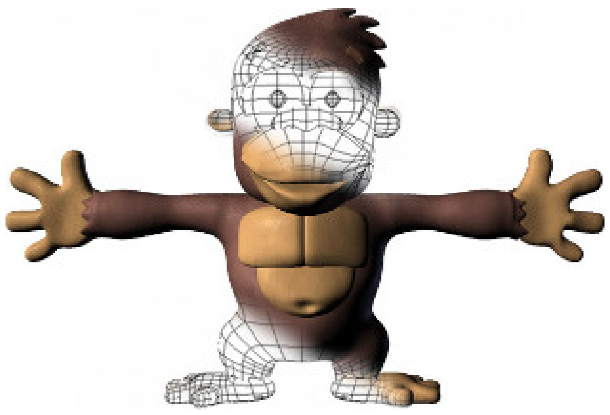

B1- C Graphical Programming

B-MUL-055

Wolf3D

First-Person Shooting Game

v1.3

Wolf3D

First-Person Shooting Game

binary name:	wolf3d
repository name:	wolf3d
repository rights:	ramassage-tek
language:	C
group size:	1
compilation:	via Makefile, including re, clean and fclean rules

- Your repository must contain the totality of your source files, but no useless files (binary, temp files, obj files,...).
- All the bonus files (including a potential specific Makefile) should be in a directory named *bonus*.
- Error messages have to be written on the error output, and the program should then exit with the 84 error code (0 if there is no error).

This project contains a bonus section, which includes a non-exhaustive list of all bonuses you may implement in addition to the other mandatory sections.

You **have to** implement more than the mandatory sections in order to validate the unit.

This project consists of creating a first-person view inside of a maze. You must program the view and the camera's movement on the map in perspective and use the raycasting design concept.

In order to do this, you must be able to load the map file that contains at least one of the following: one or several mazes, a starting position, the position of various game elements.
Feel free to use any map format that you want. We would, however, encourage you to create several unique formats that will allow you to exchange maps.

We must be able to move throughout the map, in real time, by using the keyboard keys or potentially the mouse.
The wall should change color, or even display a texture, depending on the direction.
You should install a procedure that will neatly cut your program: pressing a button, a menu etc.
The method is up to you.

Bonuses

The following list of bonuses is non-exhaustive:

- can't run into walls,
- a scrolling sky texture when running,
- a floor or ceiling texture that complies with perspective,
- animated textures,
- objects in the maze,
- can't run into certain objects,
- certain objects can be picked up,
- certain objects are intelligent beings (monster etc.),
- there are doors that can be opened/closed,
- opening/closing is animated,
- anti-aliasing,
- light, depending on the distance from the person,
- light, depending on the distance from the light type objects,
- the maps have angled walls,
- the maps show the stairs, holes and height,
- the maps show inclined planes,
- the head can be raised/lowered,
- the eye can be moved to turn the mouse,
- the character can jump (only accessible if map shows height),
- third-person view,
- sniper mode (zoom),
- 360 vision,
- game elements: lives, armor, munitions, keys, leveling,
- shooting weapon/ hand-to-hand combat weapon,
- game elements coming from dynamic library,
- mini-map,
- compass,
- key editing via the game itself,
- menu,
- cinematic intro,
- game over screen,
- transition between the different menus or game levels,
- network game,
- split-screen game,
- campaign mode (At least 5 levels),
- procedural generation of levels (not random),
- different game modes (classic, race, capture the flag if multiplayer),
- sound effects,
- music adapted to situations/levels,
- using accelerometer ang gyro embedded in your PC.

Authorized Functions

- C Math library (-lm)
- open,
- close,
- read,
- write,
- malloc,
- free,
- socket, close, bind, sendto, recvfrom,
- all libdl's functions,

CSFML functions:

- sfRenderWindow_isOpen,
- sfRenderWindow_pollEvent,
- sfRenderWindow_waitEvent,
- sfRenderWindow_clear,
- sfRenderWindow_drawSprite,
- sfRenderWindow_display,
- sfRenderWindow_create,
- sfRenderWindow_destroy,
- sfRenderWindow_close,
- sfTexture_create,
- sfTexture_updateFromPixels,
- sfTexture_destroy,
- sfSprite_create,
- sfSprite_setTexture,
- sfSprite_destroy,
- all of System module's functions,
- all of Window module's functions,
- all of Audio module's functions.

Autograder

In order to enable your work to be automatically graded, the implementation of the following function is required:

```
void my_put_pixel(t_my_framebuffer* framebuffer, int x, int y, sfColor color);
void my_draw_line(t_my_framebuffer* framebuffer, sfVector2i from, sfVector2i to, sfColor color);
```

my_put_pixel puts a color pixel in framebuffer at the *x* and *y* positions.
my_draw_line draws a line between *from* to *to* of the color *color*.

These functions must be found in files called my_put_pixel.c and my_draw_line.c, located in an **src/** folder at the root of your repository.

If your files include header (.h) files, make sure to place them in the root of your repository, either in an include or inc folder.