캡스턴디자인결과보고서

컴퓨터공학과 캡스톤디자인 결과 보고서				
팀명칭	Retro			
설계명	3D Shooting Game			
지도교수	최의인			
팀구성	학번	이름	연락처	업무
팀장	20150711	김승민	01085662181	유니티 게임 개발
팀원	20150642	정성철	01044409333	유니티 게임 개발
팀원	20192966	이희찬	01033886486	총괄, 유니티 게임 개발
팀원				
팀원				
팀원				
개발기간	시작일 2022년 03월 07일 ~ 종료일 2022년 6월 12일			

붙임. 결과보고서

2022학년도 1학기 캡스턴디자인 결과보고서를 제출합니다.
2022년 6월 16일

캡스톤 디자인 결과보고서

1. 수행 과제 개요

1.1 과제 개발 요약

옛날 오락실에서 즐겨하던 추억의 게임 '1942'를 모티브를 삼아 최근 스팀이라는 플랫폼에서 심플함과 고전게임느낌으로 많은 사람들의 인기를 얻은 'Vampire Survivors'처럼 쉽고 단순하게 즐길수 있는 슈팅게임에서 3D를 적용시킨 3D 탱크게임을 만들어 보고싶어 개발을 진행하였습니다.

1.2 과제 개발의 필요성

(1) 교육적 필요성

--

(2) 기술개발적 필요성

1.플레이어 구현

아동스크립트를 생성하고 할당하여 Player object가 방향키에 따라 움직이게 설정합니다.

2.포신 컨트롤

포신을 컨트롤 할 수 있는 스크립트를 만들어 마우스로 움직일 수 있게 제작했습니다.

3.총알 제작 및 발사

총알 오브젝트를 제작하기 위해 총알 컨트롤 스크립트 생성 후 총알이 중력을 받아 떨어지게 벡터 함수를 사용하여 제작했습니다.

4. 적 생성과 이동

적 생성 스크립트와 적 이동 함수, 적이 플레이어를 공격하게 제작했습니다.

5. 중력 적용

본인 캐릭터와 적 캐릭터에 유니티 자체 중력을 적용시켰습니다.

6. 적 HP설정 및 파괴

Hp를 설정하여 그 이하로 내려가면 총알에 맞은 적을 파괴합니다.

(3) 상업적 필요성

PC 뿐만 아니라 모바일, 콘솔 등 다양한 플랫폼에서 게임 설치 가격을 매겨 판매합니다.

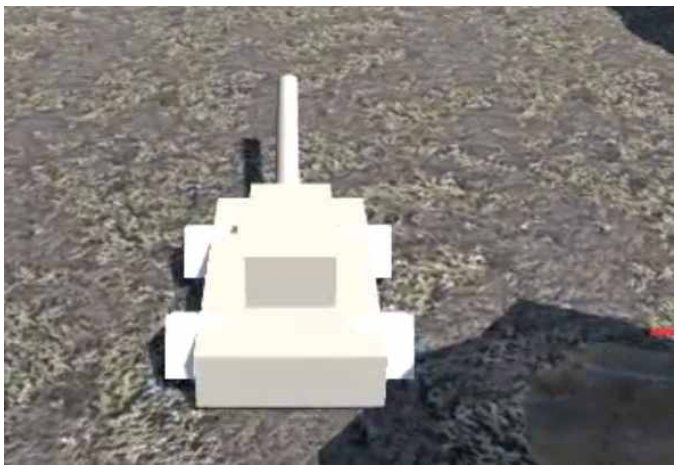
(4) 기타 필요성

--

1.3 개발 목표

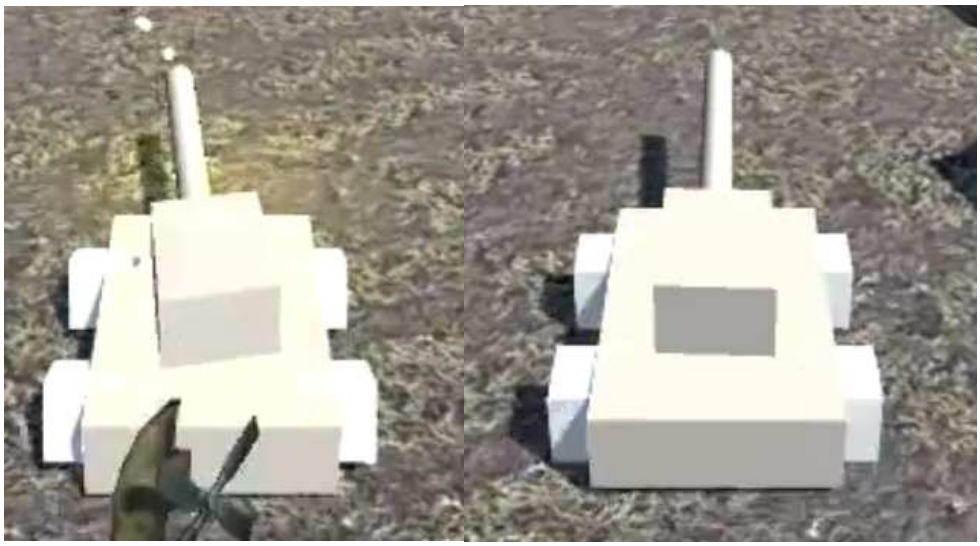
1.플레이어 구현

아동스크립트를 생성하고 할당하여 Player object가 방향키에 따라 움직이게 설정합니다.



2.포신 컨트롤

포신을 컨트롤 할 수 있는 스크립트를 만들어 마우스로 움직일 수 있게 제작했습니다.



3.중력을 적용한 총알 제작 및 발사

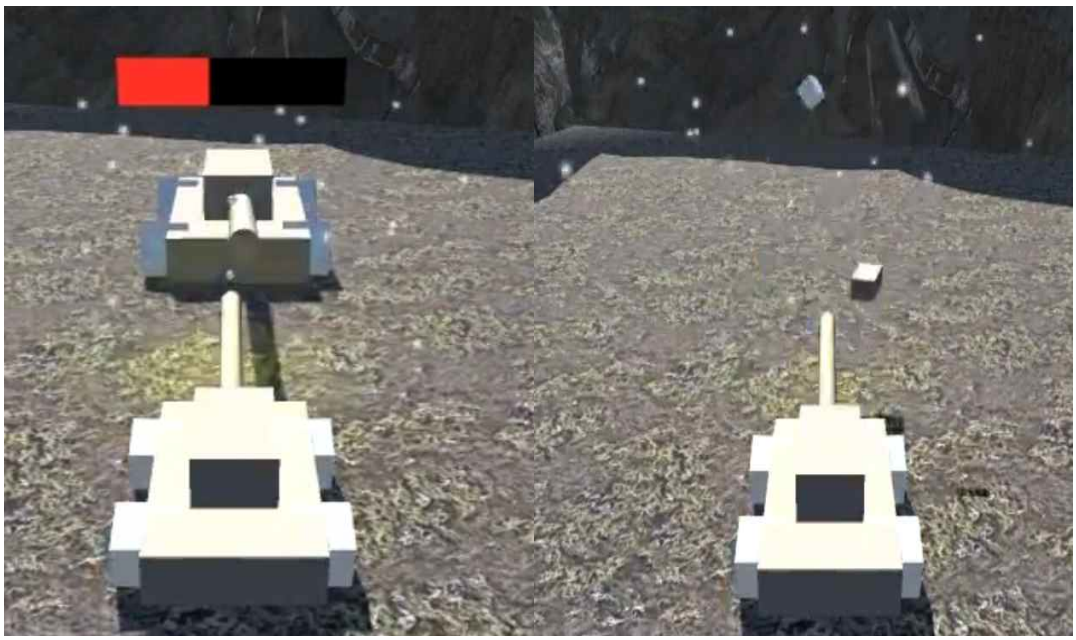
총알 오브젝트를 제작하기 위해 총알 컨트롤 스크립트 생성 후 총알이 중력을 받아 떨어지게 제작했습니다.

4. 적 생성과 이동

적 생성 스크립트와 적 랜덤 이동 함수, 적이 플레이어를 공격하게 제작했습니다.

5. 적 HP설정 및 파괴

Hp를 설정하여 그 이하로 내려가면 총알에 맞은 적을 파괴합니다.



1.4 과제 수행 내용, 범위 및 방법

(1) 과제의 내용과 범위

1.플레이어 구현

아동스크립트를 생성하고 할당하여 Player object가 방향키에 따라 움직이게 설정합니다.

2.포신 컨트롤

포신을 컨트롤 할 수 있는 스크립트를 만들어 마우스로 움직일 수 있게 제작했습니다.

3.총알 제작 및 발사

총알 오브젝트를 제작하기 위해 총알 컨트롤 스크립트 생성 후 총알이 중력을 받아 떨어지게 벡터 함수를 사용하여 제작했습니다.

4. 적 생성과 이동

적 생성 스크립트와 적 랜덤 이동 함수, 적이 플레이어를 공격하게 제작했습니다.

5. 중력 적용

본인 캐릭터와 적 캐릭터에 유니티 자체 중력을 적용시켰습니다.

6. 적 HP설정 및 파괴

Hp를 설정하여 그 이하로 내려가면 총알에 맞은 적을 파괴합니다.

(2) 과제내용별 참여학생 간의 역할 분담

김승민 - 유니티 게임 개발

정성철 - 유니티 게임 개발

이희찬 - 총괄, 유니티 게임 개발

1.5 개발 과제 활용 및 파급효과

(1) 산업적, 기술적 파급효과(시장성 등)

PC 뿐만 아니라 모바일, 콘솔 등 다양한 플랫폼에서 게임 설치 가격을 매겨 판매할 수 있습니다. 또한 Unity Ads를 통해 광고 수익을 얻을 수 있습니다.

(2) 최종결과물의 활용방안

게임 시작 인터페이스 구축, 스테이지 수의 증가, 그래픽의 업그레이드가 필요합니다.

2. 과제 개발 결과

2.1 과제 계획서 (결과)

(1) 참여자별 활동

김승민 - 유니티 게임 개발 정성철 - 유니티 게임 개발 이희찬 - 총괄, 유니티 게임 개발

(2) 활동(업무)의 정의

유니티 게임 개발 - C# 코드 작성 총괄 - 전체적인 코드 점검 및 역할 분담

(3) 활동 추진 시간 계획 (스케줄링)

	3월				4월				5월				6월	
	1주	2주	3주	4주	1주	2주	3주	4주	1주	2주	3주	4주	1주	2주
아이디어 구상														
자료 수집														
디자인														
코딩														
수정 및 보완														

(4) 예산 사용 (지원금이 있는 경우)

--

(5) 개발환경

- 개발 플랫폼 (하드웨어 포함): 컴퓨터
- 소프트웨어: C#, Unity

2.2 요구사항 분석서

(1) 요구사항 도출 방법 (자체 요구사항, 고객의 앙케이드, 웹으로부터 조사 등)

- 자체 요구사항, 웹으로부터 조사

(2) 기능적 요구사항 목록

1.플레이어 구현

아동스크립트를 생성하고 할당하여 Player object가 방향키에 따라 움직이게 설정합니다.

2.포신 컨트롤

포신을 컨트롤 할 수 있는 스크립트를 만들어 마우스로 움직일 수 있게 제작했습니다.

3.총알 제작 및 발사

총알 오브젝트를 제작하기 위해 총알 컨트롤 스크립트 생성 후 총알이 중력을 받아 떨어지게 벡터 함수를 사용하여 제작했습니다.

4. 적 생성과 이동

적 생성 스크립트와 적 랜덤 이동 함수, 적이 플레이어를 공격하게 제작했습니다.

5. 중력 적용

본인 캐릭터와 적 캐릭터에 중력을 적용시켰습니다.

6. 적 HP설정 및 파괴
Hp를 설정하여 그 이하로 내려가면 총알에 맞은 적을 파괴합니다.

(3) 비기능적 요구사항 목록 (보안, 성능 등)
네트워크보안
-정보 유출 방지
-ID , PW
-부정 접속 방지
-서버 컴퓨터 해킹, DB 공격
-개발자 컴퓨터 해킹
버그 발견 및 수정

(4) 기 개발된 유사제품이나 논문, 기술 등 조사
Unity 3D 게임 엔진을 이용한 슈팅 게임 설계 및 개발
-배재환

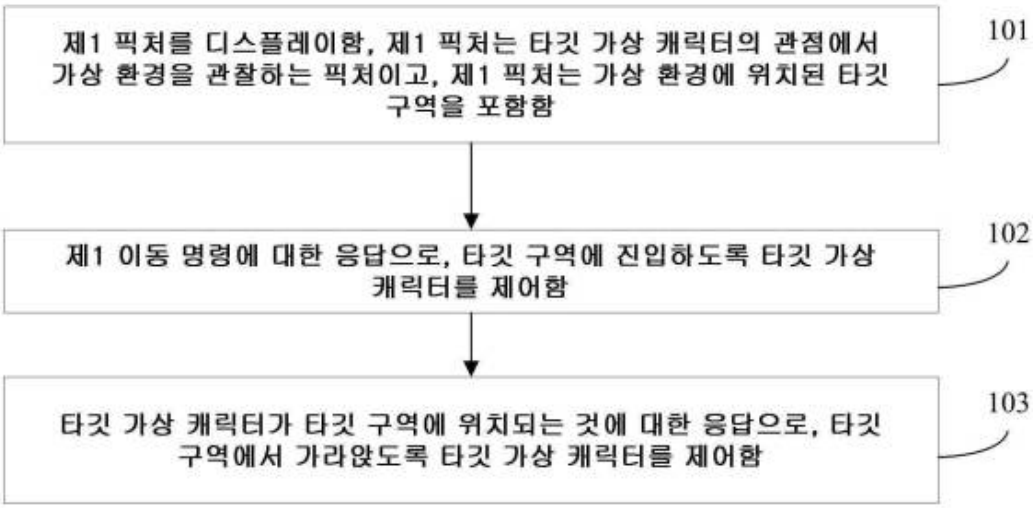
Unity 3D는 여러 플랫폼을 지원하는 2D 및 3D 게임 엔진입니다. 편집자는 Windows, Mac, Android, iPhone 및 다양한 웹 플랫폼에서 실행되는 2D 및 3D 게임을 개발할 수 있습니다. 본 논문에서는 Unity 3D 게임 엔진을 이용하여 설계 및 개발한 슈팅 게임을 제안한다. 슈터는 총이나 미사일을 쏘아 적을 무찌르는 게임 장르입니다. 게임 시스템과 같은 요소를 통해 그것은 매우 간단하며, 순간적인 성찰 능력이나 행동은 섹스 게임을 추구하는 것이 특징입니다. 또한 게임 개발을 위한 게임 및 게임 시스템 아키텍처를 분석하였다. 다양한 플랫폼 기반의 게임 디자인과 개발을 통해 이들에게 도움이 되었으면 합니다.

유니티를 이용한 탱크 게임 설계 및 구현
-이경호;임대영

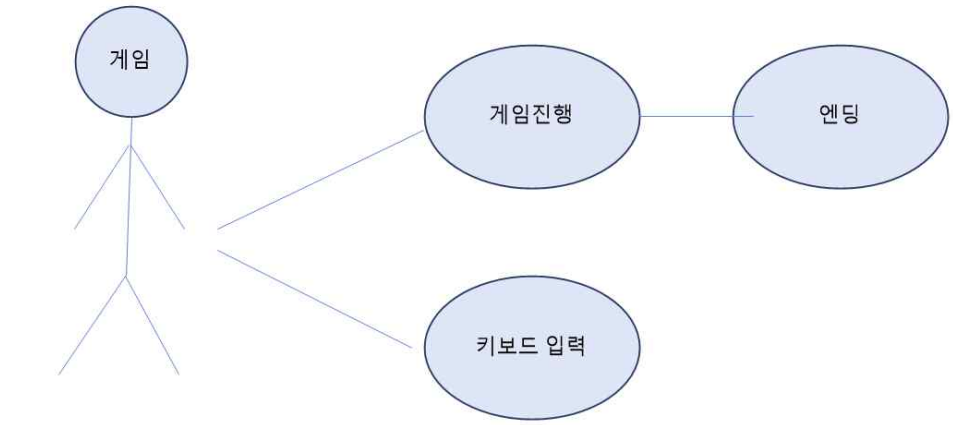
탱크 게임에 필요한 요구사항을 분석하고, 만들어진 요구사항 명세를 이용하여 설계하고, 개발하였다. 본 게임은 여러 사람들과 실시간으로 대전이 가능한 대전 모드와, 혼자서도 플레이가 가능한 싱글 모드도 지원한다. 구성은 탱크, 아이템의 구입과 선택이 가능한 로비 화면과 게임 플레이가 가능한 게임 화면으로 이루어져 수행된다.

(5) 관련특허 조사
본 출원은 가상 환경에서 가상 캐릭터를 제어하기 위한 방법 및 장치, 디바이스 그리고 매체를 개시하며, 컴퓨터 기술들의 분야에 관한 것이다. 이 방법은: 제1 픽처를 디스플레이하는 단계 — 제1 픽처는 타깃 가상 캐릭터의 관점에서 가상 환경을 관찰하는 픽처이고, 제1 픽처는 가상 환경에 위치된 타깃 구역을 포함함 —; 제1 이동 명령에 대한 응답으로, 타깃 구역에 진입하도록 타깃 가상 캐릭터를 제어하는 단계; 및 타깃 가상 캐릭터가 타깃구역에

위치되는 것에 대한 응답으로, 타깃 구역에서 가라앉도록 타깃 가상 캐릭터를 제어하는 단계를 포함한다. 이 방법은 지형 시뮬레이션 현실성을 개선하고 상호 작용 비율의 증가를 가능하게 한다.



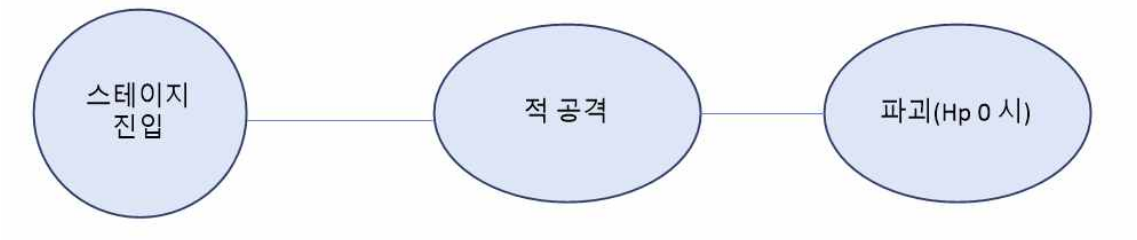
(6) 유스케이스다이어그램과 시나리오를 통해 요구사항을 표현
Use Cases Diagram



Scenario

2.3 설계서

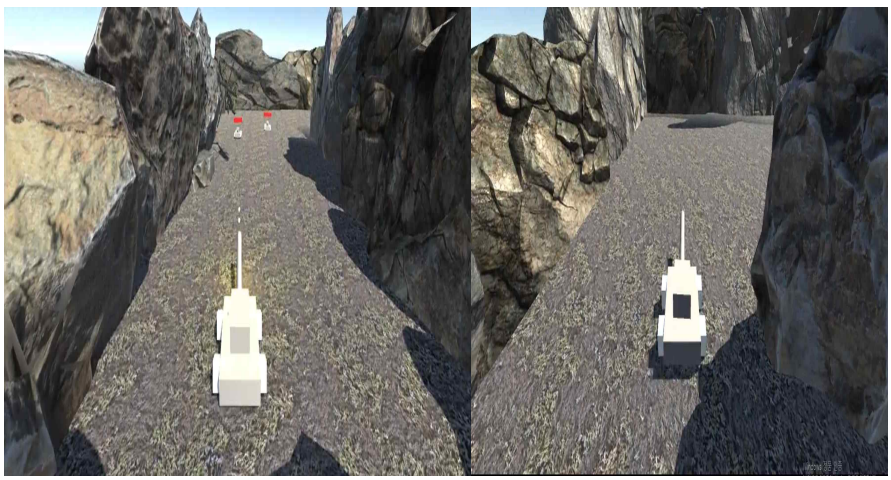
(1) 구조설계 (통신, 인터페이스 포함)

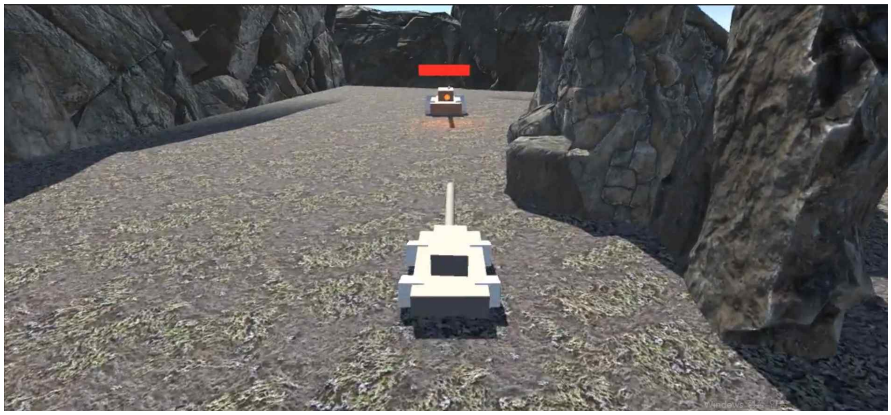


(2)상세 설계

--

(3) 화면설계(GUI 등)





(4) 스키마 설계 (데이터베이스 부분)

--

2.4 구현(코딩 등) 결과

(1) 소스코드 (필요시)

```
using UnityEngine.UI;

public class tankController : MonoBehaviour
{
    public GameObject top = null;
    public GameObject cannonRoot = null;
    public GameObject wheel01 = null;
    public GameObject wheel02 = null;
    public GameObject wheel03 = null;
    public GameObject wheel04 = null;

    public float speed = 5;
    public float rotSpeed = 120;
    public float rotSpeedTop = 200;
    public float rotSpeedCannon = 200;

    public float bullet_power = 600;

    bool bCannonMov = false;

    public GameObject bullet;
    public GameObject missile;
    public GameObject muzzle;

    float curHP = 1000f;
    float maxHP = 1000f;

    float timePassedMissile = 0;


    void Start()
    {

        GameObject.Find("Canvas_Player_HP").transform.Find("ImageHP").GetComponent<Image>().fillAmount = 1f;
    }


    void Update()
    {
        if(Input.GetMouseButtonDown(0))
            bCannonMov = true;
        if (Input.GetMouseButtonUp(0))
            bCannonMov = false;


        float front_Key = Input.GetAxis("Vertical");
        float rot_Key = Input.GetAxis("Horizontal");

        float Horizontal_Mouse = Input.GetAxis("Mouse X");
        float moveVertical_Mouse = Input.GetAxis("Mouse Y");

        float amtToMove = speed * Time.deltaTime;
        float amtToRot = rotSpeed * Time.deltaTime;
        float amtToRotTop = rotSpeedTop * Time.deltaTime;
        float amtToRotCannon = rotSpeedCannon * Time.deltaTime;

        //Debug.LogFormat("{0}", amtToRotTop);


        transform.Translate(Vector3.forward * front_Key * amtToMove);
```

<플레이어 구현>

```
if (bCannonMov)
{

    top.transform.localRotation *= Quaternion.Euler(0, Horizontal_Mouse * amtToRotTop, 0);


    Quaternion qtDelta = Quaternion.Euler(0, 0, moveVertical_Mouse * amtToRotCannon);
    Quaternion qtOld = cannonRoot.transform.localRotation;
    Quaternion qtNow = qtOld * qtDelta;


    float zAngleLocal = qtNow.eulerAngles.z;

    if (!(zAngleLocal > 20 && zAngleLocal < 340))

        cannonRoot.transform.localRotation *= Quaternion.Euler(0, 0, moveVertical_Mouse * amtToRotCannon);

}
```

<포신 구현>

```
93      if (Input.GetButtonDown("Fire1"))
94      {
95
96          var myBullet = objectPool.getBullet();
97          if (myBullet == null)
98              return;
99
100         myBullet.SetActive(true);
101         myBullet.transform.position = muzzle.transform.position + 2 * muzzle.transform.forward;
102         myBullet.transform.rotation = muzzle.transform.rotation;
103
104         Rigidbody rb = myBullet.GetComponent<Rigidbody>();
105         rb.AddForce(myBullet.transform.forward * bullet_power);
106
107
108         Light pointLight = transform.Find("PointLight").GetComponent<Light>();
109         pointLight.intensity = 3;
110
111         StartCoroutine("turnOffLight");
112     }
113
114
115     timePassedMissile += Time.deltaTime;
116
117     if ( timePassedMissile>1.2f &&
118         Input.GetButtonDown("Fire2"))
119     {
120         timePassedMissile = 0;
121
122
123         GameObject[] enemyTanks = GameObject.FindGameObjectsWithTag("EnemyTank");
124         if (enemyTanks.Length > 0)
125         {
126             var myMissile = objectPool.getMissile();
127             myMissile.SetActive(true);
128             myMissile.transform.position = muzzle.transform.position + 2 * muzzle.transform.forward;
129             myMissile.transform.rotation = muzzle.transform.rotation;
130
131
132             myMissile.GetComponent<homingMissile>().targetObj = enemyTanks[0];
133             myMissile.SetActive(true);
134
135
136             Light pointLight = transform.Find("PointLight").GetComponent<Light>();
137             pointLight.intensity = 3;
138
139             StartCoroutine("turnOffLight");
140         }
141
142     }
143
144
```

<총알 구현>

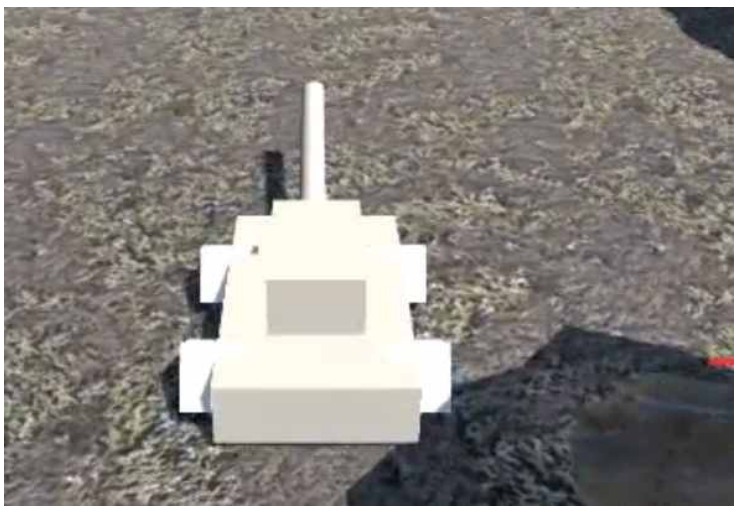
```
public class homingMissile : MonoBehaviour
{
    float time = 0;

    public ParticleSystem bulletParticle;

    public GameObject targetObj;

    Rigidbody rb;

    Vector3 vDirToTarget_Pre;

    float velocity = 20.0f;
    float accel = 15.0f;

    private void Awake()
    {
        rb = gameObject.GetComponent<Rigidbody>();
    }

    void Start()
    {
        vDirToTarget_Pre = targetObj.transform.position - transform.position;
    }


    void Update()
    {
        Vector3 vDirToTarget;
        if (targetObj != null)
            vDirToTarget = targetObj.transform.position - transform.position;
        else
            vDirToTarget = vDirToTarget_Pre;

        vDirToTarget_Pre = vDirToTarget;


        if (vDirToTarget.magnitude < 1.0) return;


        {

            Vector3 vDirInterpolate =
                transform.position + transform.forward + 0.05f * Time.deltaTime * (vDirToTarget - transform.forward);

            transform.LookAt(vDirInterpolate);


            velocity += accel * Time.deltaTime;
            Vector3 vMov = velocity * Time.deltaTime * transform.forward;
            transform.Translate(vMov.x, vMov.y, vMov.z, Space.World);
        }


        time += Time.deltaTime;
        if (time > 10.0f)
        {

            gameObject.SetActive(false);

        }
    }

    private void OnCollisionEnter(Collision collision)
    {
        ContactPoint pt = collision.GetContact(0);
        var myParticle = Instantiate(bulletParticle, pt.point + 2 * Vector3.up, Quaternion.identity); // 2 = 약간 위

        if (collision.collider.gameObject.tag == "EnemyTank")
        {
            collision.collider.gameObject.GetComponent<AIController>().takeDamage(100.0f);
        }


        gameObject.SetActive(false);

    }


    private void OnEnable()
    {

        rb.WakeUp();
    }
    private void OnDisable()
    {
        CancelInvoke();


        rb.Sleep();
        rb.velocity = Vector3.zero;


        time = 0;
        velocity = 20.0f;
    }
```

<유도미사일구현>

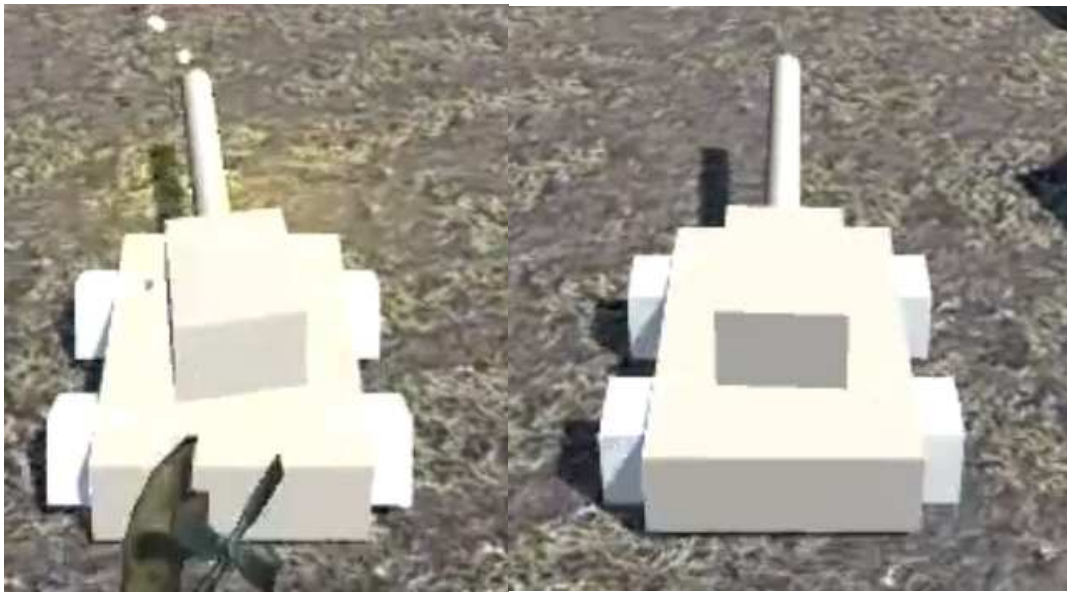
```
void movHelicopter()
{
    act += Time.deltaTime;

    if (points.Count > 0 && act > 0.05f)
    {
        act = 0;
        index = index % points.Count;
        hellicopter.transform.position = points[index];



        int indexNext;
        if (index + 1 < points.Count)
            indexNext = index + 1;
        else
            indexNext = (index + 1) % points.Count;

        hellicopter.transform.LookAt(points[indexNext]);




        Vector3 curDir = Vector3.zero;
        if (index + 5 < points.Count)
            curDir = points[index + 5] - points[index];
        else
            curDir = points[(index + 5) % points.Count] - points[index];


        int preIndex;
        if (index < 10)
            preIndex = (points.Count) - (10 - index);
        else
            preIndex = index - 10;

        Vector3 pre = points[preIndex];


        int nextIndex;
        if (index >= points.Count - 10)
            nextIndex = (index + 10) - (points.Count);
        else
            nextIndex = index + 10;

        Vector3 next = points[nextIndex];



        Vector3 cur = points[index];
        Vector3 dir1 = cur - pre;
        Vector3 dir2 = next - cur;

        Vector3 cross = Vector3.Cross(dir1.normalized, dir2.normalized);
```

<헬리콥터 구현>

(2) 구현한 시스템 (필요시): 실물, 사진, 동영상 포함



<플레이어 구현>



〈포신 구현〉

〈헬리콥터 구현〉 〈총알 구현〉

(3) 기타 결과물

--

2.5 시험결과

(1) 요구사항에대한 시험사례

--

(2) 요구사항에대한 시험결과

--

2.6 자체평가

아이디어를 생각하고 팀원들의 조율을 맞추는 1개월을 제외한 2개월, 거기다 다른 강의도 있었던 만큼 많은 진행을 하지 못했고, 부족한 점이 많은 것 같습니다. 개선점으로는 스테이지 수의 증가, 사용자 인터페이스 구축, 그래픽 등 이 필요하며, 상업성을 위해선 다른 게임들과의 개성 차이가 필요합니다.

3. 과제수행을 통해 느낀점 및 향후 연구개발내용

3.1 느낀점 (팀원별로 작성, 필욧;팀원 얼굴 사진)

김승민 : 조장으로서 책임감있게 리드하려고 노력했고, 배우는 것도 많았으나 화합면에서 많이 부족함을 느꼈습니다. 팀과제 수행하며, 프로젝트 진행, 코딩면에서 배우는게 많았습니다.

정성철 : 팀프로젝트 수행중에 많은 의견충돌이 있었지만 이번기회로 c#언어에 대해 공부하고 적용하여 팀과제 수행을 해봤고 취업할때 포트폴리오로 활용할수 있다는 점이 좋았습니다.

이희찬 : 아직 넣어보고 싶은 기능도 많고 해보고 싶은 것도 많은데 시간이 부족한 게 아쉬웠습니다. 나중에 혼자라도 개발을 하여 제대로 완성된 게임을 만들어 보고 싶었습니다.

3.2 향후 연구개발 내용

스테이지 수의 증가, 사용자 인터페이스 구축, 그래픽 등
다른 게임과의 개성차이를 위한 유니크 포인트 개발