리플레이 프로그램
<그림판>

해결방법

리뷰

게임공학과 | 2014180038 | 정명준

Concept

그림판에 그려지는 내용을 저장하고, 저장된 내용을 재생한다.

그림판에 그려지는 내용만 저장하는 것이 아니라, 전체 "그려지는 과정"을 저장한다.

"그려지는 과정"을 동영상 녹화하듯이 기록하고 저장한다.

저장된 파일을 열어서 녹화된 "과정"을 재생한다.

프로그램은 Win API 를 활용한다.

저장되는 데이터(마우스 좌표 등)는 STL 의 컨테이너로 관리한다.

Iterator, Algorithm 을 이용하여 문제를 해결한다.

주요 Concept

1. 메뉴 및 키 입력 정의

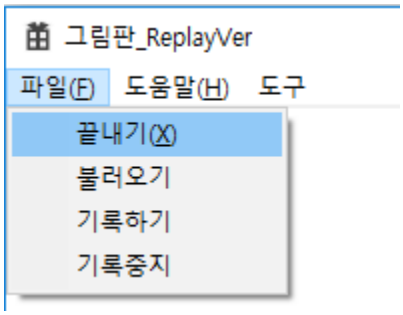
2. WIN API 를 이용한 그림판 프로그램 구현

3. 파일 입/출력 데이터 정의 및 처리

4. 플레이 예시

1. 메뉴 및 키 입력 정의

1) 파일 메뉴



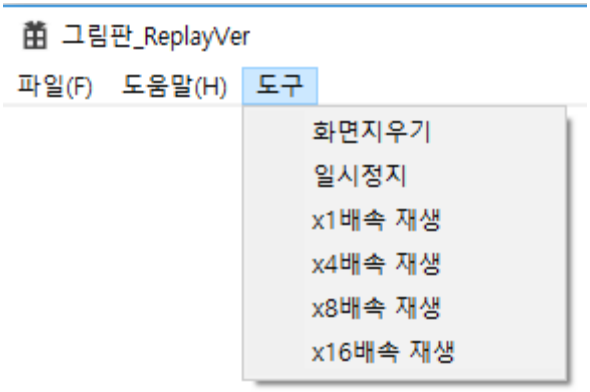
끝내기: 현재 실행되고 있는 프로그램을 종료.

불러오기: 저장된 파일의 데이터를 불러와서 Vector 로서 관리. 데이터에 따라 그리기 재생

기록하기: 기록 시점을 기준으로 현재 "그려지는 과정"을 파일로 저장.

기록중지: Open 되어 있는 파일을 Close.

2) 도구 메뉴



화면지우기: 현재 화면에 그려진 모든 내용을 지움.

일시정지: 불러오기를 통해 재생 되는 과정을 일시정지.

xN 배속 재생: 불러오기를 통해 재생 되는 과정을 배속 재생.

3) 재생바

2. Win API 를 이용한 그림판 프로그램 구현

1) 마우스 좌표에 따라 선을 그린다.

LineTo(), MoveToEx()를 이용해 단순히 두 점 사이의 직선을 그린다.

한번 그려졌던 직선은 더블 버퍼링에 의해 매번 다시 그려져야 한다.

직선을 그리는 데 필요한 점들을 마우스를 드래그할 때마다 Vector 로 저장 및 관리한다.

2) 한 프레임당 그려지는 점들을 벡터에 저장 및 관리한다.

한 프레임이 그려지는(혹은 저장되는) 점들은 최소 1 개 이상이다.

사용자가 마우스 드래그를 하는 속도에 따라, 한 프레임당 저장되는 점의 개수가 다르다.

(천천히 드래그하면 한 프레임당 저장되는 점의 개수가 적고, 반대로 빠르게 드래그하면 한 프레임당 저장되는 점의 개수가 많다.)

나중에 리플레이로서 "그려지는 과정"을 재생하기 위해서는 시간의 개념을 도입해야 한다.

이 시간의 개념을 구현하기 위해 Vector 및 파일에 저장되는 데이터는

마우스 좌표와 현재 프레임 번호를 저장해 준다.

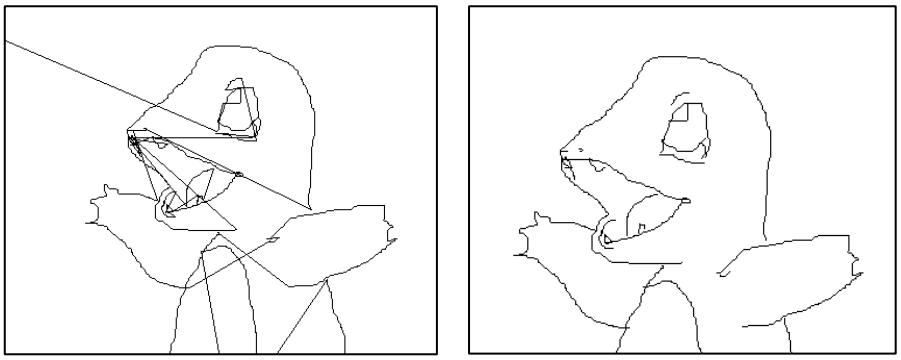
3) Vector 에 저장되는 마우스 좌표는 드래그된 점과 클릭된 점으로 구분한다.

클릭된 점과 드래그된 점으로 구분하는 이유는 두 점 사이의 직선을 그릴 때,

단순히 Vector 에 저장된 모든 점들에 대하여 순차적으로 접근하여 직선을 그리면

선의 시작점과 끝점의 구분없이 모든 점들이 이어져서 그려진다.

이를 구분하기 위해 Vector 에 저장되는 점은 드래그된 점과 클릭된 점으로 따로 구분하여 저장한다.



드래그된 점과 클릭된 점으로 구분하여 그렸을 때 비교

(좌) 구분없이 그림 / (우) 구분하여 그림

3. 파일 입/출력 데이터 정의 및 처리

1) 녹화되기 이전과 이후를 구분하여 파일에 저장한다.

현재 Vector 에 저장되어 있는 데이터들은 각각 생성됐을 때의 프레임 번호를 가지고 있다.
이를 파일에 저장하여 나중에 다시 불러와서 프레임 번호에 맞추어 재생시키게 되면
녹화되기 이전 그림의 그리는 과정이 순서대로 재생되게 된다.
녹화된 시점부터 재생되게 하려면 녹화되기 이전 데이터들의 프레임 번호를 전부 0 으로 초기화하여 저장한다. 이렇게 하면 저장된 데이터를 재생할 때 프레임 번호가 0 인 데이터들은
프로그램 시작할 때 다 그려지게 되고, 그 이후부터 현재 재생되는 프레임 번호에 맞추어
데이터를 재생하게 된다.

2) 파일에 저장되어 있는 데이터를 불러와서 재생할 때에는 프레임 번호를 기준으로 재생한다.

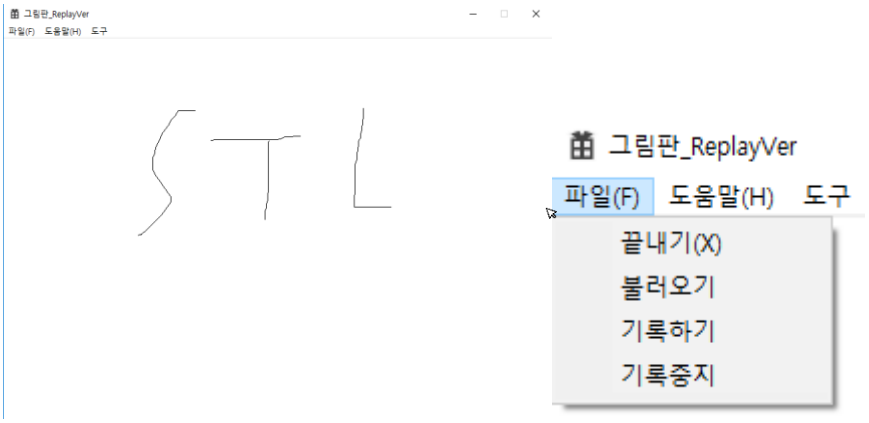
Vector 에 저장되어 있는 데이터를 재생한다라는 의미는 특정 시간간격마다 데이터를
이용해서 렌더링을 함을 뜻한다. 여기서 특정 시간간격은 1/60 초이다.(FPS:60)
응용 프로그램이 시작된 순간부터 지금까지,
한 프레임을 그릴 때마다 프레임을 몇 번 그렸는지를 저장하는 프레임 카운트 변수를
이용한다. **한 프레임에 그려지는 내용은, 저장되어 있는 데이터 중 프레임 카운트와 동일한 값을 갖는 데이터까지만 그린다.**

3) 프로그램의 상태가 "재생중"일 때에는 프레임 오프셋이 가능하도록 한다.

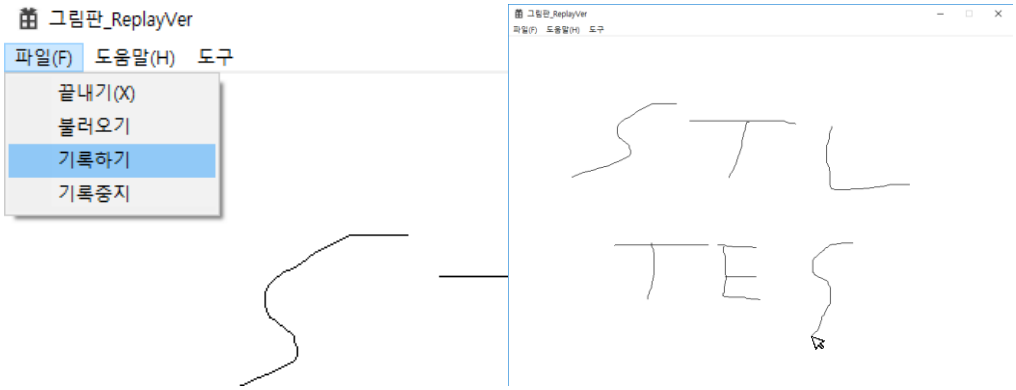
오프셋 수치에 따라 재생되는 프레임을 변경한다.
키보드 <-, -> 입력에 따라 현재 프레임에 ±5 프레임이 재생되도록 한다.
재생바 상에서의 마우스 클릭과 드래그 위치에 따라 현재 프레임을 변경한다.
마우스의 x 값에 따라 재생되어지는 프레임 번호를 변경한다.

현재 프레임 번호 = 전체 재생해야할 프레임 수 * ((마우스 x 값 - 재생바의 왼쪽) / 재생바 크기)

4. 플레이 예시

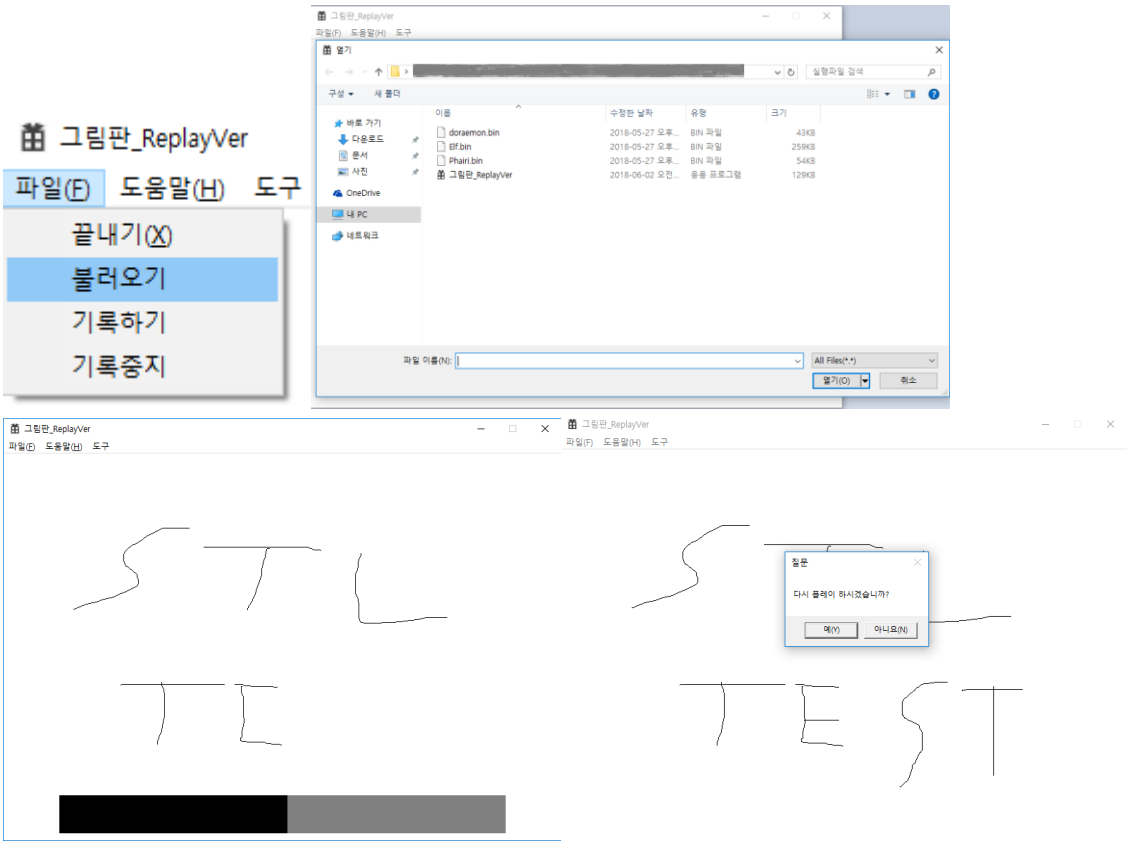


메뉴에서 불러오기를 통해 파일을 가져오지 않으면 마우스 드래그에 따라 선을 그린다.

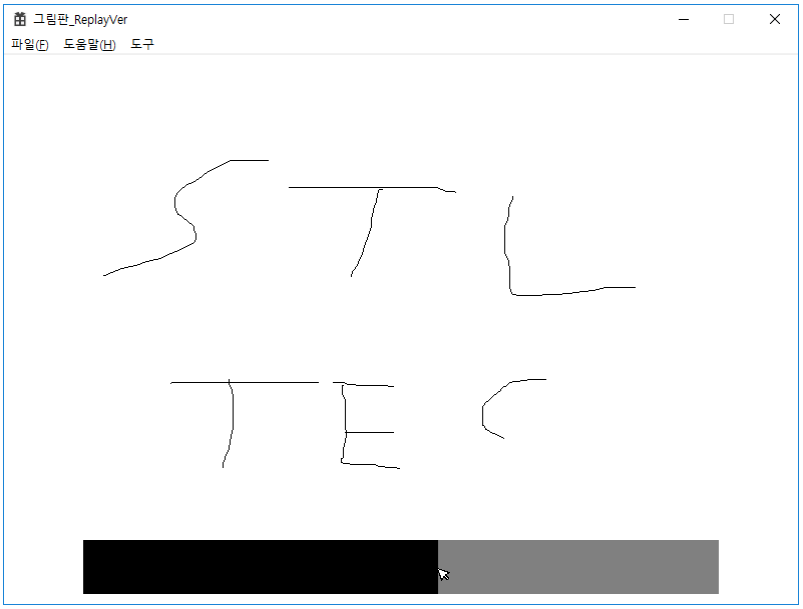


기록하기를 누른 시점부터 파일에 현재 그려진 화면의 내용이 파일로 저장된다.

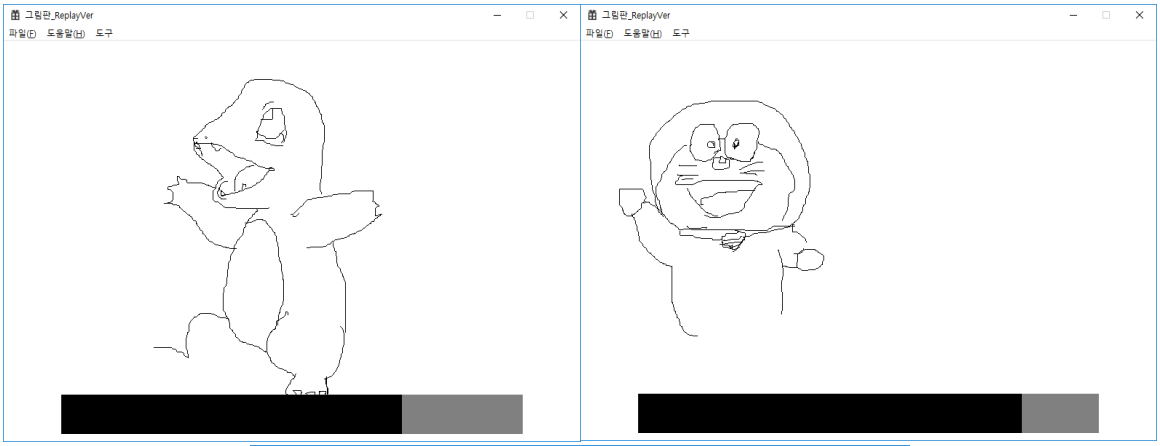
(정확히는 선을 그려내는 모든 점 데이터를 파일로 저장)

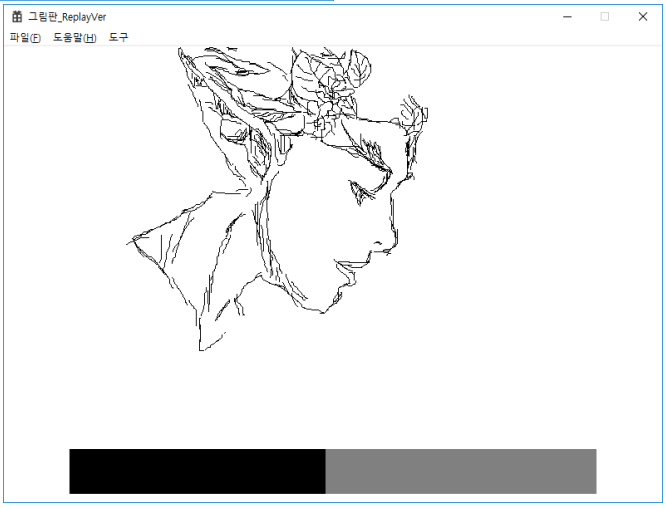


메뉴에서 파일을 불러오면 다음과 같이 재생되고, 재생이 완료되면 리플레이 여부를 물어본다.



재생바를 드래그하거나 화살표키(<-, ->)를 누르면 재생되는 프레임을 시간축에 따라 바꿀 수 있다.





위와 같이 그리는 과정을 불러와서 재생시킬 수 있다.

리뷰

시간이라는 개념을 어떻게 정의할 것인가에 대한 고민을 많이 했던 것 같다.
단순히 프로그램이 시작하고 나서 지금까지의 실행시간을 파일에 저장하는 것은 큰 의미가 없었다. 왜냐하면 프로그램 실행시간에 맞추어 점을 그려내려면, 저장되어 있는 시간과 현재 실행시간이랑 "정확히" 일치하지는 않기 때문이다.
프로그램이 한 프레임을 그려내는 데 걸리는 시간이 매번 다르기 때문에 현재 실행시간이랑 저장되어 있는 시간이 일치하는 경우는 매우 적다. 이를 해결하기 위해 저장되어 있는 시간에 맞추어 현재 프로그램의 실행시간을 조절해야 할 필요가 있었다. 다시 말해 FPS 제어와 프레임 스킵 구현해야 했었다.
CPU 의 처리 속도가 빠르면 빠를수록 프로그램은 빠르게 처리 된다. 하지만 현재 그림판 프로그램의 경우 대개 정해진 플레이 속도가 있기 때문에 CPU 성능에 따라 플레이 속도가 제멋대로 치솟는 일이 있어서는 안된다. 그렇기 때문에 이를 제어 하기 위한 제어 장치가 필요하다. FPS 제어 기법은 바로 위와 같은 상황에서 필요한 기법이다. 물론 그 방법상에는 여러가지 종류의 것들이 있겠지만 이 프로그램을 만들면서 내가 사용한 방법은 프레임을 매번 카운트하는 변수를 두고 그 변수값과 일치하는 데이터에 대해서만 재생시키는 방법이다.
예를 들어, 현재 프레임 카운트의 값이 125 이면 저장되어 있는 데이터 중 0 부터 125 의 프레임 번호를 갖는 데이터까지만 현재 프레임에 그리는 것이다. 엄밀히 따지면 프레임 처리속도를 조절한 것은 아니지만, CPU 나 GPU 의 처리속도를 조절하기 위해 프로그램을 강제로 블로킹을 하는 것보단 처리비용이 적은 방법이다. 만약 재생되는 시간이 실제 시간처럼 정밀해야 한다면 위와 같은 방법이 아니라 프로그램을 블로킹하는 방법을 선택했을 것이다.