STL

과제 2

학과 : 게임공학부 엔터테인먼트컴퓨팅 전공

학번 : 2014184018

이름 : 안 종 구

**<< 목차 >>**

**표지……………………………………………………………………………………… (1)**

**목차……………………………………………………………………………………… (2)**

1. **프로그램에 대한 설명………………………………………………. (3)**
2. **게임 소개…………………………………………………………….. (3)**
3. **게임을 선택한 이유………………………………………………..... (3)**
4. **조작 방법……………………………........................................... (4)**
5. **과제를 위해 수정한 사항.......................................................... (4)**
6. **과제 해결방법............................................................................ (5)**
7. **컨테이너의 선택....................................................................... (5)**
8. **리플레이의 저장과 호출............................................................ (6)**
9. **과제를 해결하면서 배운 점....................................................... (7)**
10. **해결하지 못한 점..................................................................... (8)**
11. **과제를 마치고 느낀 점........................................................... (10)**
12. **프로그램에 대한 설명**
13. **게임 소개**

**< 이상한 나라의 EVE >**

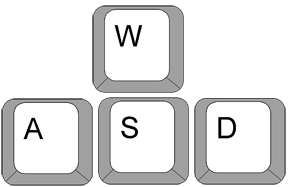


* 제가 과제2에서 사용하는 게임은 작년 2학기 컴퓨터그래픽스 시간에 텀프로젝트로 제작했던 이상한 나라의 EVE 입니다. 이상한 나라의 엘리스를 모티브로 하였고 주인공이 엘리스에서 월-E영화에 나오는 EVE로 바뀐 게임입니다., 3인칭 시점으로 여러 스테이지(방)를 클리어 하며 보스 왕토끼를 잡아내면 꿈에서 깨어나는 게임입니다.

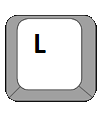
1. **게임을 선택한 이유**

* 가장 처음으로 만들었던 게임이기도 하며, 과제전에서 금상을 받아 애정도 있는 작품입니다.
* 처음엔 오픈지엘을 사용한 게임은 일반적으로 main함수의 구조가 조금 달라서 구현하는 데 어려움이 있어 콘솔게임을 골라 과제를 했었습니다. 하지만 제작 중 아쉬움과 이전 과제에서 STL에 대한 전반적인 이해의 부족 때문에 조금 더 잘 알아보고 더 잘 만들어 보고 싶었습니다.

1. **조작 방법**



방향키는 WADS로 일반적인 FPS게임과 같은 조작키를 사용합니다.



총알은 L키를 이용해 발사합니다.

- Save는 게임을 실행시킨 후 ‘t’키를 누르고 종료 지점에 ‘p’ 키를 누르면 저장이 됩니다. (저장을 원하시면 게임이 시작될 때 바로 실행시켜 주세요! 위치는 저장되지 않습니다.)

- Time을 기준으로 리플레이는 ‘o’키를, Step을 기준으로 리플레이는 ‘I’키를 눌러주시면 됩니다.

- i키를 누른 후 ‘m’키를 입력하면 앞으로, ‘n’키를 입력하면 뒤로 재생됩니다.

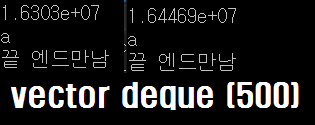
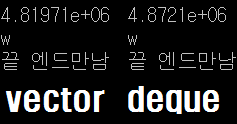
1. **과제를 위해 수정한 사항**

* 원래 KeyBoard에서 입력을 처리해주고 Timefunc에선 일부 입력을 처리해 주었지만, 입력을 처리해주는 함수를 따로 만들어 모든 입력이 있을 때 그 함수를 호출하는 형태로 수정하였습니다.
* Replay설명을 위해 Text ui를 추가하였습니다.

1. **과제 해결방법**
2. **컨테이너의 선택**

STL에서 중요한 것을 하나 선택을 하라면 **‘어떤 상황에 어느 컨테이너를 선택할 것인가’**를 가장 먼저 떠올릴 것 입니다. 이번 과제 역시 가장 먼저 다가간 방향은 컨테이너의 선택 이었습니다.

우선 리플레이를 구현하려면 **데이터의 저장 및 호출이 가장 중요**하다고 생각했습니다. 데이터의 저장과 호출은 순차적으로 저장하고**, 앞서부터 순차적으로 출력이 가능하되, 저장된 순서를 바꾸지는 않아야 합니다.** 따라서 기준이 되는 값에 따라 순서를 바꿔주는 tree형태의 컨테이너는 고민하지 않고 제외하였습니다. List컨테이너는 STL컨테이너 중 가장 강력한 예외 안정성을 제공해주고, vector과 같이 양방향으로 오갈 수 있는 좋은 컨테이너 였지만, Iterator로 끝을 불러와 줄 때 begin과 end에 -1 혹은 +1 같은 연산을 할 수 없어 다른 위치에 이러한 연산을 하는 코드를 추가해 줘야 하는 불편함이 있었고, Deque컨테이너의 경우 저장 원소가 많고 메모리 할당량이 큰 경우에는 vector에 비해서 확장 비용이 절감되고, 원소들간의 연산이 필요 없는 프로그램이기 때문에 유용하다고 생각했지만, 구동 결과 벡터가 미세하게 빨라 **결국 vector컨테이너를 사용하기로 하였습니다.**



->약 200개 출력 결과 ->약 500개 출력 결과

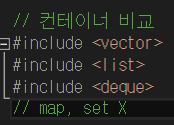
1. **리플레이의 저장과 호출**

* 콘솔창을 되도록이면 사용하지 않기 위해 리플레이 파일의 이름을 미리 지정해 두었습니다. 따라서 별도의 파일명 입력 없이 t,p키와 o,i키 만으로 저장과 불러오기를 실행시킬 수 있습니다.
* 2개의 데이터를 쉽게 저장하고 호출하기 위해 과제1과 같이 make\_pair을 사용해 보았습니다. Pair를 갖고 데이터를 관리하면, 저장과 호출 모든 부분에서 두 데이터를 연관되게 관리할 수 있어 매우 편리하다고 생각했습니다.
* 게임의 특징상 부드러운 움직임이 필요하기 때문에 키 입력을 glutTimerFunc()에서 받아왔습니다. **OpenGL에서 glutTimerFunc()은 계속해서 재귀호출되기 때문에 제가 원하는 순간에만 입력을 처리할 수가 없고, 이 안에서 for문을 사용할 경우 모든 처리가 끝난 후 애니메이션을 처리해 줍니다.**
* **위와 같은 문제점을 해결하기 위해 Load를 할 때는 bool값을 이용해 상태를 저장해 주었습니다.** 출력하는 상태가 되면 파일을 가져와 하나씩 읽어와 입력을 처리해주는 행동을 해주며, 전역 iterator I 변수를 1씩 증가시켜주다 end앞을 만나게 되면 다시 출력하는 상태를 false로 바꿔 저장하여 시간에 따라 계속해서 호출되는 함수에서도 성공적으로 출력이 가능하게 되었습니다.
* 이 문제를 해결하는 과정에서도 **for문안에 있던 반복자 i,i2 를 클래스나 전역으로 선언해서 관리해주지 않으면 지역에서 사라지고 다른 지역에서 필요하게 될 때 문제가 발생하여 전역변수로 선언해 주었습니다.** (전역변수로 선언하는 게 썩 좋지 못한 것 같지만 다른 해결 방법을 찾지 못하였습니다..)
* 



1. **과제를 해결하면서 배운 점**

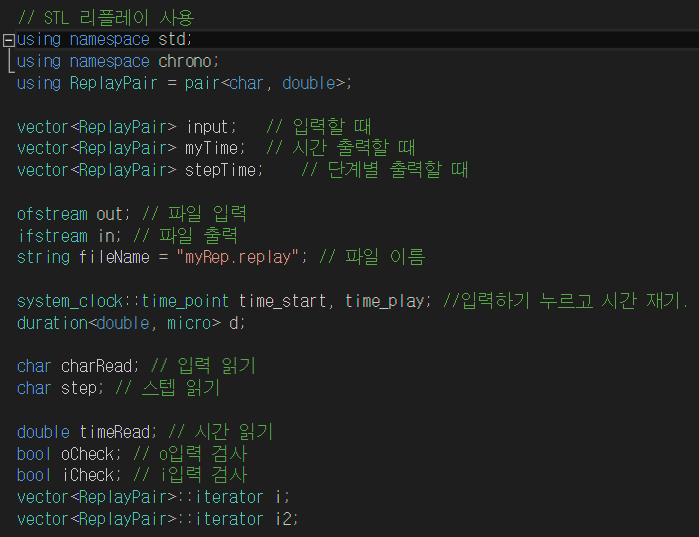
* 사실 과제1에서는 충분히 이해를 하지 못했던 것 같습니다. **컨테이너를 왜 나눠놨는 지, 각각의 컨테이너가 무슨 일을 하고, 어떤 구조로 되어 있는 지를 충분히 알지 못했습니다.** 하지만 과제2를 하면서 앞서 미처 깨닫지 못했던 컨테이너들의 장단점과 그에 따른 용도를 알 수 있었습니다. 물론 수업중에 교수님께서 충분히 설명을 해 주셨지만, 공부를 하면서 책과 인터넷의 단순한 예제들로는 의구심이 많이 들었습니다. 하지만 **직접 프로그램을 해보고 속도를 비교해보기도 하고, 각각 컨테이너가 어떤 의미를 갖는지 충분히 생각하고 어떤 컨테이너가 나에게 적합한지 직접 정해보니 이해가 잘 되었습니다.** 평소에 vector이외에 잘 사용하지 않던 컨테이너들도 각각의 장단점을 이해하고 나서는 앞으로 일단 vector가 아닌 내가 무엇을 해야 하는 지 먼저 생각하고, 그에 따라 필요한 컨테이너를 고를 수 있을 것 이란 생각이 들었습니다.



* C++을 이수하지 않은 상태에서 듣는 STL이라서 초반에는 남들보다 많이 부족했습니다. 방학과 학기초 STL수강을 위해 혼자서 공부를 했지만, 한계가 있었습니다. class와 오버로딩, 생성자, 소멸자등 어렵고 중요한 내용들은 교수님께서 학기 초에 충분히 설명해 주셔서 어느정도 이해가 갔지만, 파일 입출력은 쓸 때마다 항상 어려움을 겪어왔습니다. 시험은 물론 가장 최근에 봤던 퀴즈에서도 파일 입출력은 인터넷과 독학으로는 이해가 안되는 부분이 많았습니다. 그렇지만 이번 과제를 하면서 완벽하진 못해도 충분히 공부가 된 것 같습니다. Ofstream과 ifstream에 있는 다양한 ios::xxx의 의미도 알아보고 내가 필요한 것이 무엇인지 생각도 할 수 있는 기회가 된 것 같습니다. 아직은 모자란 부분이 많지만, 막막했던 이전과 달리 앞으로는 공부하며 충분히 채워나갈 수 있을 것 같습니다.

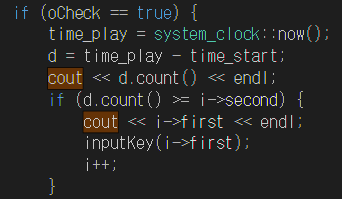
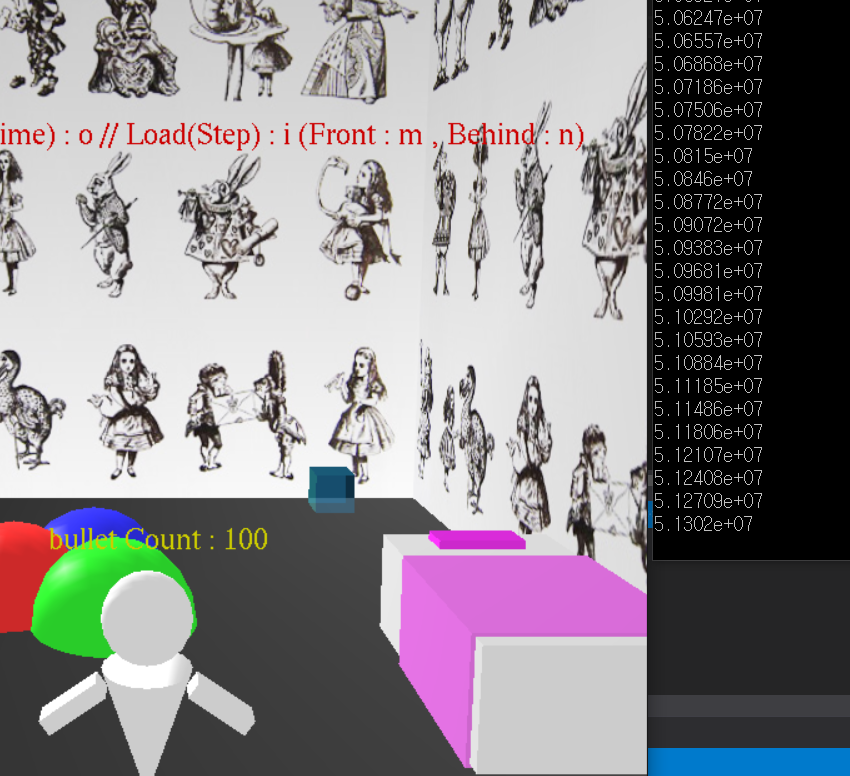
1. **해결하지 못한 점**

* **전역 변수의 불안**
* 새로 추가된 거의 모든 변수를 전역으로 선언했습니다. 분명 더 좋은 방법이 있을 것 이란 생각이 들었지만, 도저히 해결 방법을 찾아내지 못하고 있습니다. 전역 변수가 너무 많아진 것 같아서 프로그램에게 너무 미안한 마음이 들었습니다.



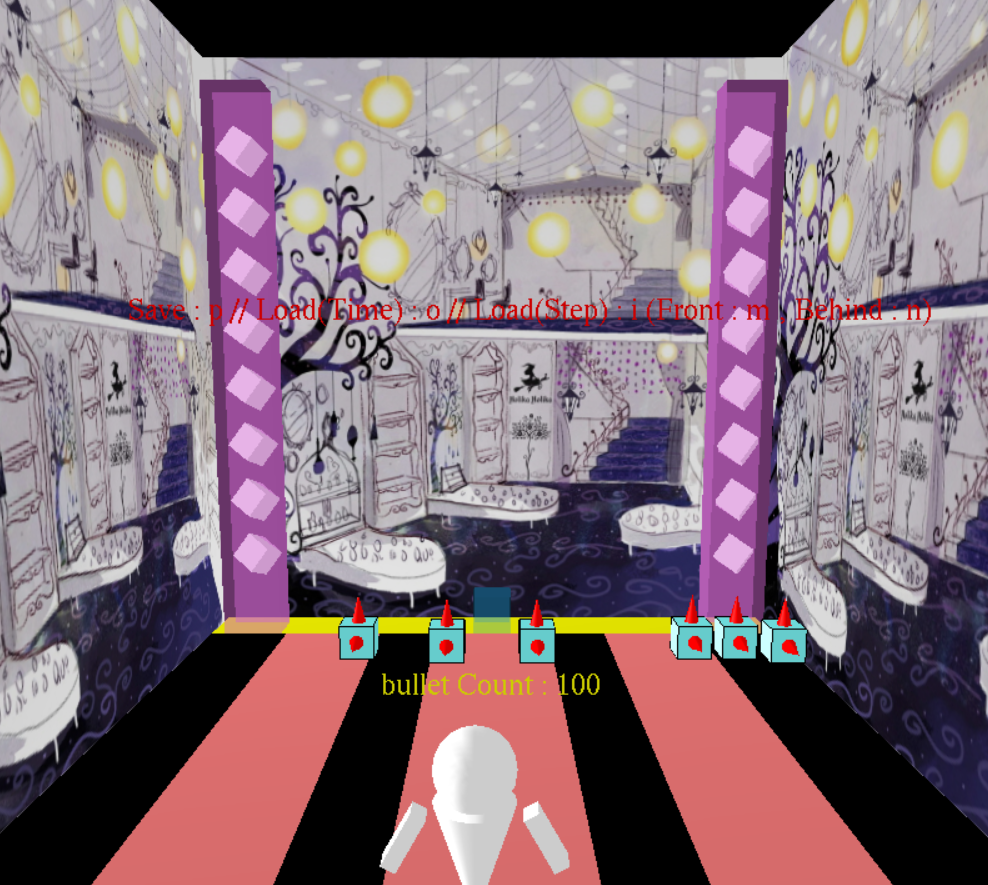
**수많은 전역변수들….**

* **움직인 후 저장? / 호출?**
* 정말 많은 시간을 투자했지만 알 수 없는 버그였습니다. 게임이 시작되고 바로 t키를 누른 뒤 원하는 지점에서 p 키를 눌렀을 때에는 제대로 동작 하지만, 살짝 혹은 어느정도 움직이고 난 뒤 t -> p 키를 누르면 Load(Time)은 1칸만 이동하고 동작을 하지 않습니다. 비슷한 방식으로 코드를 짠 Load(Step)는 정상 작동을 합니다. 정말 해결하고 싶었지만 해결하지 못했습니다.



**계속해서 d.count()만 출력하는 모습**

* **랜덤한 Enemy미사일 처리**
* 스테이지3에는 벽에서 랜덤한 미사일이 나와 내 캐릭터가 목적지에 도달하지 못하게 합니다. 데이터를 저장할 때는 내 캐릭터의 입력만 받기 때문에 랜덤으로 발사되는 미사일은 저장하지 못하고, 그 때문에 3탄부터는 리플레이 동작중에 같은 입력을 하여도 미사일은 랜덤하게 발사되어 스테이지를 클리어하지 못하는 경우가 발생합니다.

 **<<랜덤한 미사일의 발사. 맞으면 같은 스테이지 처음으로.**

1. **과제를 마치고 느낀 점**

과제1을 할 때는 컨테이너들의 비교라는 생각을 하지 못하고 있어 매우 부끄럽게 생각 했습니다. 교수님께서 과제1에 대한 이야기를 하실 때 마다, 나쁜 예를 말씀해 주실 때 마다 전부 저의 이야기 인 것 같아서 고개를 들 수가 없었습니다. 좀 더 일찍 알았다면, 조금 더 일찍 깨닳았다면 조금 더 잘 할 수 있을 것 같은데, 라는 아쉬움이 많이 남아있었습니다. 수업을 들으면서는 전부 이해한 것 같았는데, 막상 뭐가 중요한 지도 모르고 있었던 것 이었습니다. 그 후로 교수님께서 중요하다고 하시는 것은 책으로 한번 더 읽어보고, 부족한 부분이 있으면 인터넷을 찾아보면서 공부를 해왔는데 이런 것들 또한 많이 부족했다는 것을 다시 느꼈습니다.

또한 STL이라는 것을 배우면서 컨테이너들을 어떻게 이용해야 좋은 건지, 왜 이것들이 효율적이고 중요한 건지 잘 몰랐었습니다. 하지만 과제2를 하면서 ‘이런 방식으로 데이터를 관리할 수 있구나!’ 라는 것을 가장 먼저 느꼈습니다. 별 거 아닌 것 같던 컨테이너들이 갑자기 대단한 것 같아졌습니다. 무엇보다 과제를 하면서 반복자들을 여기저기 넣어보며 바꿔보다 보니 막연했던 반복자들이 조금 더 친숙하게 느껴졌습니다.

과거에 했던 소스에 과제를 진행해서 이전 소스를 들여다 보니 정말 더럽고 구조도 알아보기 힘들었습니다. 파일 분할이나 클래스 같은 좋은 구조를 잘 다룰 수 있도록 충분한 공부가 더 필요하다는 생각과 프로그램을 누가 읽어도 알아보기 쉽게 잘 설계해야 한다는 것을 알게 되었습니다. C++ 기본기가 부족해서 학기 초에는 많이 힘들었지만, 수업 내용을 이해하기 쉽게 잘 설명해 주신 덕분에 잘 따라갈 수 있었지만, 앞으로를 위해서 STL뿐만 아니라 C++에 대해서도 더 깊고 자세히 공부해야 할 것 같았습니다.

무엇보다도 이번 과제에서 가장 많이 느꼈던 것은 교수님께서 자주 말씀하시던 ‘컴퓨터는 거짓말을 하지 않는다’라는 말이었습니다. 예전에는 “나는 맞는데 내 컴퓨터 왜이래!”하며 화를 내던 것들을 다시 찾아보고 수정하고 하다가 보니 결국 전부 저의 잘못이었다는 것을 다시 한 번 느낄 수 있었습니다.

작년 컴퓨터 그래픽스 수업을 들을 때 텀프로젝트 과제때문에 교수님께서 선배님들을 불러 수업 중에 보여주신 적이 있었습니다. 그 때 STL이라는 것을 이용해 리플레이를 구현하신 선배님들을 보며 정말 멋지다고 생각해서 그것을 목표로 엔터테인먼트 컴퓨팅과에서는 4학년 수업 임에도 불구하고 수강을 하였습니다. ‘내가 만든 첫 게임에도 리플레이 기능을…’ 이라는 소박한 목표를 마침내 조금은 미숙하지만 이룬 것 같아서 너무 행복했습니다. 앞으로도 작은 목표를 한 계단 한 계단 밟아가며 좋은 프로그래머가 될 수 있도록 노력하겠습니다.