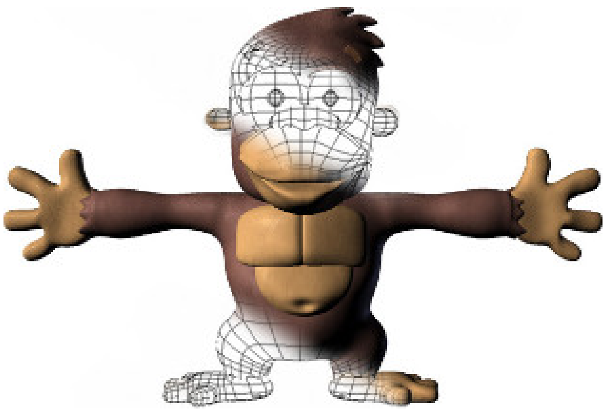


B1- C Graphical Programming

B-MUL-055

WireFrame

Parallel projection for 3d imagery

v1.3

WireFrame

Parallel projection for 3d imagery

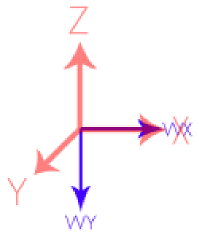
binary name:	wireframe
repository name:	wireframe
repository rights:	ramassage-tek
language:	C
group size:	1
compilation:	via Makefile, including re, clean and fclean rules

- 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.

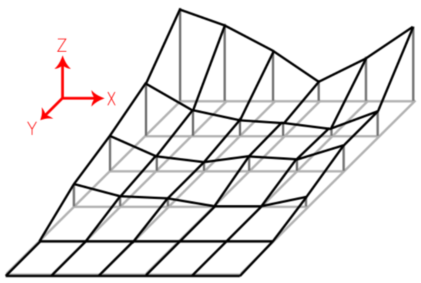
You **have to** implement more than the mandatory sections in order to validate the unit.

This project consists of displaying the relief of a terrain, in an on-screen graphic window, by using a parallel projection. The terrain, itself, is contained in a `.wf` file that is passed as parameter to the program.



The first step consists of displaying a flat terrain by using a **3D projection**. A 3D projection is the interpretation of 3D coordinates (X, Y, Z), with 2D coordinates (WX,WY), and, therefore, is representable on the screen.

The second step consists of loading the file and applying the information that is found there to your terrain. In this way, instead of a flat terrain, the file's contents will be displayed.



The program should exit with ESC.

Scene description file

Your program must take a file, representing the scene, as first argument.

```
~/B-MUL-055> ./wireframe file.wf
```

The file will look like this:

```
~/B-MUL-055> cat -e file.wf
6,5,3,2,3,4,$
4,2,1,1,1,2,$
3,1,1,1,1,2,$
2,1,1,0,0,1,$
0,0,0,0,0,0,$
0,0,0,0,0,0$
```

Each number corresponds to the altitude (Z coordinate), while its position in the array corresponds to the X and Y coordinates.

This is an example, you are free to improve it!

Bonuses

The following list of bonuses is non-exhaustive, but should give you an idea of how this project can be customized:

- anti-aliasing,
- color gradient on the lines between two different-colored dots,
- rounded angles,
- zoom,
- rotation (even partial) around the X axis,
- rotation (even partial) around the Z axis,
- editing the execution of a dot's altitude,
- translation of 3D objects,
- rotation of 3D objects,
- terrain animation,
- filling in shapes with color,
- color gradation on the parellelograms between four different-colored dots.

You can find more bonuses!

Authorized Functions

- C Math library (-lm)
- open,
- close,
- read,
- write,
- malloc,
- free,

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 functions 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);
sfVector2i my_parallel_projection(sfVector3f pos3d, float angle);
sfVector2i my_isometric_projection(sfVector3f pos3d);
```

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*.
my_parallel_projection returns the parallel projection coordinates of the 3D point *pos3d* of angle *angle*.
my_isometric_projection returns the isometric projection coordinates of the 3D point *pos3d*.

The absence of these functions will make grading impossible, and you will receive a 0.5 grade.

These functions must be found in files called my_put_pixel.c, my_draw_line.c, my_parallel_projection.c and my_isometric_projection.c. These files must 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 an *inc* folder.

my_put_pixel **and** my_draw_line **must** be functional.