {
        GameObject explosion = Instantiate(explosionFactory);
        explosion.transform.position = transform.position;
        PlayerController.getChargeHP = 1;
```

```
if(getChargeHP == 1)  // 체력 충전 아이템 먹으면 체력 증가
{
    hp += 20;
    if (hp > 100)
        hp = 100;
    imgHpBar.fillAmount = (float)hp / (float)initHp;
    getChargeHP = 0;
}
```

3. 아이템



3) 아이템

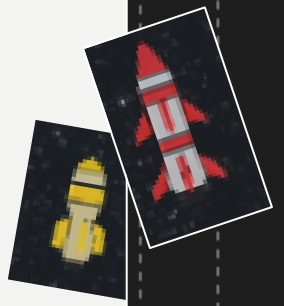


연료

플레이어가 연료를 먹으면
일반 총알보다 공격력이 2배 높은 총알이
일정 개수 만큼 생성됩니다.
이펙트와 효과음도 발생

```
private void OnCollisionEnter(Collision collision)
{
    if (collision.gameObject.name.Contains("Player")) //Player
    {
        GameObject explosion = Instantiate(explosionFactory);
        explosion.transform.position = transform.position;
        PlayerFire.setBullet2 = 1;
```

```
else if (setBullet2 == 1)  // 공격력 증가
{
    if (cnt < poolSize2) {
        if (bulletObjectPool2.Count > 0) //오브젝트풀에 총알
        {
            print(cnt);
            GameObject bullet2 = bulletObjectPool2[0]; // 비
            bullet2.SetActive(true); // 활성화
            bulletObjectPool2.Remove(bullet2); // 오브젝트 풀
            bullet2.transform.position = transform.position;
            cnt++;
        }
    }
```



3. 아이템

3) 아이템

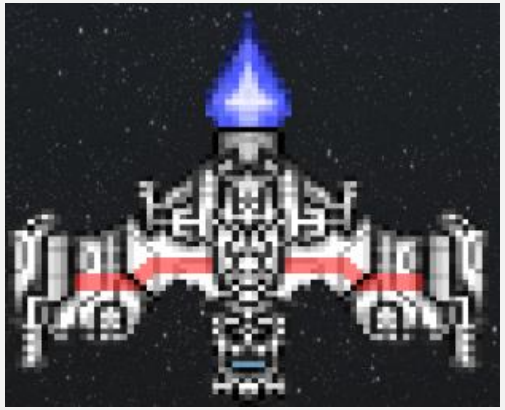
스타

플레이어가 스타를 먹으면
적이나 공격에 맞아도
일정 시간 동안 체력이 줄지 않는
무적 상태가 됩니다.
이펙트와 효과음도 발생

```
private void OnCollisionEnter(Collision collision)
{
    if (collision.gameObject.name.Contains("Player")) //Player
    {
        GameObject explosion = Instantiate(explosionFactory);
        explosion2 = Instantiate(explosionFactory2);
        explosion.transform.position = transform.position;
        PlayerController.getItemStar = 1;
```

```
if(getItemStar == 1)  // 무적 아이템 먹으면 몇초간 무적
{
    if (starTime > 0.0f)  // 무적 상태
    {
        starTime -= Time.deltaTime;
        Plane1.GetComponent<MeshRenderer>().material = Material2;  // 매터
        GameObject.Find("Background").GetComponent<AudioSource>().Stop();
    }
    else
    {
        Plane1.GetComponent<MeshRenderer>().material = Material;
        GameObject.Find("Background").GetComponent<AudioSource>().Play();
        Item_Star.explosion2.SetActive(false);
        getItemStar = 0;
        starTime = 5.0f;
    }
}
if(isDamagedByFlare == 1)
{
    if (getItemStar == 1)  // 무적이면 체력 유지
        isDamagedByFlare = 0;
```

1-1) 보스 추가

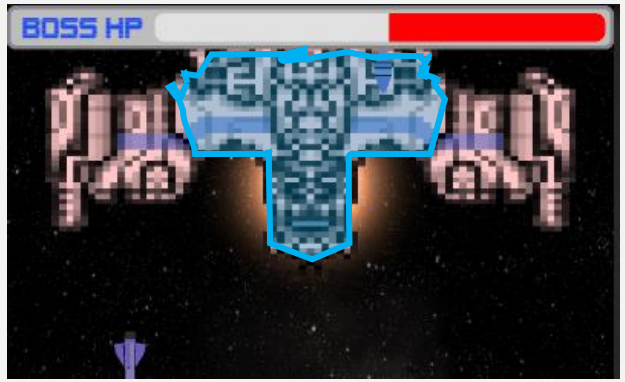


BOSS

난이도 조절과 게임의 성취감, 스트레스 해소를 위해
고난이도 보스 맵을 만들었습니다.

다양한 공격 패턴과 미사일 생성 패턴으로 난이도를 조절하였습니다.

1-1) 보스 추가

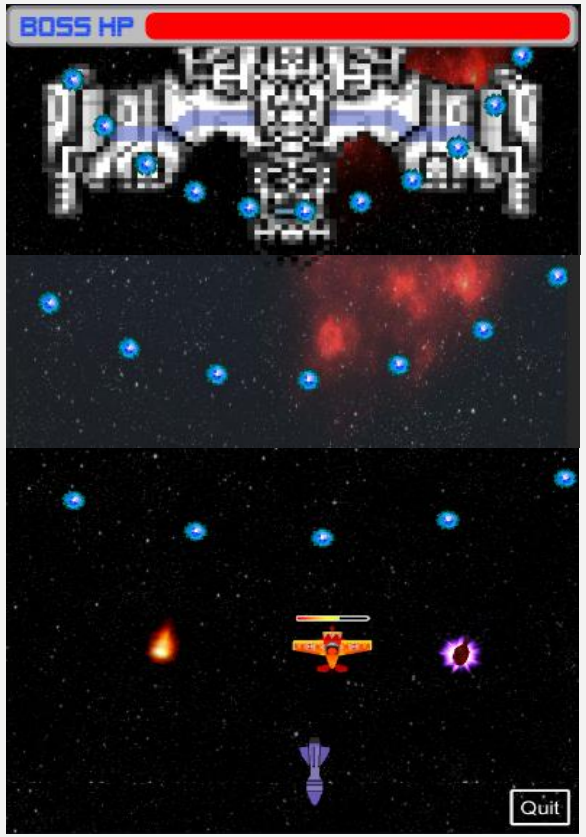


체력에 따른 보스의 상태 비교 사진

체력이 어느 수치 이하가 되면 SpriteRenderer 컴포넌트를 얻어와 color 속성을 바꿔주었고, particle 효과도 추가하였습니다.

보스의 헤드(급소)가 플레이어의 총알에 맞으면 체력 감소

1-1) 보스 추가



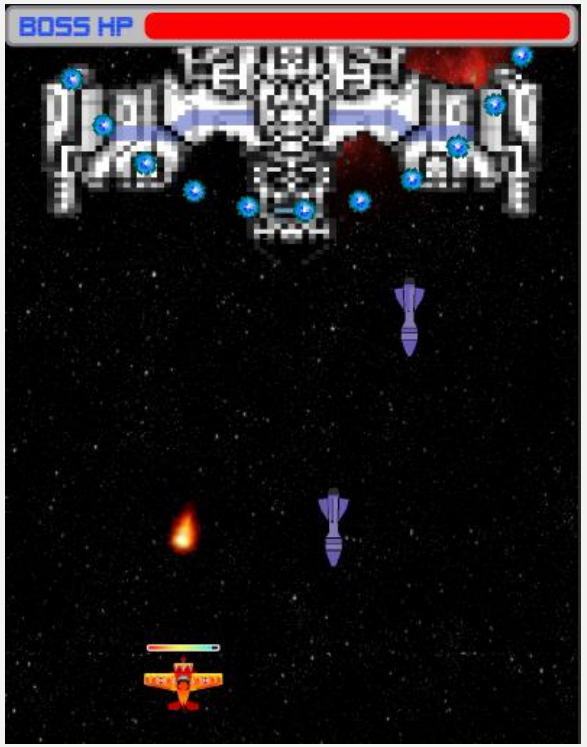

```
public GameObject bullet;
Unity 메시지 | 참조 0개
private void Start()
{
    StartCoroutine(cntTime2(3.0f));
}
참조 2개
IEnumerator cntTime2(float delayT)
{
    float randomTime = Random.Range(3f, 6f);

    shot();
    yield return new WaitForSeconds(delayT);
    StartCoroutine(cntTime2(randomTime));
}
참조 1개
void shot()
{
    for (int i = 0; i < 360; i += 13)
    {
        GameObject temp = Instantiate(bullet); //총알 생성
        Destroy(temp, 2f); //2초마다 삭제
        temp.transform.position = new Vector2(0f, 5.5f); //총알 생성 위치
        temp.transform.rotation = Quaternion.Euler(0, 0, i); //Z에 값이 변해야 회전
    }
}
```

반복적인 공격 패턴을 위해 코루틴을 사용했습니다.
3~6초에 1번 씩 원형 전기꽃을 발사합니다.

총알 생성위치를 정하고 Z 값변화를 주어
특정 시간마다 반복되는 원형 공격 패턴을 구현하였습니다.

1-1) 보스 추가

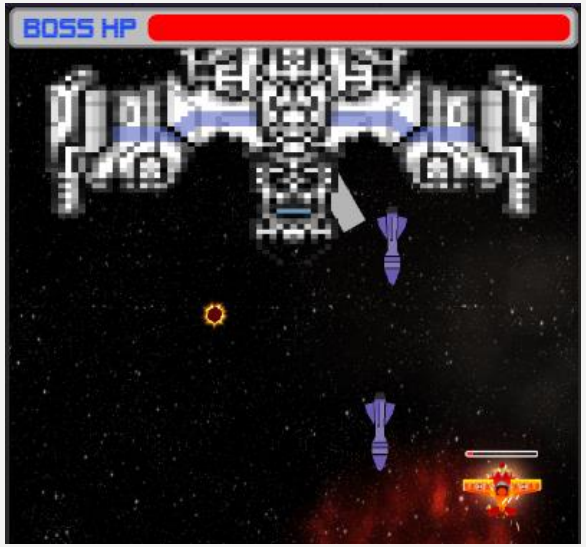
```
private void Start()
{
    StartCoroutine(cntTime(3.0f));
}
참조 2개
IEnumerator cntTime(float delayT)
{
    float randomTime = Random.Range(3f, 6f);
    var temp = Instantiate(bullet);

    //총알 생성 위치를 머즐 입구로 한다.
    temp.transform.position = pos.position;

    //총알의 방향을 Center의 방향으로 한다.
    //->참조된 Center오브젝트가 Target을 바라보고 있으므로, Rotation이 방향이 됨.
    temp.transform.rotation = Center.rotation;

    yield return new WaitForSeconds(delayT);
    StartCoroutine(cntTime(randomTime));
}
```

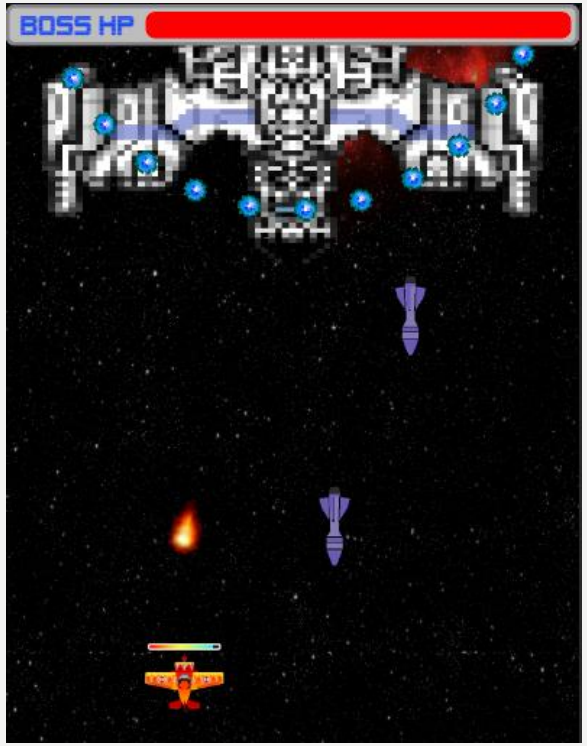


반복적인 공격 패턴을 위해 코루틴을 사용했습니다.
3~6초에 1번 씩 화염을 발사합니다.

총구의 입구가 타겟을 향하고 있으므로 그 방향으로 위치를 정해서
특정 시간마다 반복되는 타겟을 따라다니는 공격 패턴을 구현하였습니다.

1-1) 보스 추가





보라색 미사일

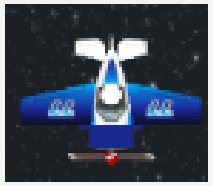
랜덤 시간마다 생성되어 맵을 활보하고 다녀
플레이어의 진로를 방해하고 공격합니다.

플레이어의 총알에 맞으면 자폭합니다.

1-2) 적 비행기 추가

Strong Enemy

일반 비행기보다 2배 강합니다.
내구도에 따라 크기가 달라집니다.

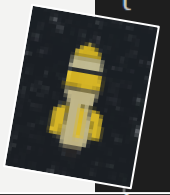


Enemy, Enemy2

생성패턴과 속도, 방향이 서로 다른
일반 비행기들 입니다.

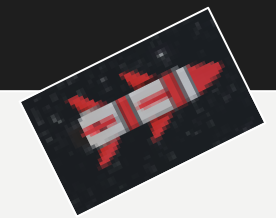
```
void OnEnable() // 객체가 활성화 될 때 호출되는 함수 //OnD
{
    durability = 2;  //Strong Enemy 내구도
    transform.localScale = new Vector3(0.7f, 0.7f, 0.7f);
```

```
if (collision.gameObject.name.Contains("Bullet"))
{
    durability -= 1;  // 내구도 -1
    GameObject vsB = Instantiate(vsBullet);
    vsB.transform.position = transform.position;
    collision.gameObject.SetActive(false);
```



```
if (durability == 1)
{
    transform.localScale = new Vector3(0.5f, 0.5f, 0.5f);
    return;
}
```

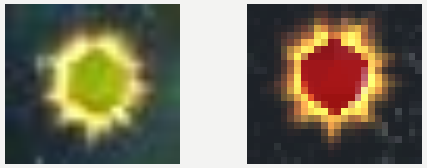
```
else if (collision.gameObject.name.Contains("newBull"))
{
    durability -= 2;  // 내구도 -2
```



```
if (isDamaged == 1)
    hp -= 25;
else
    hp -= 40;
```

내구도가 강할 때는
플레이어의 체력이 더 많이 줄어듭니다.

1-2) 공격 패턴 추가



FlareObstacle

랜덤 시간마다 생성되어 맵을 활보하고 다녀 플레이어의 진로를 방해하고 공격합니다.

색상 깜빡임을 위해 코루틴을 사용하였습니다.

```
private void Start()
{
    Flare = transform.GetChild(0).gameObject;
}
참조 1개
IEnumerator timer()
{
    int time = 0;
    WaitForSeconds wfs = new WaitForSeconds(0.2f);
    while (true)
    {
        time++;
        //print("time " + time);
        if (time % 2 == 0)
            Flare.GetComponent<MeshRenderer>().material = Material;
        else
            Flare.GetComponent<MeshRenderer>().material = Material2;

        yield return wfs;
    }
}
```

```
void Update()
{
    StartCoroutine(timer());
    transform.position += dir * speed * Time.deltaTime;
}
Unity 메시지 | 참조 0개
private void OnCollisionEnter(Collision collision)
{
    if (collision.gameObject.name.Contains("Player")) //Playe
    {
        GameObject explosion = Instantiate(explosionFactory);
        explosion.transform.position = transform.position;
        PlayerController.isDamagedByFlare = 1;
```