SO LONG

**Суть проекта, основная и бонусная часть**

**Как подключать либу**

Because the miniLibX uses Appkit and MX11 you need to link them adding:

 -lmlx -framework OpenGL -framework AppKit when you compile.

gcc -Wall -Wextra -Werror -Llibmlx -lmlx -framework OpenGL -framework AppKit solong.c

-L путь где лежит либа

-l далее имя либы без lib

# include "./mlx/mlx.h"

-Lmlx -lmlx -framework OpenGL -framework AppKit

**Структуры, которые используем и их заполнение**

typedef struct s\_game\_map

{

int map\_height;

int map\_length;

int player\_position\_x;

int player\_position\_y;

int game\_score;

int max\_score;

char \*\*map\_data;

int game\_over;

int steps;

void \*end\_img;

void \*wall;

void \*exit;

void \*player;

void \*floor;

void \*items;

void \*enemy;

int \*enemypos;

} t\_game\_map;

typedef struct s\_render\_v {

void \*mlx;

void \*win;

} t\_render\_v;

vars->mlx = mlx\_init();

vars->win = mlx\_new\_window(vars->mlx, ww, wh, "So\_long");

**Алгорит исполнения по блокам**

-карта

*Считывание*

*Верификация:*

*-прямоугольник*

*-окружена препятствиями*

*-наличие хотя бы одного выхода и позиции игрока*

*-наличие других символов на карте*

*-имя передаваемой карты \*.ber*

-Получение данных и заполнение структуры

void \*mlx\_xpm\_file\_to\_image(void \*mlx\_ptr, char \*filename,

int \*width, int \*height);

void ft\_wall\_image(t\_render\_v \*\*vars, t\_game\_map \*\*map)

{

int pos[2];

(\*map)->wall = mlx\_xpm\_file\_to\_image((\*vars)->mlx, "./textures/wall.xpm", &pos[0], &pos[1]);

}

Создает изображение, содержащее изображение .xpm, найденное в relative\_path, и сохраняет его ширину и высоту в пикселях для заданных указателей. Вы можете использовать изображения только в формате xpm.

-Отрисовка карты

int mlx\_put\_image\_to\_window(void \*mlx\_ptr, void \*win\_ptr, void \*img\_ptr,

int x, int y);

Чтобы нарисовать изображение в окне, вам просто нужно вызвать mlx\_put\_image\_to\_window ().

Передача указателя на окно, изображение и позицию. Координаты (0,0) окна и изображений находятся в верхнем левом углу.

mlx\_put\_image\_to\_window((\*vars)->mlx, (\*vars)->win, (\*map)->floor, x, y);

if (t == 'P')

mlx\_put\_image\_to\_window((\*vars)->mlx, (\*vars)->win, (\*map)->player, x,y);

if (t == '1')

mlx\_put\_image\_to\_window((\*vars)->mlx, (\*vars)->win, (\*map)->wall, x, y);

if (t == 'C')

mlx\_put\_image\_to\_window((\*vars)->mlx, (\*vars)->win, (\*map)->items, x, y);

if (t == 'E')

mlx\_put\_image\_to\_window((\*vars)->mlx, (\*vars)->win, (\*map)->exit, x, y);

x, y \*100

-mlx функции и захват событий

int mlx\_hook(void \*win\_ptr, int x\_event, int x\_mask,

int (\*funct)(), void \*param);

Some useful events:

|  |  |
| --- | --- |
| 2 | Key pressed |
| 3 | Key released |
| 4 | Mouse pressed |
| 5 | Mouse released |
| 6 | Motion |
| 17 | Window closed |

mlx\_hook(vars->win, 2, 0, key\_hook, &vars);

mlx\_hook(vars->win, 17, 0, exit\_hook, vars);

mlx\_loop(vars->mlx);

static int exit\_hook(t\_render\_v \*\*vars)

{

ft\_free(vars);

exit(0);

}

-движения игрока (wasd)

if (keycode == 1)

return (ft\_moving\_up\_down(game\_map, 1));

if (keycode == 13)

return (ft\_moving\_up\_down(game\_map, -1));

if (keycode == 2)

return (ft\_moving\_left\_right(game\_map, 1));

if (keycode == 0)

return (ft\_moving\_left\_right(game\_map, -1));

-признаки окончания игры

if (keycode == 53)

{

mlx\_destroy\_window((\*vars)->mlx, (\*vars)->win);

ft\_free\_stuff(vars);

exit(0);

return (0);

}

Сбор коллекционных предметов и позиция выхода.

**Бонусная часть**

-враги и выведение числа шагов на экран

go = itoa()

int mlx\_string\_put(void \*mlx\_ptr, void \*win\_ptr, int x, int y, int color,

char \*string);

mlx\_string\_put((\*vars)->mlx, (\*vars)->win, 20, 20, 0x000000ff, go);

Текстуры, подбор размера/конвертация