INVERSUS

✓ 2022180024 유영빈
GITHUB : https://github.com/kevin0181/INVERSUS

기본 조작 방법

> 게임을 진입하게 되면 Enter을 클릭하여 시작하고, 키보드의 방향키를 통해서 게임 초기 설정이 가능하다.
> 키보드 조작 명령어를 통해서 플레이어를 움직일 수 있다.
> 게임 진행 중, esc를 통해서 fuzz가 가능하다.
> esc를 누른 후 진입 상태에서 게임 설정이 가능하다.

키보드

> **조작 명령어**
>
> 1. w,a,s,d : 메인 블럭의 움직이는 키
> 2. 방향키 : 총알을 발사 키, 설정 움직이는 키

> **기본 명령어**
>
> 1. esc 👉 설정 창으로 노래 변경 및 무적모드 on/off
> 2. [ ] 👉 소리 크기 변경
> 3. l 👉 보드 뒷 라인 보이기 on/off

마우스 - 사용 ❌

난이도

> - 쉬움 : 보드 및 적 블록 수가 적고, 적 블록의 크기가 크다.

- 보통 : 보드 및 적 블록 수가 적당하고, 적블록의 크기가 조금 크다.
- 어려움 : 보드 및 적 블록 수가 많고, 메인 블록의 크기보다 적 블록의 크기가 조금 크다.
- 매우 어려움(초고수) : 보드 및 적 블록 수가 매우 많고, 전체적인 블록의 크기가 메인 블록의 크기보다 작다.

스코어

- 생존하는 시간에 비례하여 점수가 오른다.
- 적 블럭을 총알로 부셔 스코어를 얻을 수 있다.
- 적 블럭을 연속적(콤보)으로 부셔 배수의 스코어를 얻을 수 있다.

점수에따른 업그레이드

- 1000점 달성 시 : 방향키를 1.3초간 누르면 빠른 총알 발사 가능
- 5000점 달성 시 : 총알의 장전 최대 수량을 10발로 늘림
- 10000점 달성 시 : 메인돌을 중심으로 위성 총알이 생김

규칙

- 메인 블록을 움직일 수 있고, 만약 빨간색 블록과 연한 빨간색 블록과 부딪힌다면 데미지가 들어온다.
- 메인 블록은 방향키를 통해 총알을 발사할 수 있고, 총알은 총 6발로 장전하여 발사가 가능하다.
- 게임은 무한으로 반복되고, 오른쪽 상단의 피가 0에 가까워 진다면 게임이 끝난다.

게임 시작하기

1. 플레이어 선택하기 (2인 플레이어 불가❌)
2. 난이도 선택하기
3. 게임 플레이 🎮

구현한 내용 👍

1. 보드 구현

1. 먼저 보드판을 그리기 위해서 전체 화면의 크기를 가져온다.
2. 레벨에 따른 셀 크기를 다르게 만든다.
3. 보통 모니터는 가로 값이 크니깐 가로를 기준으로 한 칸의 크기 정하여 보드 그린다.
4. 보드는 선으로 그리고 그 위에 검은 블록을 하나씩 생성한다.

2. 메인 블록 이동 구현

1. 블록 구조체에 상, 하, 좌, 우에 대한 방향값 (bool)을 넣어준다.
2. 타이머를 통해서 방향에 대한 값이 존재하면 돌의 스피드 만큼 이동한다.
3. 특수 블록은 지나갈 수 없다.

3. 총알 발사 구현

1. 방향키를 통한 총알 발사가 가능하다.
2. 방향키를 눌렀을때 누른 방향에 대한 값과 누르고 있는 시간을 계산 한다. (총알 속도를 다르게 하기 위해)
3. 방향키를 땠을때 상태값을 바꿔 장전되어있는 첫번째 총알을 발사 한다.
4. 땠을때의 방향으로 총알이 날라간다. (눌렀을때 x)
5. 날라가는 방향으로 총알 머리부터 꼬리까지 그라데이션이 발생한다.
6. 그라데이션은 날라가는 방향에 맞춰 반복문을 돌려 RGB(255,255,255)라면 점점 RGB의 값이 내려가도록 만들

4. 총알 재장전 구현

1. 처음 게임이 시작되면 0발의 총알로 시작한다.
2. 게임이 시작까지 대략 3초가 걸리므로 게임이 시작하면 3발의 총알이 장전된다.
3. 타이머를 통해서 장전 가능한 총알 개수보다 적으면 1.3초에 한발 씩 장전된다.
4. 장전되어 있는 상태값과 총알을 발사하고 있는 상태값 두개를 변수로 선언 하였다. (status, bullet_move_
5. 장전된 총알은 메인 블록(플레이어)을 중점으로 시계 방향 회전한다.
6. rotateBullets() -> 중점으로부터 각각의 장전된 총알의 거리를 주고, 중점과의 현재 총알의 거리를 angl

5. 플레이어(메인 블록)과 검은 블록 충돌 처리

1. 플레이어는 방향의 상태값에 따라서 타이머를 통해 움직인다.
2. 플레이어의 진행 방향에 검은 블록이 충돌하게 된다면, 진행 방향의 반대 방향으로 플레이어의 속도만큼 빼 진행

6. 플레이어와 적(레드 블록)의 충돌처리

1. moveRedBlock() -> 적이 플레이어를 쫓아옴과 동시에 충돌 체크를 하는 함수
2. 무적모드가 off 상태인지 체크한다
3. 적이 활성화 되어 있는 상태라면 IntersectRect()를 통해서 충돌 체크한다.
4. 만약 충돌되었다면 플레이어의 상태값을 false로 변경한다.

7. 플레이어 부활

1. 플레이어와 적이 충돌한다면 플레이어의 방향 초기화 및 위치를 가운대로 이동 후, 타이머(2)를 통해서 부활 타
2. 3초의 부활타이머가 진행 되고, 플레이어의 상태값을 true로 변경하여 부활한다.

8. 특수 총알

1. 적은 기본적으로 일반 총알(흰색)과 특수 총알(빨간색)을 드랍한다.
2. 특수 총알은 특수 블록을 부술 수 있다.
3. 총알 구조체에 특수 총알을 구분하기 위한 변수를 선언했다. (special)
4. 적은 총알을 드랍하는데 50%의 확률로 특수 총알을 1~3개를 드랍한다.
5. 특수 총알의 경우 변수로만 구분하고, 구조체에 color값만 빨간색으로 구분지었다.

9. 드랍된 총알

1. 적은 죽으면 1~3개의 총알을 드랍한다.
2. 총알은 적이 죽은 위치를 중점을 기준으로 시계 방향으로 회전한다.
3. 플레이어의 장전된 총알과 마찬가지로, 죽은 적(레드 블록)의 위치(RECT)를 가져온다.
4. 해당 위치의 중점을 구한 후, 중점을 기준으로 반복문을 드랍된 총알의 수 만큼 돈다. (1~3)
5. dropBulletAngle 변수 -> 다음 변수를 통해서 드랍된 총알의 각도를 구한다.
6. bullet_drop_print() -> 함수를 통해서 중심으로 부터의 거리 를 구하여 출력한다.
7. !!문제점!! : 거의 마지막에 구현한 기능이라, 타이머를 통해서 구현하려 했지만 문제(메모리)가 발생하여
계산을 WM_PAINT 부분에다가 했다. 게임을 진행하면서 문제는 발생하지 않았지만,
만약 플레이어가 죽으면 모든 타이머를 멈추고 2번 타이머만 작동하게 만들고 싶었는데
플레이어가 죽으면 WM_PAINT에 작성하여 총알은 계속 회전한다.

10. 드랍된 총알 먹기

1. 드랍된 총알은 1~3개이다.
2. 플레이어의 최대 장전 가능한 총알 수와 장전되어 있는 총알의 수를 계산한다.
3. 드랍된 총알이 만약 3개이고, 최대 장전 가능한 총알 수는 6개, 현재 장전되어 있는 총알이 5발이라면
드랍된 총알의 1발의 총알만 플레이어에게 장전한다.
즉, 드랍된 총알을 먹으면 남은 개수의 총알만 회전한다.

11. 적(레드 블록) 생성 및 검은 블록 생성

1. 타이머(4)를 통해서 적을 생성한다.
2. 모든 적이 죽어 크기가 0이 되면, 적을 1개 블록부터 최대 10개까지 레벨에 따라 다르게 생성한다.
3. 먼저 검정 블록의 전체 크기를 구한다.
4. 블록의 전체 크기안에서 랜덤한 수를 구한 후, 랜덤한 수의 검은 블럭의 RECT를 새로 생성하는 빨간 블록 REC
5. 빨간 블록은 vector에 담는다.
6. 혹시 같은 수가 나와서 빨간 블록이 같은 자리에 나오는걸 방지하기 위하여, set을 사용했다.
7. 적은 스코어에 따라 스폰(재생성) 시간이 짧아진다. 초기에는 3초에서 10000점이 넘으면 1.1초로 짧아진다.
8. 적 블록은 블록 한개의 크기 만큼, 추가적인 크기를 갖는다.
해당 크기는 한 셀의 크기만큼 갖고 빗금 모양으로 되어 있다.
9. 빗금 모양 부분은 현재 적 블록의 크기에서 한 셀만큼 계산하여 RECT의 크기를 늘렸고,
빗금을 먼저 출력한 후 적 블록을 그 위에 출력하였다.

---

1. 적 블록이 지나간 자리가 만약에 검정 블록이 활성화 된 상태가 아니라면,
(false 라면) true로 변경하여 활성화 상태로 변경한다.
2. 검정 블록을 반복문을 돌아 현재 빨간 블록의 위치와 충돌 체크한다.

12. 총알과 적(레드 블록) 충돌 체크

1. 총알의 size만큼 반복문을 돌린다.
2. checkRedBlockBullet() -> 총알과 적 블록의 충돌 체크 함수
3. 적 블록을 반복문을 돌려 적 블록과 총알의 충돌 처리를 체크한다.
4. 적 블록의 크기와 총알의 맨 앞부분의 크기만큼 충돌해야 적 블록이 사망한다. (그라데이션 충돌 체크 X)

---

13. 폭발 애니메이션

1. Explosion class -> 폭발 애니메이션을 그리기 위해서 위치, 색깔을 생성자를 통해 받는다.
2. 만약 적 블록을 사망시킨다면 explodes 배열을 통해서 위치, 색을 받는다.
3. 입력받은 위치에 한해서, 총 8프레임으로 나누어 폭발(Ellipse) 애니메이션을 그린다.
4. 예를 들어, 플레이어 블록이 사망한다면 플레이어 블록의 사망 위치, 색을 넣어 폭발 애니메이션을 보여준다.
5. 1프레임마다 죽은 위치의 랜덤한 위치에 랜덤한 사이즈로 Ellipse 형태로 나오게 된다. 타이머(1)

14. 두 가지의 적 블록

1. 기본적인 빨간 블록은 특수 블록 위를 지나서 플레이어에게 다가온다.
2. 연한 빨간 블록은 특수 블록을 지나갈 수 없다. 특수 블록과 부딪히게 된다면 진행 방향에 플레이어가 있다면 진
3. 적 블록을 생성할 때, 50%의 확률로 연한 빨간 블록이 생성된다.
4. 연한 빨간 블록은 Block class의 red_special 변수를 통해 관리된다.
5. 특수 블록과 충돌 처리할 때, red_special 변수가 true인 것만 체크한다.

15. 화면 흔들림 효과

1. explodes 변수는 사망한 블록들이 있는 배열이다.
2. explodes에 있는 값들이 폭발 애니메이션을 실행 후, 값들이 지워지면 화면 흔들림을 실행한다.
3. moveRect() -> 함수를 통해서 검은 블록의 원래 위치를 기억한다.
4. 타이머(7)를 통해서 move_count의 값 만큼 랜덤한 값 만큼 x, y로 계속 움직인다.
5. move_count가 0이 되면 원래 위치로 되돌아 온다.
6. !!문제점!! : 만약에 두 적 블록 객체를 빠른 속도로 동시에 죽인다면,

화면 흔들림이 진행 되고 있는 와중에 원래 위치를 흔들리고 있는 값으로 저장하게 되어 화면 전체가 랜덤한 위치 만
해결 방안 : 적 블록을 사망시켰을때 검정 블록의 원래 위치를 저장하는게 아니고, 초기에 따로 저장을 해두고 가져

16. 플레이어 및 적 생성 애니메이션

1. 플레이어의 상태값이 false일때 타이머(2)가 실행되며 모서리가 둥근 테두리 사각형이 점점 줄어드
2. 적 블록도 초기에 생성되면 false 상태이고 타이머(3)에서 처리되며 색은 오로지 빨간색인 생성 0
3. 타이머를 통해서 테두리 사각형이 줄어들게 만들었고, 타이머가 끝나면 적 블록의 상태값을 true로

17. 콤보

1. 적 블록을 처치하면 주변의 적 블록도 동시에 처치 된다.
2. aroundRect -> 변수를 통해서 빗금 부분을 표시했다.
3. 해당 부분의 크기와 다른 블록의 위치와 충돌 체크한다.
4. aroundBroken()-> 함수를 통해서 주변에 적 블록과 충돌 처리한다.
5. 존재한다면 폭발 애니메이션을 위해 값을 explodes에 넣고, 값을 지운다.

구현하지 못한 내용 👎

1. 2인 플레이

1. 개발 초기에는 생각해놓고 만들었는데, 만들다보니깐 사이즈가 커져서 코옵은 못했다.

2. 관통형 총알

1. Bullet class에 through변수가 있는데 이 부분을 활성화 하면 총알이 적들을 관통하며 부신다.
2. 기능적으로 구현을 해놔서 변수 값만 변경하면 작동한다.
3. 추가하기 애매한거 같아서 뺐다.

추가적인 구현 🖐

1. BGM

- 기본 브금과 그 외 2개의 추가적은 브금이 있다.
- esc를 눌러 설정창에서 브금을 변경할 수 있다.
- [와] 를 통해서 인게임 브금의 소리 크기를 변경할 수 있다.
- 소리 두개를 동시에 틀려면 thread를 추가해야해서 thread를 통해서 브금을 관리했다.

2. 빠른 총알

- 방향키를 1.3초간 누른다면 누른 시간을 계산하여 빠른 속도의 총알이 발사된다.

- 기본적인 총알의 발사 속도는 speed(7)이고, 빠른 발사 속도는 max_speed(15)이다.
- 방향키를 누르고 있는 시간만큼 타이머를 통해 값을 계산하여 누른 시간이 1.3초를 누른다면 빠른속도의 총알을 발사 할 수 있다.
- 스코어를 1000점 넘긴다면 사용할 수 있다.

3. 총알 장전 최대 수 변경

- 기본적으로 총알의 최대 수는 6발이다.
- 만약 스코어 5000점을 넘기면 최대 10발을 장전할 수 있다.

4. 위성 총알

- 스코어 10000점을 넘기면 위성 총알이 생긴다.
- 위성 총알은 초록색으로 rotatingBullet 변수에 저장되어 있다.
- 위성 총알은 적 블록에 닿으면 처치할 수 있지만, 콤보는 적용되지 않는다.
- 위성 총알도 장전과 드롭된 총알과 마찬가지로 중점을 두고 해당 중점을 기준으로 반시계 방향으로 움직인다.

개발하며 알게된 점❗

1️⃣

타이머를 사용할 때, 화면 갱신을 위해서 InvalidateRect 함수를 많이 사용했다. 하지만 만약 1번 타이머가 1ms마다 해당 함수를 요청하면 1ms마다 창을 새로 그린다. 근데 2번 타이머가 10ms에다가 InvalidateRect를 지속적으로 요청한다면 화면에 부하(?)가 온다는것을 알았다. 어느순간 게임을 진행하는데 화면에서 뭔가 끊기는 느낌을 받아서 하나씩 코드를 찾아봤는데 InvalidateRect를 1번 타이머에서만 요청하게 만드니 사라졌다.

2️⃣

플레이어 블록과 검정 블록의 충돌 검사를 한다고 예시를 둬보자.
그러면 충돌 검사 부분을 함수로 빼놨다고 할때, 매우 어려움(초고수) 모드에서는 약 800개의 검정 블록을 반복문을 돌린다.
나는 보통 함수를 불러올때 매게 변수를

예시 -> 함수명 (Class 변수이름){};
이렇게 불러왔다. 근데 어느 순간부터 매우 어려움 모드에 들어가면 화면이 조금씩 멈추는 느낌이 들었다. 느낌적으로 메모리 사용이 많아져서 그렇겠다 생각해서 하나씩 확인해봤다.

해결 방법을 찾았는데 값을 변경할 필요가 없는 변수에는 const를 사용하고 레퍼런스를 사용하는것이다. 내 생각에는 함수의 매게변수도 새로운 메모리에 넣는거라 레퍼런스를 사용하면 메모리 주소를 가져오기 때문에 부하(?)가 덜하고

const를 사용하면 값을 추가할 일이 없는 것이니깐 메모리 사용에 좋은거 같다는 생각이다. 아마도..? const는 거의 사용해본적이 없어서 이번에 메모리 관리가 꽤나 중요할 수도 있겠다는 생각을 했다.