[ 개발 문서 ]

Defence Game

프로젝트 진행 인원 : 1인

프로젝트 기간 : 20.07.17. ~ 20.07.23.

작성자 : 김예슬

목차

**1. 프로젝트 진행 2**

1) 일정 2

2) 개발 환경 2

**2. 상세 설계 3**

1) 최초 설계 3

2) 최종 설계 4

**3. 프로젝트 흐름 6**

1) 순서도 6

2) UML 7

**4. 함수 설명 9**

1) Function 9

2) Obstacle Class 12

2) Weapon Class 17

**5. 문제 해결 과정 및 개발 후기 21**

**1. 프로젝트 진행**

1) 일정

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 프로젝트 진행 기간 | | | | | | | |
| 2020. 07. 17 ~ 2020. 07. 23. (7일) | | | | | | | |
|  | 7/17 | 7/18 | 7/19 | 7/20 | 7/21 | 7/22 | 7/23 |
| 설계 &  프로토타입 제작 |  |  |  |  |  |  |  |
| 장애물 구현 |  |  |  |  |  |  |  |
| 무기 구현 |  |  |  |  |  |  |  |
| 충돌 구현 |  |  |  |  |  |  |  |
| 랭크 시스템 구현 |  |  |  |  |  |  |  |
| 이펙트 추가 |  |  |  |  |  |  |  |
| 버그 수정 |  |  |  |  |  |  |  |

2) 개발 환경

|  |  |
| --- | --- |
| 운영체제 | Window 10 |
| 개발 프로그램 | Microsoft Visual Studio Community 2015  (Version : 14.0.25431.01 Update 3) |
| 언어 | C, C++ |
| 사용 API | Window API |
| 버전 관리 프로그램 | SourceTree  Git Bash (Version : 2.24.1.window.2) |

**2. 상세 설계**

1) 최초 설계

[ 시작 화면 ]

화면 가운데 제목이 위치함

ID를 입력 받고 시작(Enter 입력 시, 화면 전환)

[ 게임 화면 ]

랜덤 위치 장애물 생성

화면 맨 밑 부분에 벽돌 or 체력 게이지

화면 하단 중앙에 포탑(포신 회전 가능, 총알 발사 )

화면 상단에 표시되는 것(이름, 스코어)

스코어 계산 필요

게임오버 시, 화면 전환

키보드 입력으로 포신 조정 및 총알 발사

+) 이펙트 추가, 장애물 설정, 아이템 추가 및 모드

[ 종료 화면 ]

1 ~ 3등 랭킹이 보여야 함

버튼을 눌러서 해당하는 화면으로 돌아갈 수 있음(시작화면, 게임 종료)

2) 최종 설계

[ 시작 화면 ]

1) 화면 가운데 제목이 위치함

2) ID를 입력 받고 시작(Enter 입력 시, 화면 전환)

3) 화면 하단에 종료 버튼 위치

[ 게임 화면 ]

1) 화면 상단에 아이디와 스코어 표시

- 벽돌 1개당 10점으로 계산됨

2) 화면 하단에 체력 게이지 및 포탑 존재

- 포탑의 포신은 키보드로 회전 가능, 스페이스바 입력으로 총알 발사

3) 랜덤 위치 장애물 생성

- Obstacle 클래스, Vector 이용

- 장애물이 파괴될 때, 장애물이 체력바에 닿을 때 각각 이펙트 출력

4) 총알 생성

- Bullet 클래스, Vector 이용

- 스페이스바를 누르면 생성, 장애물과 충돌하거나 화면 밖으로 나갈 경우 삭제

- 한 화면에 존재할 수 있는 최대 총알의 수 제한

5) 게임오버 시, 화면 전환

- Multimap을 이용하여 자동으로 랭킹 설정

[ 종료 화면 ]

1) 1 ~ 5등 랭킹 출력

- Multimap, reverse\_iterator 이용

2) 버튼을 눌러서 해당하는 화면으로 돌아갈 수 있음(시작화면, 게임 종료)

3) 게임 종료 시, 기존의 점수들이 저장됨

- 파일 입출력과 간단한 암호화를 사용함

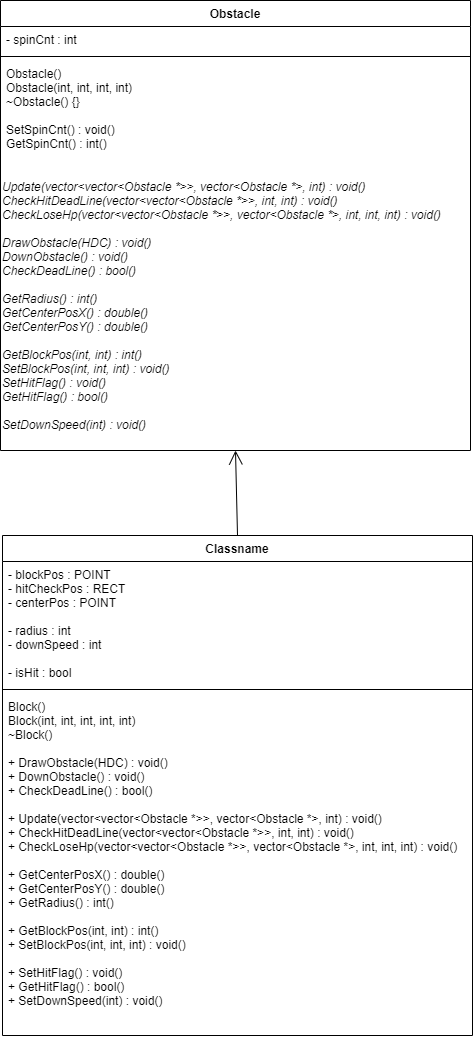
**3. 프로젝트 흐름**

1) 순서도

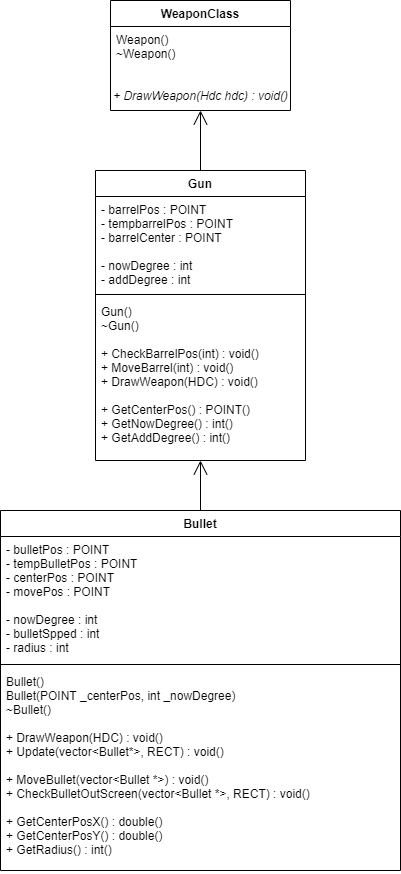
내용 추가

2) 클래스 다이어그램

[ Obstacle Class ]



[ Weapon Class ]



**4. 함수 설명**

[ Function ]

double DegreeToRadian(int degree);

* 매개변수 : 각도
* 해설 : 각도를 라디안으로 변환

POINT PointRotate(int centerX, int centerY, int degree, const POINT \* point);

* 매개변수 : 중점의 x 좌표, 중점의 y 좌표, 각도, 회전할 점의 배열
* 해설 : 중점과 각도를 기준으로 점을 회전

POINT PointRotate(int centerX, int centerY, double rad, const POINT \* point);

* 매개변수 : 중점의 x 좌표, 중점의 y 좌표, 라디안, 회전할 점의 배열
* 해설 : 중점과 라디안을 기준으로 점을 회전

void SetColor(HDC hdc, HPEN &hPen, HPEN &oldPen, int r, int g, int b);

* 매개변수 : HDC 변수, 생성한 펜, 기존 펜, rgb 값
* 해설 : 펜의 색상을 rgb 값으로 변경함

void SetColor(HDC hdc, HBRUSH &hBrush, HBRUSH &oldBrush, int r, int g, int b);

* 매개변수 : HDC 변수, 생성한 브러쉬, 기존 브러쉬, rgb 값
* 해설 : 브러쉬의 색상을 rgb 값으로 변경함

void DeleteColor(HDC hdc, HPEN &hPen, HPEN &oldPen);

* 매개변수 : HDC 변수, 생성한 펜, 기존 펜
* 해설 : 생성한 펜을 삭제하고 기존 펜 적용

void DeleteColor(HDC hdc, HBRUSH &hBrush, HBRUSH &oldBrush);

* 매개변수 : HDC 변수, 생성한 브러쉬, 기존 브러쉬
* 해설 : 생성한 브러쉬를 삭제하고 기존 브러쉬 적용

void DrawHpBar(HDC hdc, int \_hpPoint);

* 매개변수 : HDC 변수, hp 포인트
* 해설 : 플레이어의 hp 바를 그림

void SetTextColor(HDC hdc, int \_loseHpPoint, int checkNum);

* 매개변수 : HDC 변수, hp 포인트, hp 체크
* 해설 : hp체크 값에 따라 글자색 변경

void ResultScreen(HDC hdc, TCHAR \*tcharScore, TCHAR \*playerName, int playerScore, multimap<int, string> \*playerData);

* 매개변수 : HDC 변수, 플레이어 점수(TCHAR), 플레이어 이름, 플레이어 점수(int), 기존 랭킹
* 해설 : 결과 화면을 그리는 함수. 플레이어의 점수를 출력하고 기존의 랭킹 출력

void WriteRanking(multimap<int, string> \*playerData);

* 매개변수 : 랭킹
* 해설 : 게임을 종료하기 전, 랭킹을 파일로 저장

void ReadRanking(multimap<int, string> \*playerData);

* 매개변수 : 기존 랭킹
* 해설 : 게임을 시작하기 전, 파일에서 기존 랭킹을 불러와서 저장

void SaveData(multimap<int, string> \*playerData, TCHAR \*playerName, int playerScore);

* 매개변수 : 기존 랭킹, 플레이어 이름, 플레이어 점수
* 해설 : 결과화면으로 넘어가기 전 플레이 기록을 랭킹에 추가

void PrintRank(HDC hdc, RECT resultScreen, multimap<int, string> \*playerData);

* 매개변수 : HDC 변수, 결과화면 크기, 랭킹
* 해설 : 결과화면에서 랭킹 출력

[ Obstacle Class ]

Private:

Int spinCnt : 이펙트 회전 횟수

Public:

Obstacle() : spinCnt(0) { }

Obstacle(int left, int top, int right, int bottom){ }

~Obstacle() {}

* 해설 : 생성자 및 소멸자

virtual void Update(vector<vector<Obstacle \*>> &obstacle, vector<Obstacle \*> &deleteEffect, int &\_loseHpPoint) = 0

* 매개변수 : 장애물 2차원 벡터, 이펙트 벡터, hp포인트
* 해설 : 가상함수

virtual void CheckHitDeadLine(vector<vector<Obstacle \*>> &obstacle, int &hitCnt, int linePos) = 0

* 매개변수 : 장애물 2차원 벡터, 충돌 횟수, 줄 위치
* 해설 : 가상함수

virtual void CheckLoseHp(vector<vector<Obstacle \*>> &obstacle, vector<Obstacle \*> &deleteEffect, int &hitCnt, int &\_loseHpPoint, int linePos) = 0

* 매개변수 : 장애물 2차원 벡터, 이펙트 벡터, 충돌 횟수, hp포인트, 줄 위치
* 해설 : 가상함수

virtual void DrawObstacle(HDC hdc) = 0

* 매개변수 : HDC 변수
* 해설 : 가상함수

virtual void DownObstacle() = 0

virtual bool CheckDeadLine() = 0

virtual int GetRadius() = 0

virtual double GetCenterPosX() = 0

virtual double GetCenterPosY() = 0

virtual void SetHitFlag() = 0

virtual bool GetHitFlag() = 0

virtual void SetDownSpeed(int speed) = 0

* 해설 : 가상함수

virtual int GetBlockPos(int i, int check) = 0

* 매개변수 : 벡터 내 장애물 위치, x, y 제어 변수
* 해설 : check값에 따른 장애물의 x, y값 반환

virtual void SetBlockPos(int i, int check, int val) = 0

* 매개변수 : 벡터 내 장애물 위치, x, y 제어 변수, 장애물 위치 설정 변수
* 해설 : check값에 따른 장애물의 위치 설정

void SetSpinCnt()

* 해설 : 이펙트 회전 횟수 설정

int GetSpinCnt()

* 해설 : 이펙트 회전 횟수를 가져옴

[ Block Class : public Obstacle ]

Private:

POINT blockPos[4] : 장애물 좌표

POINT centerPos : 중점

int radius : 반지름

int downSpeed : 장애물 하강 스피드

bool isHit : 충돌 횟수

public:

Block();

Block(int left, int top, int right, int bottom, int \_downSpeed)

~Block();

* 해설 : 생성자 및 소멸자

void DrawObstacle(HDC hdc)

* 매개변수 : HDC 변수
* 해설 : 장애물을 화면에 그려줌

void DownObstacle()

* 해설 : 장애물 하강

bool CheckDeadLine()

* 해설 : 장애물이 체력바에 닿았는지 체크

void Update(vector<vector<Obstacle \*>> &obstacle, vector<Obstacle \*> &deleteEffect, int &\_loseHpPoint)

* 매개변수 : 장애물 2차원 벡터, 이펙트 벡터, hp포인트
* 해설 : 장애물의 현재 상태 업데이트

void CheckHitDeadLine(vector<vector<Obstacle \*>> &obstacle, int &hitCnt, int linePos)

* 매개변수 : 장애물 2차원 벡터, 충돌 횟수, 줄 위치
* 해설 : 장애물이 체력바에 닿았는지 체크

void CheckLoseHp(vector<vector<Obstacle \*>> &obstacle, vector<Obstacle \*> &deleteEffect, int &hitCnt, int &\_loseHpPoint, int linePos)

* 매개변수 : 장애물 2차원 벡터, 이펙트 벡터, 충돌 횟수, hp포인트, 줄 위치
* 해설 : 체력바와 충돌이 일어난 장애물 수 만큼 체력 감소 및 장애물 제거

double GetCenterPosX()

double GetCenterPosY()

* 해설 : 장애물의 중심 좌표 반환

int GetRadius()

* 해설 : 장애물의 반지름 반환

int GetBlockPos(int i, int check)

void SetBlockPos(int i, int check, int val)

* 해설 : 해당하는 위치의 장애물 반환 및 해당 좌표로 설정

void SetHitFlag()

bool GetHitFlag()

* 해설 : 충돌 플래그 설정 및 반환

void SetDownSpeed(int speed)

* 해설 : 해당 속도 만큼 장애물 하강

[ Weapon Class ]

Public:

Weapon() { }

~Weapon() { }

* 해설 : 생성자 및 소멸자

virtual void DrawWeapon(HDC hdc) = 0

* 매개변수 : HDC 변수
* 해설 : 가상함수

[ Gun Class : public Weapon ]

Private:

POINT barrelPos[4] : 포신 위치

POINT tempbarrelPos[4] : 포신 회전을 위한 임시 변수

POINT barrelCenter = { 250, 650 } : 포신의 중점

int nowDegree : 현재 각도

int addDegree : 더해질 각도

Public:

Gun()

~Gun()

* 해설 : 생성자 및 소멸자

void CheckBarrelPos(int inputKey)

* 매개변수 : 입력 방향
* 해설 : 입력 방향에 따른 회전각 설정

void MoveBarrel(int inputKey)

* 매개변수 : 입력 방향
* 해설 : 입력 방향에 따라 포신 회전

void DrawWeapon(HDC hdc)

* 매개변수 : HDC 변수
* 해설 : 무기를 화면에 그림

POINT GetCenterPos()

* 해설 : 포신의 중점을 반환

int GetNowDegree()

* 해설 : 포신의 현재 각도를 반환

int GetAddDegree()

* 해설 : 포신에 더해질 각도를 반환

[ Gun Class : public Gun ]

Private:

POINT bulletPos[4] : 총알의 위치

POINT tempBulletPos[4] : 총알의 회전을 위한 임시 변수

POINT centerPos : 총알의 중점

POINT movePos : 이동 방향

int nowDegree : 현재 각도

int bulletSpped : 더해질 각도

int radius : 반지름

Public:

Bullet()

Bullet(POINT \_centerPos, int \_nowDegree)

~Bullet()

* 해설 : 생성자 및 소멸자

void DrawWeapon(HDC hdc)

* 매개변수 : HDC 변수
* 해설 : 총알을 그림

void Update(vector<Bullet\*> &bullet, RECT viewRect)

* 매개변수 : 총알 벡터, 화면 크기
* 해설 : 총알의 현재 상태 업데이트

void MoveBullet(vector<Bullet \*> &bullet)

* 매개변수 : 총알 벡터
* 해설 : 총알을 주어진 이동방향으로 이동시킴

void CheckBulletOutScreen(vector<Bullet \*> &bullet, RECT viewRect)

* 매개변수 : 총알 벡터, 화면 크기
* 해설 : 총알이 화면 밖으로 나갔는지 확인

double GetCenterPosX()

double GetCenterPosY()

* 해설 : 총알의 중심점 반환

int GetRadius()

* 해설 : 총알의 반지름 반환

**5. 문제 해결 과정 및 개발 후기**

생성한 블록을 라인으로 저장, 라인을 장애물 클래스에 저장하는 형태로 구현해서 이차원 벡터 객체를 사용했는데, 삭제하는데 어려움이 있었다.

충돌 구현을 하던 도중, 총알이 블록을 관통하거나 이상한 곳의 블록을 없애는 문제가 발생하였다. 계산에는 문제가 없는 것 같아서 며칠간 고민하다 실시간으로 중점 좌표를 찍어보게 되었는데, 총알의 중점을 이동방향으로 이동하던 것이 이상하게 적용되어 중점좌표가 음수 값이 되는 것을 발견할 수 있었다. 결국 계산 직전에 총알의 현재 위치에 따른 중점을 반환 받아서 충돌 구현을 완료할 수 있었다.