cub3D
My first RayCaster with miniLibX

요약:. 이 프로젝트는 세계적으로 유명한 90년대 게임인 FPS에서 영감을 받았습니다. 그러면 레이캐스팅을 탐색할 수 있습니다. 당신의 목표는 미로 안에서 당신의 길을 찾아야 하는 역동적인 경치를 만드는 것이 될 것이다.

Summary: This project is inspired by the world-famous eponymous 90's game, which
was the first FPS ever. It will enable you to explore ray-casting. Your goal will be to
make a dynamic view inside a maze, in which you'll have to find your way

Contents

I Foreword
II Goals
III Common Instructions
IV Mandatory part - cub3D
V Bonus part
VI Examples

Chapter I
Foreword

1992년 아포지 소프트웨어(Apogee Software)가 출판한 Id Software가 개발한 울펜슈타인 3D는 비디오 게임 역사상 최초의 진정한 "1인칭 슈터"이다.

Developed by Id Software by the über famous John Carmack and John Romero, published in 1992 by Apogee Software, Wolfenstein 3D is the first true “First Person Shooter” in the history of video games.

울펜슈타인 3D는 둠(Id Software, 1993), 둠 II(Id Software, 1994), 듀크 누켐(Duke Nukem) 3D(3D Realm, 1996), 퀘이크(Id Software, 1996)와 같은 게임의 조상이다.
이제 네가 역사를 다시 살아날 차례야...

Wolfenstein 3D is the ancestor of games like Doom (Id Software, 1993), Doom II (Id Software, 1994), Duke Nukem 3D (3D Realm, 1996) and Quake (Id Software, 1996), that are additional eternal milestones in the world of video games.
Now, it’s your turn to relive History...

Chapter II
Goals

이 프로젝트의 목표는 엄격함, C 사용, 기본 알고리즘 사용, 정보 연구 등 모든 첫 해 목표와 유사합니다.
그래픽 디자인 프로젝트로서 cub3D를 사용하면 창, 색상, 이벤트, 채우기 모양 등의 영역에서 스킬을 향상시킬 수 있습니다.
This project's objectives are similar to all this first year's objectives: Rigor, use of C, use of basic algorithms, information research etc.
As a graphic design project, cub3D will enable you to improve your skills in these areas: windows, colors, events, fill shapes, etc.

cub3D를 결론짓는 것은 구체적인 내용을 이해할 필요 없이 수학의 장난스러운 실제 응용을 탐구할 수 있는 주목할 만한 놀이터이다.
To conclude cub3D is a remarkable playground to explore the playful practical applications of mathematics without having to understand the specifics.

인터넷에서 이용할 수 있는 수많은 문서의 도움으로, 당신은 우아하고 효율적인 알고리즘을 만들기 위한 도구로 수학을 사용할 것이다.
With the help of the numerous documents available on the internet, you will use mathematics as a tool to create elegant and efficient algorithms.

Chapter III
Common Instructions

• 본 규정에 따라 프로젝트를 작성해야 합니다. 보너스 파일/기능이 있는 경우 표준 검사에 포함되며, 내부에 표준 오류가 있는 경우 0을 받게 됩니다.
• Your project must be written in accordance with the Norm. If you have bonus files/functions, they are included in the norm check and you will receive a 0 if there is a norm error inside.

• 기능이 정의되지 않은 동작 외에 예기치 않게 중단되어서는 안 됩니다 (segmentation fault, bus error, double free, etc). 이 경우 프로젝트가 작동하지 않는 것으로 간주되고 평가 중에 0을 받게 됩니다.
• Your functions should not quit unexpectedly (segmentation fault, bus error, double free, etc) apart from undefined behaviors. If this happens, your project will be considered non functional and will receive a 0 during the evaluation.

• 필요한 경우 할당된 모든 힙의 메모리 공간을 적절히 확보해야 합니다. 누수는 용납되지 않습니다.
• All heap allocated memory space must be properly freed when necessary. No leaks will be tolerated.

• 만약 해당 프로젝트에서 Makefile 을 요구한다면 반드시 소스 파일을 -Wall -Wextra -Werror 플래그를 이용해 컴파일하고 적절한 출력물을 만들 수 있는 Makefile 을 제출해야 합니다. 또한 Makefile 은 리링크 (relink) 되면 안 됩니다.
• If the subject requires it, you must submit a **Makefile** which will compile your source files to the required output with the flags **-Wall, -Wextra** and **-Werror**, and your Makefile must not relink.

• Makefile 은 반드시 최소한 $(NAME), all, clean, fclean 의 규칙을 포함하고 있어야 합니다.
• Your **Makefile** must at least contain the rules **$(NAME), all, clean, fclean** and **re**.

• 해당 프로젝트의 보너스를 제출하기 위해서는 반드시 Makefile 에 필수 지침에서 금지된 다양한 해더, 라이브러리, 금지된 함수를 포함하는 bonus 규칙을 만들어야 합니다. 보너스는 _bonus.{c/h} 의 다른 파일로 존재해야합니다. 필수 지침과 보너스 지침은 각각 개별로 평가됩니다.
• To turn in bonuses to your project, you must include a rule bonus to your Makefile, which will add all the various headers, librairies or functions that are forbidden on the main part of the project. Bonuses must be in a different file _bonus.{c/h}. Mandatory and bonus part evaluation is done separately.

• 만약 프로젝트에서 libft 사용을 허용한다면, 반드시 해당 소스와 그 소스 파일을 컴파일할 수 있는 독립된 Makefile 을 libft 폴더 안에 복사해야 합니다. 프로젝트의 Makefile 은 반드시 독립된 libft 의 Makefile 을 통해 libft 를 컴파일한 후 프로젝트를 컴파일해야 합니다.
• If your project allows you to use your **libft**, you must copy its sources and its associated **Makefile** in a **libft** folder with its associated **Makefile**. Your project's **Makefile** must compile the library by using its **Makefile**, then compile the project.

• 우리는 여러분이 해당 프로젝트에 대한 테스트 프로그램을 만드는 것을 권장합니다. 해당 프로그램은 제출되지 않고 평가되지도 않지만 여러분과 동료의 프로그램을 쉽게 테스트할 수 있게 해줄겁니다. 해당 프로그램은 특히 평가받을 때 유용하게 사용될 것입니다. 또한, 평가를 받을 때, 동료의 테스트 프로그램을 평가에 사용할 수 있습니다.
• We encourage you to create test programs for your project even though this work **won't have to be submitted and won't be graded.** It will give you a chance to easily test your work and your peers' work. You will find those tests especially useful during your defence. Indeed, during defence, you are free to use your tests and/or the tests of the peer you are evaluating.

• 반드시 할당된 깃 레포지토리에 프로그램을 제출하세요. 오직 깃 레포지토리에 올려진 프로그램만 평가될 것입니다. 만약 Deepthought 가 여러분의 프로그램을 평가하게 된다면, 여러분의 피어 평가가 이루어진 후 평가하게 될 것입니다. 만약 Deepthought 가 평가할 때 여러분의 프로그램 어느 부분에라도 에러가 있다면, 평가는 그 즉시 멈출 것입니다.
• Submit your work to your assigned git repository. Only the work in the git repository will be graded. If Deepthought is assigned to grade your work, it will be done after your peer-evaluations. If an error happens in any section of your work during Deepthought's grading, the evaluation will stop.

Mandatory part

Program name	cub3D
Turn in files	All your files
Makefile	all, clean, fclean, re, bonus
Arguments	a map in format *.cub
External functs.	• **open, close, read, write, printf, malloc, free, perror, strerror, exit** • All functions of the **math** library (-lm man man 3 math) • All functions of the **MinilibX**
Libft authorized	Yes
Description	1인칭 관점에서 미로 내부에 대한 "현실적인" 3D 그래픽 표현을 만들어야 합니다. 앞에서 언급한 레이캐스팅 원칙을 사용하여 이 표현을 생성해야 합니다. You must create a “realistic” 3D graphical representation of the inside of a maze from a first person perspective. You have to create this representation using the Ray-Casting principles mentioned earlier.

지켜야 할 사항은 다음과 같습니다.
The constraints are as follows:

• miniLibX 를 **반드시** 사용해야 합니다. 다른 버전이 OS 에 깔려있거나 직접 받아올 수도 있습니다. 하지만 다른 버전을 사용하게 된다면, libft 와 같은 **기본 규칙**을 지켜야 합니다.
• You must use the miniLibX. Either the version that is available on the operating system, or from its sources. If you choose to work with the sources, you will need to apply the same rules for your libft as those written above in Common Instructions part.

• 윈도우의 움직임은 반드시 부드러워야 합니다 (다른 윈도우로 전환할 때, 최소화할 때 등).
• The management of your window must remain smooth: changing to another window, minimizing, etc.

• 벽이 어느 쪽(북, 남, 동, 서)을 향하느냐에 따라 달라지는 서로 다른 벽 질감을 표시합니다.
• Display different wall textures (the choice is yours) that vary depending on which side the wall is facing (North, South, East, West).

• 당신의 프로그램은 벽 대신 아이템(스프라이트)을 표시할 수 있어야 합니다.
• Your program must be able to display an item (sprite) instead of a wall.

• 당신의 프로그램은 바닥과 천장 색상을 서로 다른 두 가지 색으로 설정할 수 있어야 합니다.
• Your program must be able to set the floor and ceilling colors to two different ones.

• 언젠가 **Deepthought** 가 이 프로젝트를 평가할 수 있는 눈을 갖게 되는 날을 위해서, 두 번째 인수가 "--save"일 때 프로그램은 첫 번째 렌더링된 이미지를 bmp 형식으로 저장해야 합니다.
• In case the Deepthought has eyes one day to evaluate your project, your program must save the first rendered image in bmp format when its second argument is "--save".

• 두 번째 인수가 제공되지 않는 경우 프로그램은 이미지를 창에 표시하고 다음 규칙을 준수합니다.
• If no second argument is supllied, the program displays the image in a window and respects the following rules:

  ◦ 키보드의 왼쪽과 오른쪽 화살표 키를 사용하여 미로를 왼쪽과 오른쪽으로 볼 수 있어야 합니다.
  ◦ The left and right arrow keys of the keyboard must allow you to look left and right in the maze.

◦ W, A, S, D 키를 사용하여 미로를 통해 시야를 이동할 수 있어야 합니다.
◦ The W, A, S and D keys must allow you to move the point of view through the maze.

◦ **ESC**를 누르면 창이 닫히고 프로그램이 완전히 종료됩니다.
◦ Pressing **ESC** must close the window and quit the program cleanly.

◦ 프로그람 창의 빨간 십자가(=닫기) 를 클릭하면 창이 닫히고 프로그램이 깨끗하게 종료됩니다.
◦ Clicking on the red cross on the window's frame must close the window and quit the program cleanly.

◦ 맵에서 선언된 화면 크기가 디스플레이 해상도보다 크면 현재 디스플레이 해상도에 따라 윈도우 크기가 설정됩니다.
◦ If the declared screen size in the map is greater than the display resolution, the window size will be set depending to the current display resolution.

◦ **minilibX**의 **Images**를 사용하는 것이 좋습니다.
◦ The use of **images** of the **minilibX** is strongly recommended.

• 프로그램은 반드시 첫 번째 인자로 장면이 정의되어있는 **.cub** 확장자명의 파일을 받아야 합니다.
• Your program must take as a first argument a scene description file with the .cub extension.

◦ 지도는 빈 공간이면 0, 벽이면 1, 아이템이면 2, 플레이어의 시작 위치 및 스폰 방향에 N, S, E 또는 W의 가능한 4자만 구성해야 합니다.
◦ The map must be composed of only 4 possible characters: 0 for an empty space, 1 for a wall, 2 for an item and N,S,E or W for the player's start position and spawning orientation.

This is a simple valid map:

```
111111
100101
102001
1100N1
111111
```

◦ map은 반드시 벽으로 폐쇄된/둘러싸여야 합니다. 만약 그렇지 않다면, 프로그램은 에러를 반환해야합니다.
◦ The map must be closed/surrounded by walls, if not the program must return an error.

◦ map content를 제외하고 각 요소 유형을 하나 또는 그 이상의 빈 줄로 구분할 수 있습니다.
◦ Except for the map content, each type of element can be separated by one or more empty line(s).

◦ 항상 마지막이어야 하는 map content을 제외하고, 각 요소 유형을 파일의 순서에 상관없이 설정할 수 있습니다.
◦ Except for the map content which always has to be the last, each type of element can be set in any order in the file.

◦ map을 제외하고 요소의 각 정보 유형은 하나 또는 그 이상의 공백으로 구분될 수 있습니다.
◦ Except for the map, each type of information from an element can be separated by one or more space(s).

◦ map은 반드시 파일에 있는 것처럼 파싱되어야 합니다. 공백은 맵에서 유효한 부분이며 사용자가 처리할 수 있습니다. 지도 규칙을 존중하는 한 모든 종류의 지도를 파싱할 수 있어야 합니다.
◦ The map must be parsed as it looks like in the file. Spaces are a valid part of the map, and is up to you to handle. You must be able to parse any kind of map, as long as it respects the maps rules.

○ 각 요소(map 제외)가 먼저 정보를 유형 식별자(하나 또는 두 개의 문자로 구성됨)로 구성되고 그 다음이 표시됩니다.
○ Each element (except the map) firsts information is the type identifier (composed by one or two character(s)), followed by

＊ 해상도 Resolution:

```
R 1920 1080
```

· identifier: **R**
· x render size
· y render size

＊ 북쪽 텍스쳐 North texture:

```
NO ./path_to_the_north_texture
```

· identifier: NO
· path to the north texure

＊ 남쪽 텍스쳐 South texture:

```
SO ./path_to_the_south_texture
```

· identifier: **SO**
· path to the south texure

＊ 서쪽 텍스쳐 West texture:

```
WE ./path_to_the_west_texture
```

· identifier: **WE**
· path to the west texure

＊ East texture:

```
EA ./path_to_the_east_texture
```

· identifier: **EA**
· path to the east texure

∗ 스프라이트 텍스쳐 Sprite texture:

```
S ./path_to_the_sprite_texture
```

· identifier: **S**

· path to the sprite texure

∗ 바닥 색상 Floor color:

```
F 220,100,0
```

· identifier: **F**

· R,G,B colors in range [0,255]: **0, 255, 255**

∗ 천장 색상 Ceilling color:

```
C 225,30,0
```

· identifier: **C**

· R,G,B colors in range [0,255]: **0, 255, 255**

◦ 메인 파트에서의 최소한의 .cub 장면의 예제

◦ Example of the mandatory part with a minimalist **.cub** scene:

```
R 1920 1080
NO ./path_to_the_north_texture
SO ./path_to_the_south_texture
WE ./path_to_the_west_texture
EA ./path_to_the_east_texture
S ./path_to_the_sprite_texture
F 220,100,0
C 225,30,0
        1111111111111111111111111
        1000000000110000000000001
        1011000001110000002000001
        1001000000000000000000001
111111111011000001110000000000001
100000000011000001110111111111111
11110111111111011100000010001
11110111111111011101010010001
11000000110101011100000010001
10002000000000001100000010001
10000000000000001101010010001
11000001110101011111011110N0111
11110111 1110101 101111010001
11111111 1111111 111111111111
```

◦ 파일에 잘못된 구성이 있는 경우 프로그램이 제대로 종료되고 "Error\n"을 반환한 후 사용자가 선택한 오류 메시지가 명시적이어야 합니다.

◦ If any misconfiguration of any kind is encountered in the file, the program must exit properly and return "Error\n" followed by an explicit error message of your choice.

Chapter V
Bonus part

보너스는 **mandatory part**가 완벽한 경우에만 평가됩니다.
완벽(PERFECT)이라 우리는 이것이 완전하다고 정의하고 있으며, 잘못된 사용 등과 같은 끔찍한 실수들에도 실패하지 않아야 한다는 것을 의미한다.
기본적으로, 이것은 당신의 **mandatory part**를 채점하는 동안 모든 포인트를 획득하지 못하면 당신의 보너스는 완전히 무시된다는 것을 의미합니다.
Bonuses will be evaluated only if your mandatory part is PERFECT.
By PERFECT we naturally mean that it needs to be complete, that it cannot fail, even in cases of nasty mistakes like wrong uses etc.
Basically, it means that if your mandatory part does not obtain ALL the points during the grading, your bonuses will be entirely IGNORED.

Bonus list:
- 벽 충돌. Wall collisions.
- 스카이박스. A skybox.
- 바닥 및/또는 천장 텍스처. Floor and/or ceiling texture.
- An HUD.
- 위아래를 보는 능력. Ability to look up and down.
- 점프 또는 웅크리기. Jump or crouch.
- 거리 관련 그림자 효과. A distance related shadow effect.
- Life bar.
- 미로에 있는 더 많은 아이템들. More items in the maze.
- 개체 충돌. Object collisions.
- 객체/트랩을 주워 포인트를 획득하거나 라이프를 잃습니다.

Earning points and/or losing life by picking up objects/traps.
- 열고 닫을 수 있는 문. Doors which can open and close.
- 비밀의 문. Secret doors.
- 총 쏘는 애니메이션 또는 스프라이트

Animations of a gun shot or animated sprite.
- 몇 가지 레벨. Several levels.
- 소리와 음악. Sounds and music.
- 마우스로 시점을 회전합니다. Rotate the point of view with the mouse.

• 무기와 나쁜 놈들과 싸우기! Weapons and bad guys to fight!

모든 보너스 포인트를 획득하려면 최소한 14개의 보너스 포인트를 검증해야 합니다. 그러니 현명하게 선택하되 시간을 낭비하지 않도록 주의하십시오!
To earn all bonus points you need to validate at least 14 of them, so choose wisely but be careful to not waste your time!

평가 중에 보너스 부분의 사용이 정당화되면 다른 함수를 사용하여 보너스 부분을 완료할 수 있습니다. 또한 필요에 맞게 예상된 장면 파일 형식을 수정할 수 있습니다.
스마트하게!
You are allowed to use other functions to complete the bonus part as long as their use is justified during your evaluation. You are also allowed to modify the expected scene file format to fit your needs.
Be smart!