2012182045 최재용

Replay

STL 과제 2 보고서

내용

[소개 2](#_Toc452698193)

[진행과정 2](#_Toc452698194)

[기본 프로그램 구현 2](#_Toc452698195)

[리플레이 구현 3](#_Toc452698196)

[프로그램 조작법 6](#_Toc452698197)

[소감 7](#_Toc452698198)

소개

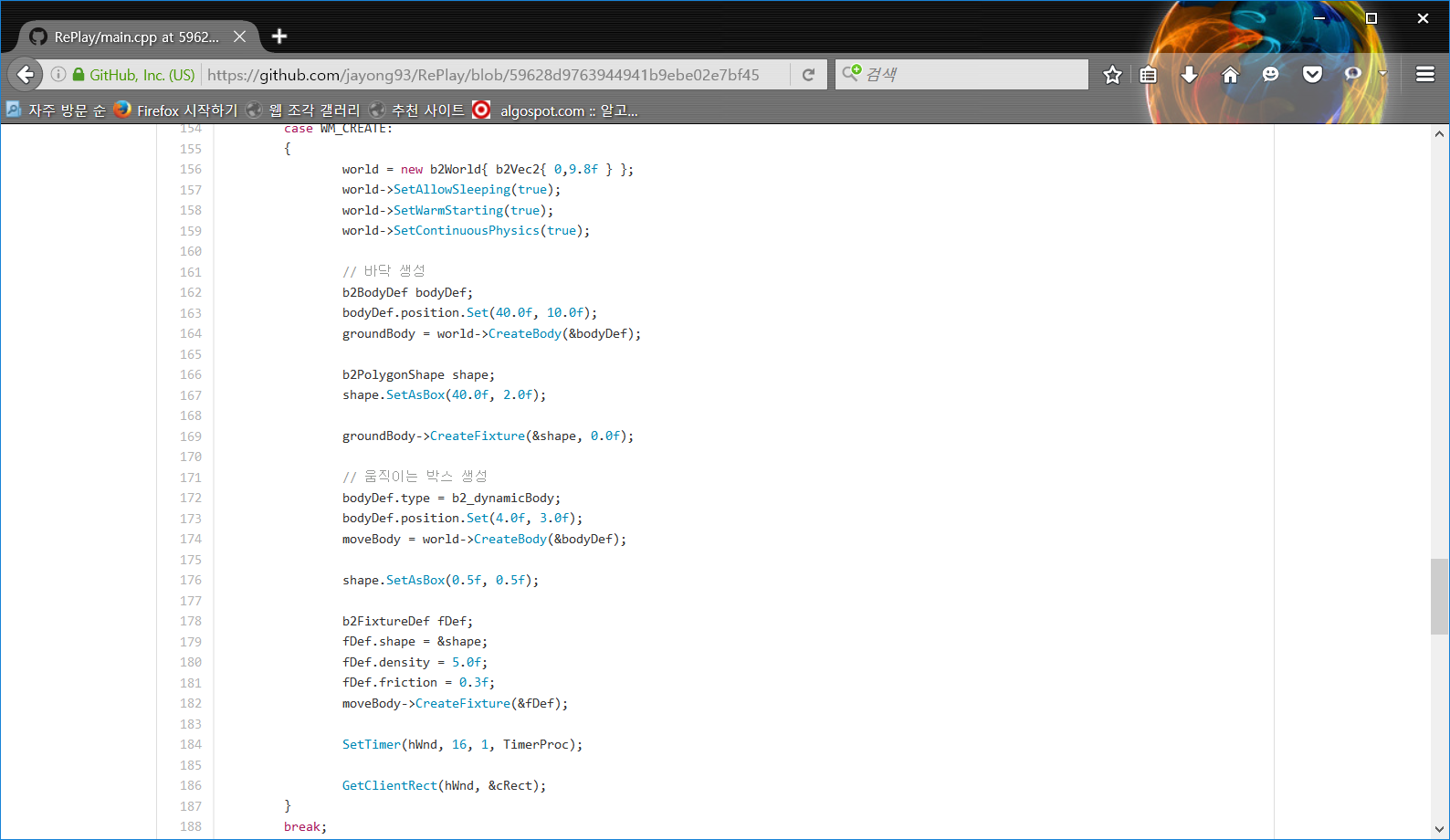
저는 이번 리플레이 과제에서 박스들을 가지고 노는 프로그램을 만들었습니다. 이 박스들은 Box2D 강체 물리엔진 라이브러리를 이용해서 물리법칙에 따라 움직이도록 구현했습니다. 제 목표는 이 복잡한 물리 계산이 들어간 움직임을 리플레이로 그대로 복구하는 것이었습니다.

처음 기획때는 당구대 위에 공들은 만들고 공을 마음대로 굴리는 느낌으로 표현하려고 했지만 원 보다는 사각형이 움직이는게 시각효과가 좋을 것 같아서 바꾸었습니다.

진행과정

기본 프로그램 구현

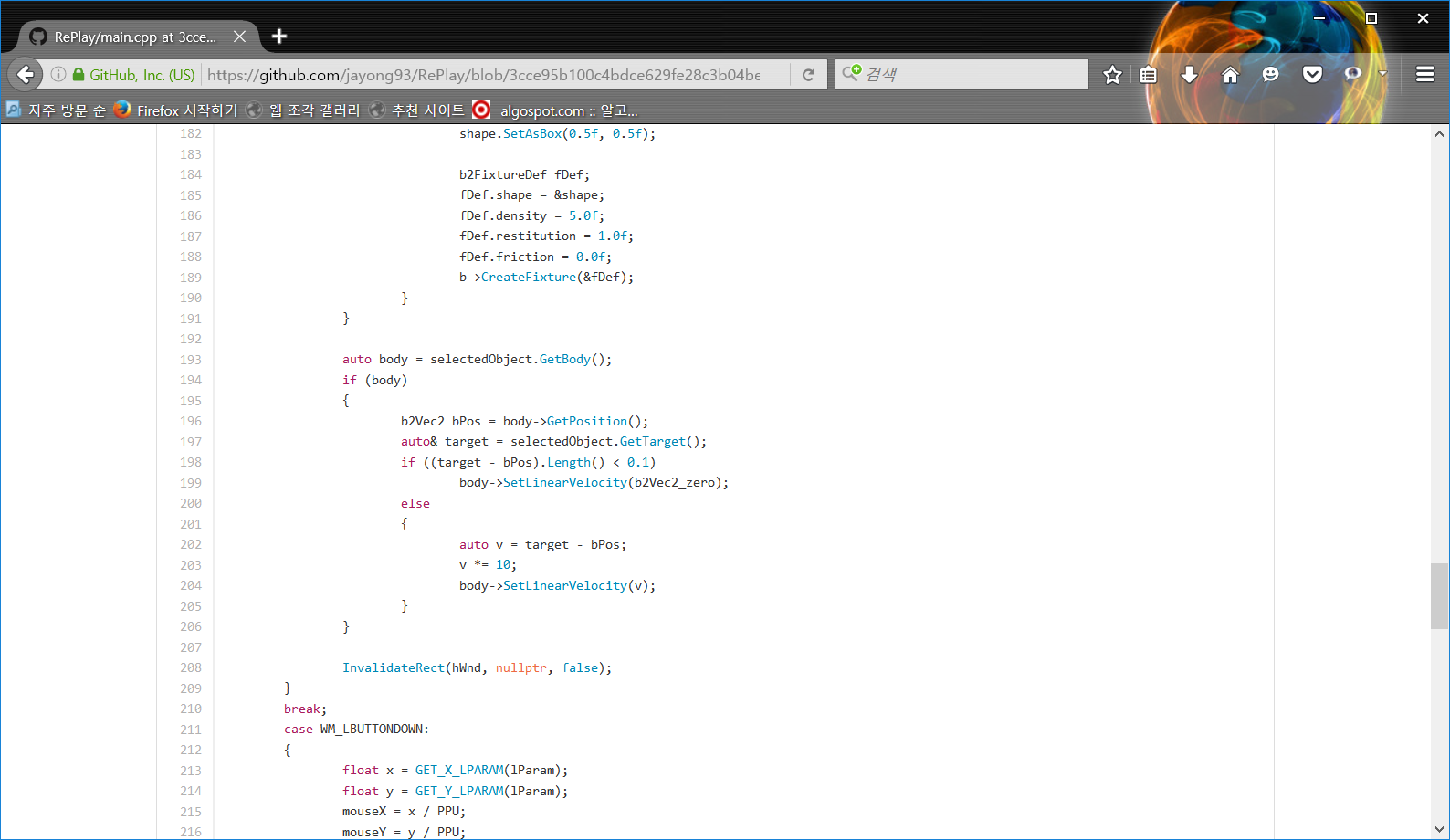
프로그램을 만들기 전에 먼저 Box2D에 대해서 공부를 했습니다. 찾아보니 Box2D는 World라는 객체를 통해서 물리 강체들을 관리하고 물리 연산을 모두 처리하는 방식으로 동작했습니다. 그래서 원래는 화면 상의 객체 모두를 STL 컨테이너로 관리하려고 했지만, 박스들은 Box2D에게 관리를 맡기고 저는 리플레이에 필요한 데이터만 STL 컨테이너로 관리하기로 했습니다.



Box2D에서 강체를 나타내는 Body는 이렇게 world 객체의 생성함수를 통해서 만들어지고 내부의 Tree 안에서 관리가 됩니다

Box2D 라이브러리가 객체를 모두 관리해주다 보니 굉장히 편하게 프로그램을 만들 수 있었습니다. 리플레이 기능을 구현하기 전까지는 제가 한 일은 사용자의 입력을 받아서 알맞은 행동(박스를 생성하거나 움직이는 것)을 연결해주는 것 밖에 없었습니다.

그렇게 해서 빈 공간에 마우스를 클릭하면 박스가 생성되고, 박스를 클릭해서 드래그하면 박스가 움직이도록 만들었습니다. 또 빈 공간에서 마우스를 드래그하면 마우스 위치에 박스를 놓았을 때 다른 박스와 충돌하는지 검사한 후에, 충돌하는 박스가 없다면 박스를 새로 생성하도록 했습니다. 이렇게 하면 선을 그리듯이 박스들을 만들 수 있습니다.



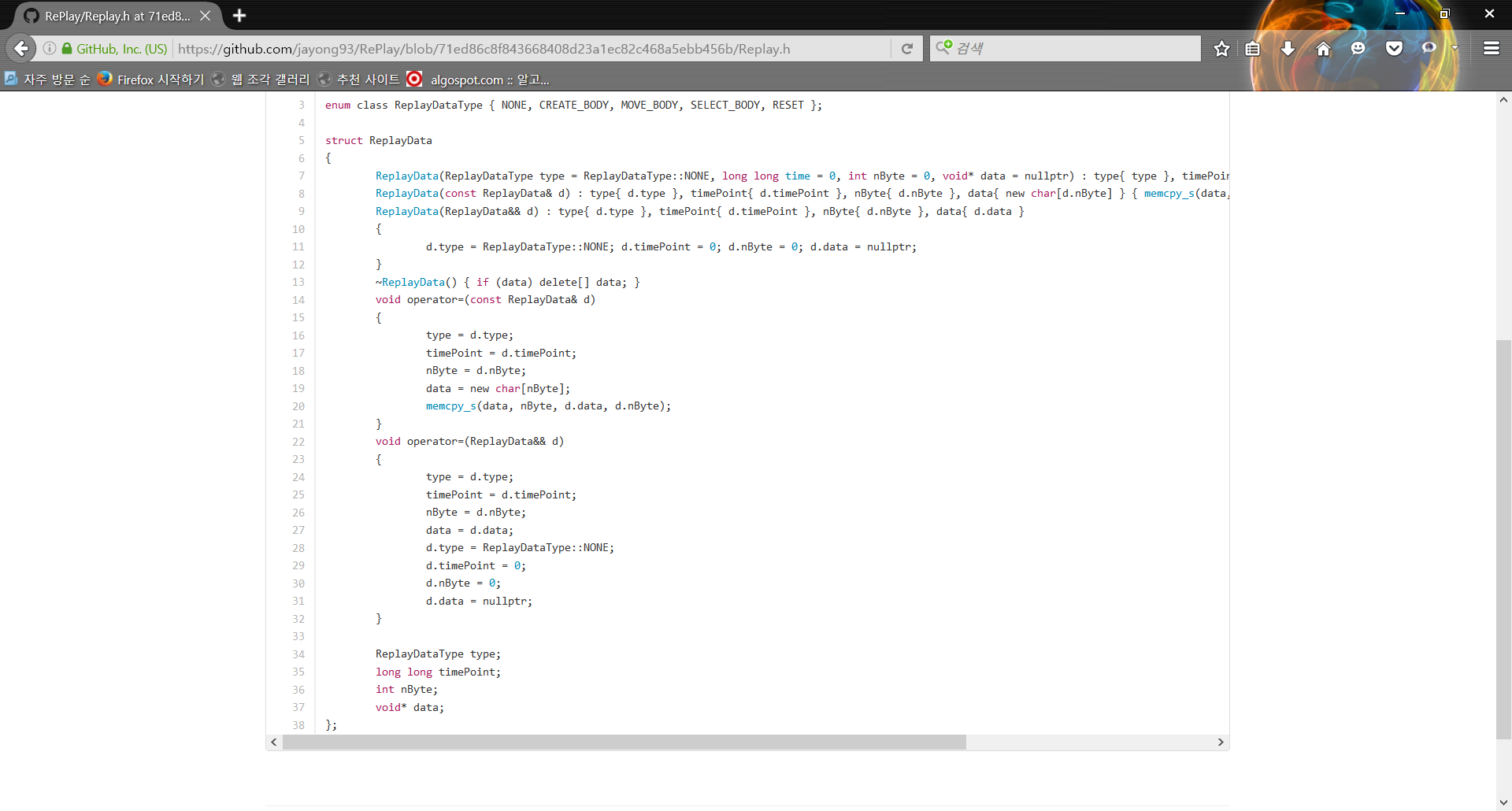
이런 식으로 박스에게 SetLinearVelocity 함수로 속도 값을 주면 물리엔진이 박스들의 상태를 계속 갱신해주고, 저는 갱신된 값들을 받아와서 계속 그려주었습니다.

리플레이 구현

기본 프로그램 구현은 Box2D의 힘 덕분에 쉽게 끝낼 수 있었지만 문제는 리플레이 기능의 구현이었습니다. 무언가를 계속 저장해 놓아야 한다는 것은 알고 있었지만 정확히 무엇을 저장해야 하는지를 몰라서 꽤 오래 고민을 했습니다.

우선 어떤 데이터들을 저장할지 생각해보았습니다. 처음엔 매 프레임의 오브젝트 위치를 저장할까 했지만 그렇게 되면 리플레이 파일 용량이 너무 커질 것이라고 생각했습니다. 단순하게 계산해서 박스 각각의 위치는 x,y 좌표를 모두 float로 저장하면 8바이트 입니다. 그렇다면 박스를 100개만 만들어도 한 프레임에 대한 정보는 800바이트가 될 것입니다. 리플레이 길이가 1분이고, 초당 60프레임이 그려진다고 하면 2880000바이트, 거의 3MB에 가까운 공간을 차지합니다.

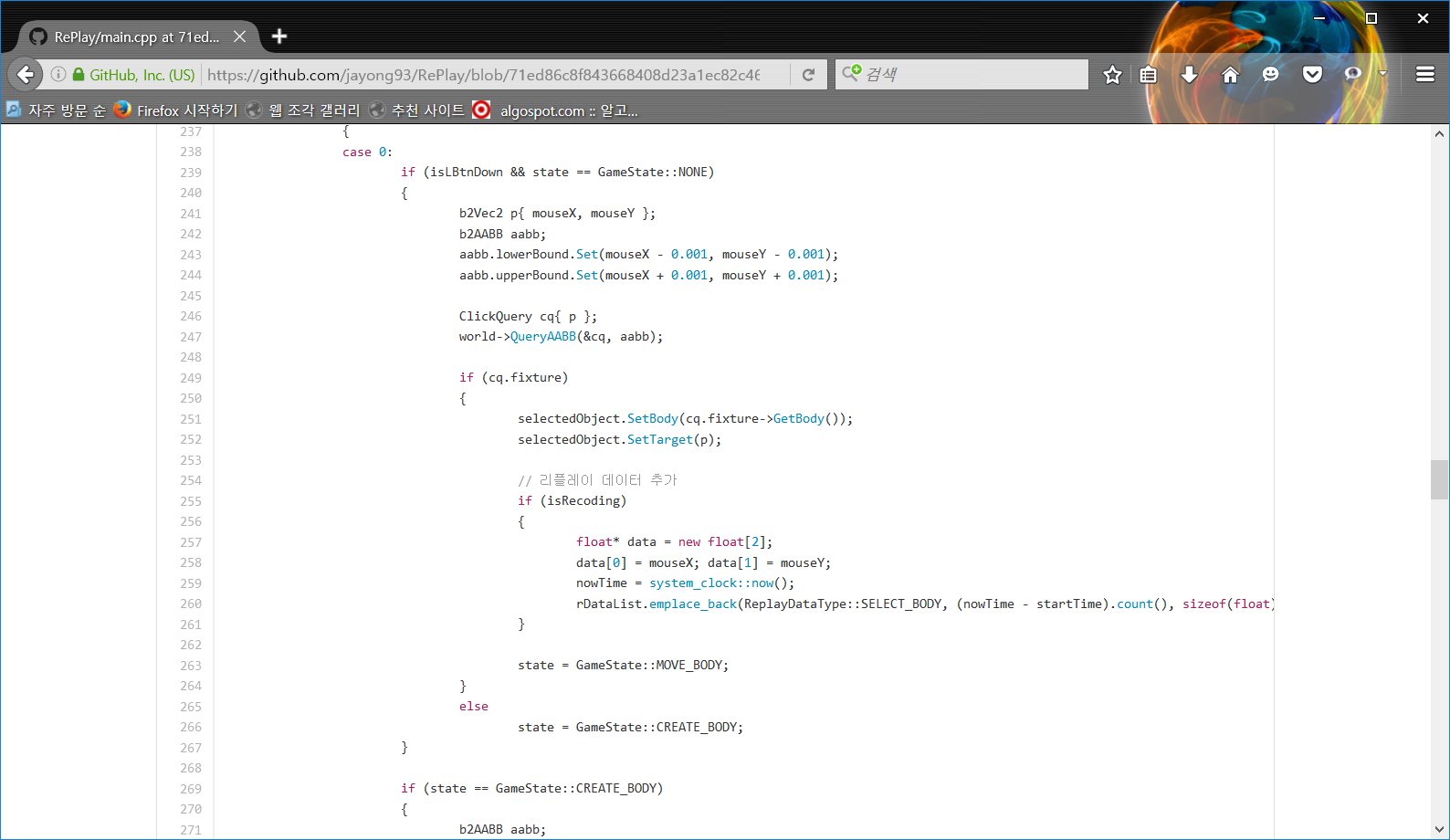
저는 그래서 특정 이벤트(객체 생성, 선택, 이동, 삭제)가 일어날 때 그 이벤트를 저장하는 방식을 시도했습니다. 그러기 위해서 ReplayDataType 열거체과 ReplayData 구조체를 만들었습니다.



ReplayData는 이벤트가 일어났을 때 그 이벤트의 세부 정보들을 저장하는 구조체입니다. type 변수는 이벤트의 종류를 나타내는 데이터로 ReplayDataType 열거형으로 만들었습니다. CREATE\_BODY는 박스 생성, MOVE\_BODY는 박스 움직임, SELECT\_BODY는 마우스를 통한 박스 선택, RESET은 모든 박스 제거를 나타냅니다.

timePoint 변수는 어느 시점에 이벤트가 발생했는지를 저장합니다. Chrono 라이브러리를 이용해서 프로그램 시작한 때를 기준으로 얼마의 시간이 흐른 뒤에 이벤트가 일어났는지 time\_point끼리 뺄셈을 해서 duration을 구한 후, count 함수로 값을 받아와서 여기에 저장했습니다.

이벤트를 저장할 때 추가 데이터가 필요한 경우 동적할당 된 메모리의 주소를 넣게 했는데, nByte는 메모리의 바이트 수, data는 메모리의 주소를 가지게 됩니다.

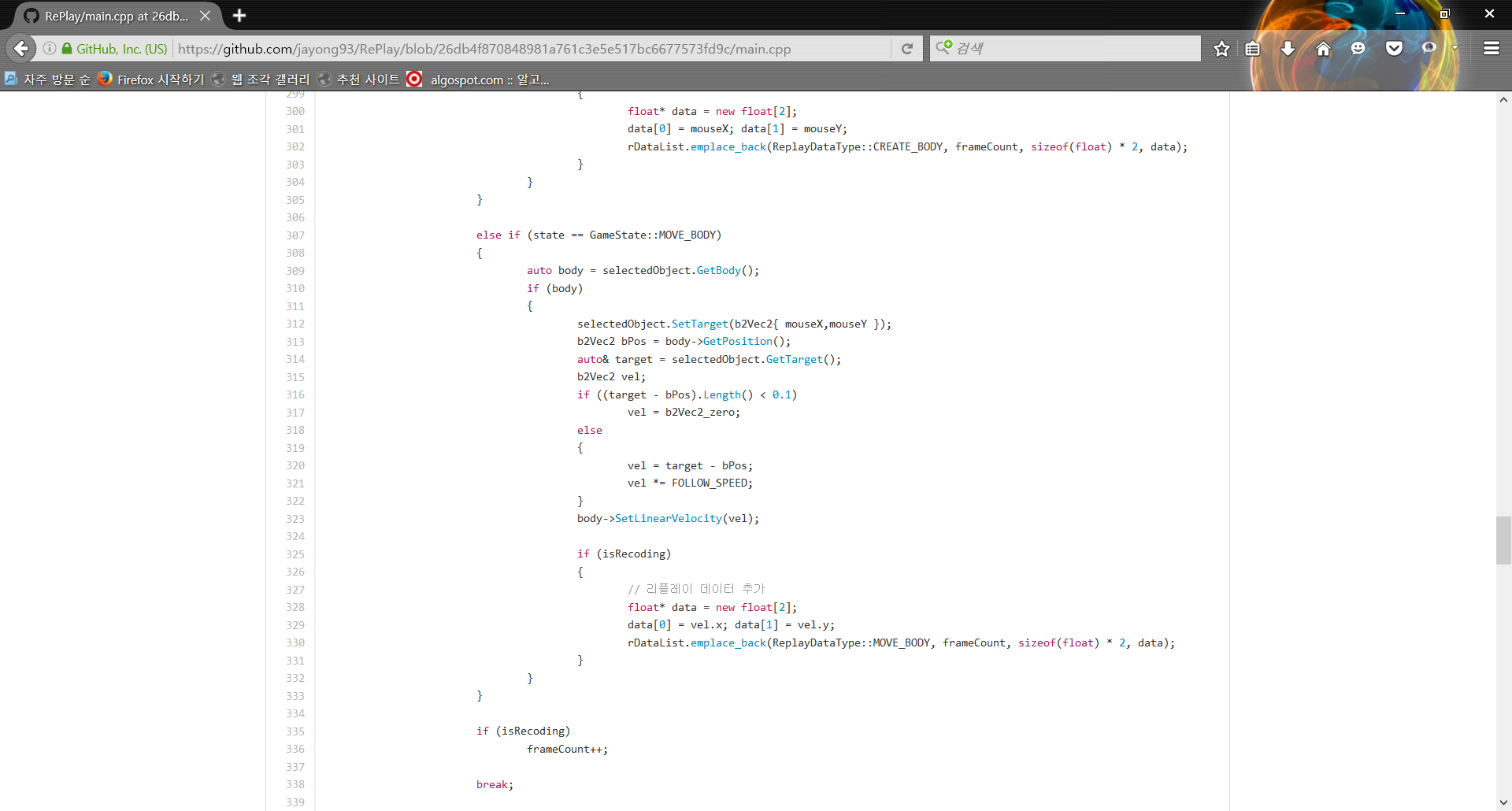


실제 데이터는 위의 코드처럼 저장했습니다. 이 데이터들은 STL 컨테이너에 저장됩니다. 여러가지 컨테이너를 고려했는데 이 데이터들은 리플레이에 쓰이고, 시간 순서에 따라서 추가되기 때문에 검색과 정렬이 필요가 없었고 그래서 map, set은 후보에서 제외했습니다. 남은 vector, list, deque 중에서는 이 프로그램에서는 항상 컨테이너의 마지막에서만 데이터의 삽입이 일어나고, 리플레이를 할 때는 임의접근 없이 앞에서부터 뒤로 순차접근만 하기 때문에 list가 가장 적합하다고 생각했습니다.

리플레이를 할 때는 이렇게 저장된 데이터를 처음부터 읽어오면서 리플레이를 시작하고 지금까지 흐른 시간이 데이터에 기록된 시간보다 크거나 같으면, 데이터를 다시 이벤트로 바꿔서 프로그램이 처리할 수 있도록 만들었습니다.

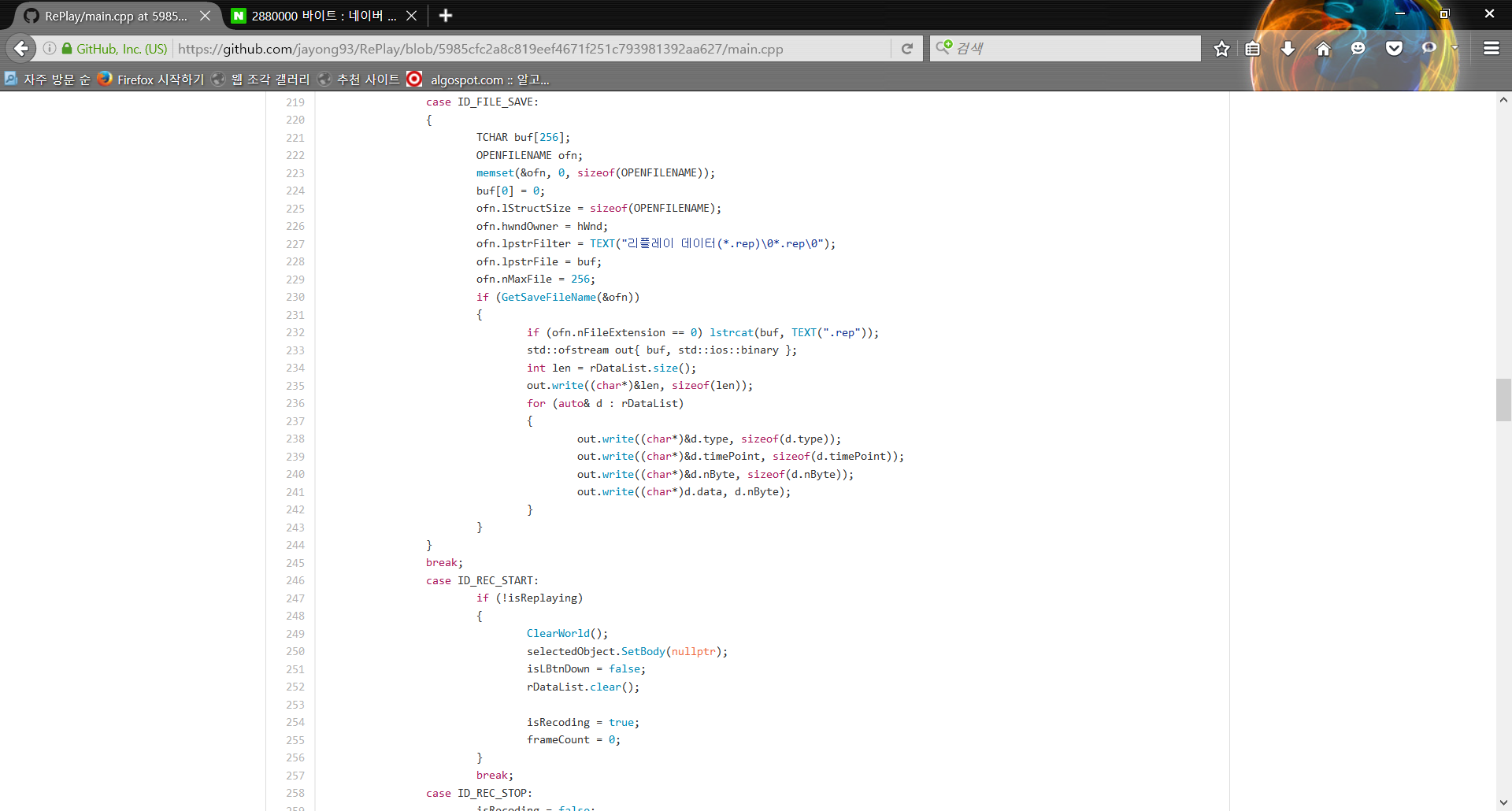
그런데 이렇게 만들어보니 리플레이를 반복할 때 마다 조금씩 다른 결과물이 나왔습니다. 원인을 고민해본 결과, 박스 객체들은 Box2D 물리엔진의 연산으로 움직이기 때문에 시간의 오차가 조금만 있어도 박스의 이동 방향이 달라질 수 있다는 것을 알게 됐습니다. 매 프레임의 처리시간은 계속 달라질 수 있기 때문에 이벤트를 처리하는 시점의 오차는 생길 수 밖에 없어서 이 방법은 포기해야 했습니다.

그래서 저는 방향을 바꿔서 '몇 번째 프레임에 해당 이벤트가 일어났는가' 를 찾아서 저장했습니다.



frameCount는 현재 처리되고 있는 프레임이 몇 번째 프레임인지 나타내는 정수입니다. 초기값은 0이고 리플레이 녹화 중에 frameCount를 매 프레임마다 1씩 증가시켜서 저장합니다. 이벤트가 발생한 경우 ReplayData에 timePoint 대신 이 값을 넣어서 저장하고, 리플레이 도중에 처리되고 있는 프레임의 순서가 데이터에 저장된 순서와 같으면 데이터를 이벤트로 바꿔주었습니다. 이렇게 바꾸고 나니까 저장된 리플레이를 몇 번 실행시켜도 항상 같은 결과화면이 나오게 되었습니다.

이제 남은 일은 파일에 저장하는 것 뿐이었습니다. 최대한 리플레이 파일의 크기를 줄이고 싶었기 때문에 binary 형식으로 저장하기로 했습니다.



제일 처음엔 데이터의 개수를 4바이트에 기록하고, 범위 기반 for문을 이용해서 각 데이터를 모두 2진수로 파일에 작성했습니다. 읽어올 때는 반대로 파일의 처음 4바이트를 읽어서 데이터의 개수를 얻은 다음에 그 개수만큼 for문을 돌려서 파일에서 데이터를 읽고 list에 저장하는 것을 반복했습니다. 이렇게 저장하니까 모든 박스의 위치를 저장하는 방식보다 약 20배 이상 적은 크기로 리플레이 파일을 만들 수 있었습니다.

프로그램 조작법

메뉴

* 녹화 시작 : 리플레이 녹화 시작
* 녹화 종료 : 리플레이 녹화 종료
* 초기화 : 리플레이 데이터 리스트 초기화

마우스

* 왼쪽 클릭(빈 공간) : 박스 생성
* 왼쪽 드래그(빈 공간) : 연속적으로 박스 생성
* 왼쪽 드래그(박스) : 박스 이동

소감

평소에 여러가지 게임들, 특히 다른 사람과 대전하는 게임들을 하면서 리플레이 기능이 굉장히 유용하고 신기해서 어떻게 만드는지 궁금했습니다. 마침 STL 강의에서 과제로 리플레이 기능을 만든다는 이야기를 예전부터 들어서 기대하고 있었는데 역시 만만하지는 않았습니다. 물론 용량이나 시간 같은 것을 생각하지 않고 대충 만든다면 훨씬 간단하긴 했겠지만 효율을 생각하려고 보니 고민할 것이 한두개가 아니었습니다. 게다가 직접 동적 배열을 만들어서 데이터를 관리하려고 했다면 굉장히 귀찮고 불편했을 것입니다. 그나마 STL 컨테이너를 이용해서 속도와 안정성에 크게 신경을 쓰지 않고도 프로그램을 만들 수 있어서 고민을 덜 수 있었습니다.

개인적으로 정밀한 Chrono 라이브러리를 제대로 이용해보고 싶었는데 제 프로그램의 특성상 사용하기 힘들어서 아쉬웠습니다.

제가 필요한 데이터를 binary로 나름의 형식에 맞춰서 파일에 저장해본 것은 이번이 처음이었는데 텍스트 파일에 비해 굉장히 적은 용량임에도 잘 작동해서 굉장히 뜻 깊었습니다. 정말 오랜만에 마치고 나서 보람이 있었던 과제였습니다.