

**Министерство образования и науки Российской Федерации**

**Федеральное агентство по образованию**

**Государственное образовательное учреждение высшего профессионального образования**

**«Московский государственный технический университет имени Н.Э. Баумана»**

**(МГТУ им. Н.Э. Баумана)**

**Факультет «Робототехника и комплексная автоматизация» (РК)**

**Кафедра «Системы автоматизированного проектирования» (РК6)**

****

**Отчет по лабораторной работе рубежного контроля №1 по курсу**

**«Программирование графических приложений»**

**Студент: Сергеева Диана**

**Группа:** РК6-46Б

**Преподаватель:** Волосатова Т.М.

Проверил:

Дата:

2021 год

**XGUI вариант W15**

**Задание:**

Разработать графическую для интерактивной расстановки 5 ферзей на шахматной доске и анализа всех полей, которым они угрожают. Основное графическое окно программы должно отображать конфигурацию шахматной доски и текущее расположение ферзей в её клетках. Каждый ферзь условно изображается квадратной ячейкой в центре своей клетки, а все поля, которым угрожают ферзи, должны отличаться цветом фона от тех полей, которым они не угрожают. При этом при во время передвижения ферзя указатель мыши не должен выходить за рамку его ячейки и шахматной доски. При опускании зажатой кнопки мыши ферзь должен занять свободную клетку, которая имеет максимальную площадь перекрытия с его ячейкой. После этого производится анализ угроз для клеток доски в новой позиции и необходимое изменение цвета их фона. Передвижение ферзей в занятые клетки шахматной доски должно блокироваться автоматическим возвратом в исходную позицию. Нажатие клавиши Esc на клавиатуре должно восстанавливать начальный вариант расстановки ферзей. Завершение программы должен обеспечивать щелчок любой из кнопок мыши по свободной клетке шахматной доски. При разработке программы все клетки шахматной доски и ячейки ферзей должны быть реализованы графическими окнами с соответствующей обработкой событий и захватов указателя мыши в них библиотечными функциями программного интерфейса Xlib X WindowSystem.

**Описание алгоритма:**

Настраиваем графические параметры в функции xcustom, открывая соединение с X-сервером, располагаем фигуры – черные квадраты на шахматной доске так, что они угрожают всем ячейкам на доске. Каждый раз проверяем события, которые происходят в игре, и либо перерисовываем поле, либо прерываем программу при нажатии любой кнопки мыши, либо корректируем позиции фигуры. Диспетчер событий также обрабатывает нажатия клавиш на клавиатуре.

**Входные данные:**

На вход в программу ничего не подаётся.

**Выходные данные:**

Шахматная доска с 5 ферзями на поле так, что они угрожают каждой ячейке доски.

**Текст программы:**

**xchess.h**

#ifndef RK1\_XPAT\_H

#define RK1\_XPAT\_H

#include <X11/Xlib.h>

#include <X11/Xutil.h>

#define ROWS 8

#define COLS 8

#define CELLSIZE 100

#define FIGURESIZE 60

#define UNDERRATACK 1

#define NOTUNDERRATACK 0

typedef struct cell{

Window window;

short figure;

unsigned short type;

} cell;

/\*xpat1.c\*/

int xcustom();

int do\_step(XEvent \*);

int dispatch();

int redraw();

int key\_analiz(XEvent\*);

int main();

/\*xpat2.c\*/

int relink(cell \*\*);

int set\_attack(int, int );

int reattack();

int CheckRules(int , int , int , int );

int desk\_setter();

#endif //RK1\_XPAT\_H

**xpat1.c**

#include <X11/Xlib.h>

#include <unistd.h>

#include <X11/Xutil.h>

#include <X11/keysymdef.h>

#include <stdlib.h>

#include "xchess.h"

static Display \*dpy; //Адрес дисплейной структуры

static GC gc;

static Window desk; //Окно игрового поля программы

static cell\*\* box; //Адрес массива NYxNX клеток

static int done = 0;

static unsigned long colors[2];

//Настройка графических параметров

int xcustom() {

int x, y; //Позиции окон

unsigned w, h; //Габариты окон

int depth = DefaultDepth(dpy, 0); //Глубина экрана 0

Window root; //Корневое окно экрана

XSetWindowAttributes attr; //Атрибуты окон

unsigned long amask; //Маска оконных атрибутов

XSizeHints hint; //Геометрия оконного менеджмента

int i, j; //Индексы окон

Colormap colormap; //Карта цветов

//Настройка графических контекстов

root = DefaultRootWindow(dpy); //Корневое окно экрана

gc = XCreateGC(dpy, root, 0, NULL);

//Настройка цветов палитры

colormap = XDefaultColormap(dpy, DefaultScreen(dpy));

XColor RGBpixel;

XParseColor(dpy, colormap, "#ffffff", &RGBpixel); //White

XAllocColor(dpy, colormap, &RGBpixel);

colors[0] = RGBpixel.pixel;

XParseColor(dpy, colormap, "#8A8BEE", &RGBpixel); //Pink

XAllocColor(dpy, colormap, &RGBpixel);

colors[1] = RGBpixel.pixel;

//Настройка игрового окна программы

attr.override\_redirect = False; //WM обрамление окна

attr.background\_pixel = colors[0]; //0xFFFFFF - white

amask = (CWOverrideRedirect | CWBackPixel );

//Габариты игрового окна

w = COLS \* CELLSIZE;

h = ROWS \* CELLSIZE;

//Начальные координаты окна игры

x = 0;

y = 0;

desk = XCreateWindow(dpy, root, x, y, w, h, 1, depth, InputOutput,

CopyFromParent, amask, &attr); //Геометрические рекомендации оконного менеджера

hint.flags = (PMinSize | PMaxSize | PPosition);

//Фиксируем габариты и позицию окна игрового поля в свойстве WM

hint.min\_width = hint.max\_width = w;

hint.min\_height = hint.max\_height = h;

hint.x = x;

hint.y = y;

XSetNormalHints(dpy, desk, &hint);

XStoreName(dpy, desk, "xchess"); //Заголовок окна

//Настройка окон клеток

amask = CWOverrideRedirect | CWBackPixel | CWEventMask | CWBackingStore;

attr.override\_redirect = True; //Отмена обрамления окна

attr.background\_pixel = XWhitePixel(dpy, XDefaultScreen(dpy));

attr.backing\_store = Always; //Сохранение конфигурации окна при заслонении

attr.event\_mask = (KeyPressMask | ExposureMask | ButtonPressMask);

//Габариты окна клетки

w = CELLSIZE;

h = CELLSIZE;

x = 0;

y = 0;

box = (cell \*\*)calloc(ROWS, sizeof(cell\*));

for(i = 0; i < ROWS; i++) { //Цикл по рядам клеток

box[i] = (cell \*) calloc(COLS, sizeof(cell));

x = 0;

for (j = 0; j < COLS; j++) { //Создать окна клеток

box[i][j].window = XCreateWindow(dpy, desk, x, y, w-1, h-1, 1, depth,

InputOutput, CopyFromParent, amask,

&attr); //Отображение всех окон на экране \*/

box[i][j].type = NOTUNDERRATACK; //Значение по умолчанию не под атакой

box[i][j].figure = 0;

x += CELLSIZE;

}

y += CELLSIZE;

}

XMapWindow(dpy, desk);

XMapSubwindows(dpy, desk);

return(0)

;}

//Диспетчер событий

int dispatch()

{

XEvent \*event = calloc(1, sizeof(XEvent)); //Структура событий

//Флаг выхода

while(done == 0) { //Цикл обработки событий

XNextEvent(dpy, event); //Чтение событий

switch(event->type) {

case Expose:

redraw(); //Перерисовка

break;

case ButtonPress:

done = do\_step(event);

break; //Позиционирование

case KeyPress:

done = key\_analiz(event);

break; //Прерывание

default: break;

}

}

free(event);

return(0);

}

//Ход игры или корректировка позиций

int do\_step(XEvent\* ev)

{

int cur\_time = CurrentTime;

Window root, child;

int root\_x\_return, root\_y\_return, win\_x\_return, win\_y\_return;

unsigned mask\_return;

//Опрос позиции курсора

XQueryPointer(dpy, desk, &root, &child, &root\_x\_return, &root\_y\_return, &win\_x\_return, &win\_y\_return, &mask\_return);

int start\_y = win\_y\_return / CELLSIZE;

int start\_x = win\_x\_return / CELLSIZE;

if (box[start\_y][start\_x].figure == 0){

return 1;

}

XGrabPointer(dpy, desk, True, 0

, GrabModeAsync, GrabModeAsync, desk, None, cur\_time);

//Создание окна фигуры

XSetWindowAttributes attr;

unsigned long amask = CWOverrideRedirect | CWBackPixel | CWEventMask;

attr.override\_redirect = True; /\* Отмена обрамления окна \*/

attr.background\_pixel = XBlackPixel(dpy, XDefaultScreen(dpy));

attr.event\_mask = (KeyPressMask | ExposureMask | ButtonPressMask);

Window Figure = XCreateWindow(dpy, desk, win\_x\_return - FIGURESIZE/2, win\_y\_return - FIGURESIZE/2, FIGURESIZE, FIGURESIZE, 0, DefaultDepth(dpy, DefaultScreen(dpy)),

InputOutput, CopyFromParent, amask,

&attr);

//Создание фигуры для drag&drop

XMapWindow(dpy, Figure);

unsigned long mask\_return\_past = mask\_return;

while (mask\_return == mask\_return\_past){

XQueryPointer(dpy, desk, &root, &child, &root\_x\_return, &root\_y\_return, &win\_x\_return, &win\_y\_return, &mask\_return);

XMoveWindow(dpy, Figure, win\_x\_return - FIGURESIZE/2, win\_y\_return - FIGURESIZE/2);

}

XDestroyWindow(dpy, Figure);

//Проверка правил

CheckRules(start\_x, start\_y, win\_x\_return/CELLSIZE, win\_y\_return/CELLSIZE);

XUngrabPointer(dpy, cur\_time);

return 0;

}

//Перерисовка состояния игры на карте

int redraw(){

for (int i = 0; i < ROWS; i++)

for (int j = 0; j < COLS; j++)

{

XSetForeground(dpy, gc, colors[box[i][j].type]);

XFillRectangle(dpy, box[i][j].window, gc, 0, 0, CELLSIZE, CELLSIZE);

if (box[i][j].figure != 0) {

XSetForeground(dpy, gc, BlackPixel(dpy, DefaultScreen(dpy)));

XFillRectangle(dpy, box[i][j].window, gc, 20, 20, FIGURESIZE, FIGURESIZE);

}

}

return 0;

}

//Анализ нажатых клавиш

int key\_analiz(XEvent\* ev){

if (ev->xkey.keycode == XKeysymToKeycode(dpy, XK\_Escape)) {

desk\_setter();

redraw();

return 0;;

}

return 0;

}

int main() {

dpy = XOpenDisplay(NULL);

xcustom();

relink(box);

desk\_setter();

dispatch();

XFreeGC(dpy, gc);

XDestroySubwindows(dpy, desk);

XDestroyWindow(dpy, desk);

XCloseDisplay(dpy);

for(int i = 0; i < ROWS; ++i)

free(box[i]);

free(box);

return(0);

}

**xpat2.c**

#include "xchess.h"

static cell \*\*box;

//Получение ссылки на одноименный массив структур из дисплейного модуля

int relink(cell \*\*boxing)

{

box = boxing;

return 0;

}

//Функция атаки одиночной фигуры

int attack(int start\_x, int start\_y, int end\_x, int end\_y)

{

if ((end\_x - start\_x == 0) || (end\_y - start\_y == 0) || (end\_x - start\_x == end\_y - start\_y) || (end\_x - start\_x == start\_y - end\_y))

return 0;

return 1;

}

//Установка полей атаки для отдельной фигуры

int set\_attack(int y, int x)

{

int ny, nx;

for (ny = y, nx = x+1; nx < COLS; nx++)

box[ny][nx].type = UNDERRATACK;

for (ny = y+1, nx = x; ny < 8; ny++)

box[ny][nx].type = UNDERRATACK;

for (ny = y, nx = x-1; nx >= 0; nx--)

box[ny][nx].type = UNDERRATACK;

for (ny = y-1, nx = x; ny >= 0; ny--)

box[ny][nx].type = UNDERRATACK;

for (ny = y+1, nx = x+1; (nx < 8) && (ny < 8); nx++, ny++)

box[ny][nx].type = UNDERRATACK;

for (ny = y+1, nx = x-1; (nx >= 0) && (ny < 8); nx--, ny++)

box[ny][nx].type = UNDERRATACK;

for (ny = y-1, nx = x+1; (nx < 8) && (ny >= 0); nx++, ny--)

box[ny][nx].type = UNDERRATACK;

for (ny = y-1, nx = x-1; (nx >= 0) && (ny >= 0); nx--, ny--)

box[ny][nx].type = UNDERRATACK;

return 0;

}

//Перерисовка полей атаки фигур

int reattack()

{

for (int i = 0; i < ROWS; i++)

for (int j = 0; j < COLS; j++)

box[i][j].type = NOTUNDERRATACK;

for (int i = 0; i < ROWS; i++)

for (int j = 0; j < COLS; j++)

if (box[i][j].figure == 1)

set\_attack(i,j);

return 0;

}

//Проверка правил изменения позиции фигуры

int CheckRules(int start\_x, int start\_y, int end\_x, int end\_y)

{

if (box[end\_y][end\_x].figure == 1)

return 0;

if (attack(start\_x, start\_y, end\_x, end\_y) == 0) {

box[end\_y][end\_x].figure = 1;

box[start\_y][start\_x].figure = 0;

reattack();

}

return 0;

}

//Задание начального расположения фигур

int desk\_setter()

{

static short position[8][8] = {{0, 0, 0, 0, 0, 0, 0, 0},

{0, 0, 0, 0, 0, 0, 1, 0},

{0, 0, 0, 0, 0, 0, 0, 0},

{0, 0, 0, 0, 1, 0, 0, 0},

{0, 0, 0, 1, 0, 0, 0, 0},

{0, 0, 1, 0, 0, 0, 0, 0},

{0, 0, 0, 0, 0, 0, 0, 0},

{1, 0, 0, 0, 0, 0, 0, 0}};

for (int i = 0; i < ROWS; i++)

for (int j = 0; j < COLS; j++)

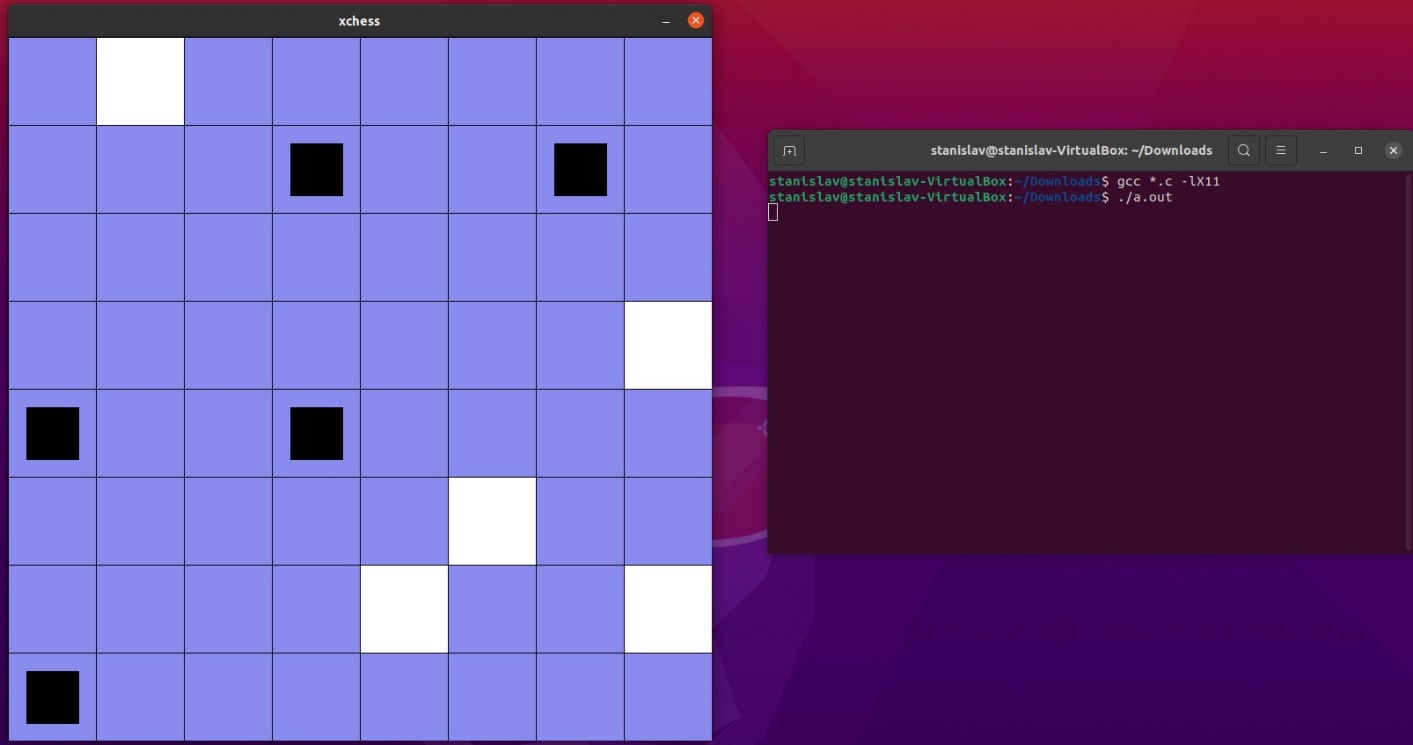
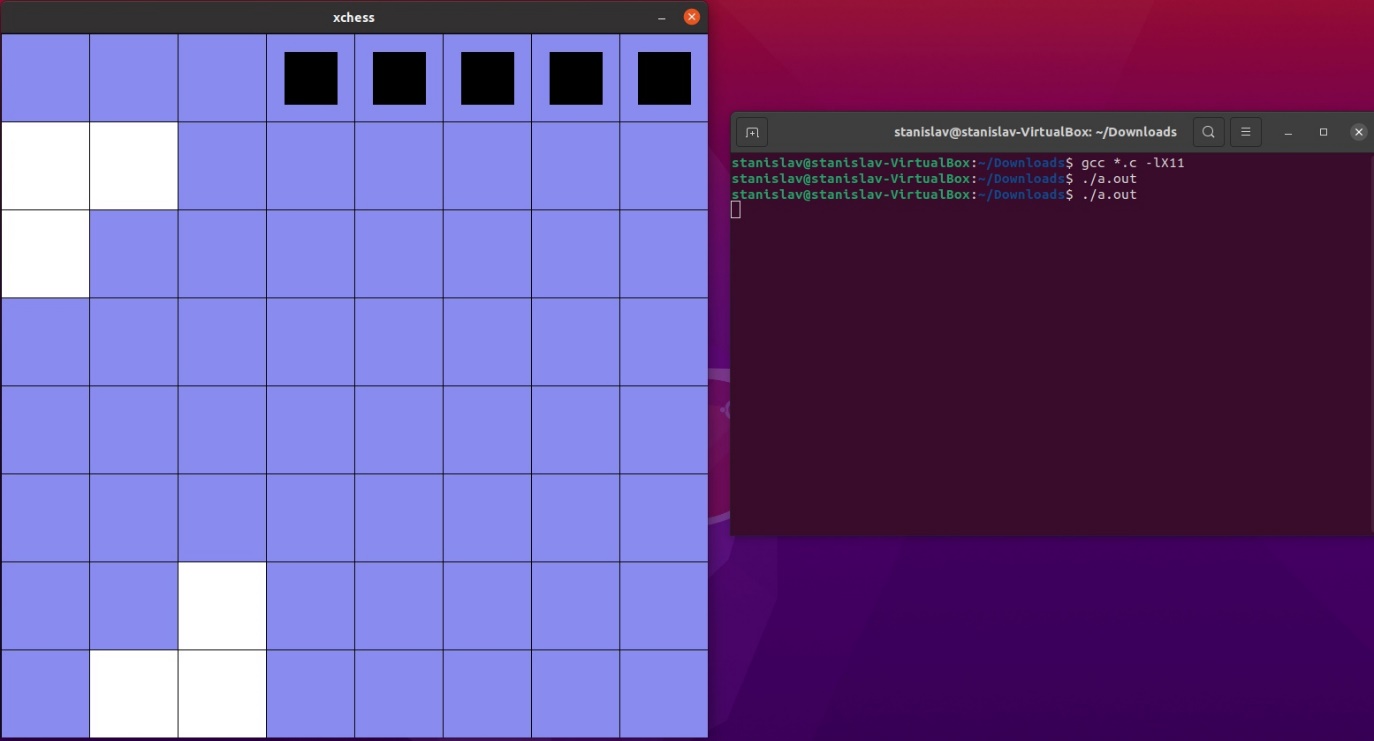
box[i][j].figure = position[i][j];

reattack();

return 0;

}

**Результат работы программы:**

** **

**Литература:**

1) bigor.bmstu.ru – база и генератор образовательных ресурсов

2) Лекции по курсу «Программирование графических приложений»

3) Электронный учебник: http://eufs.bmstu.ru/ee8d5d88-abfa-11e6-aa39-005056960017/09-02-2021-%D0%AD%D0%BB%D0%B5%D0%BA%D1%82%D1%80%D0%BE%D0%BD%D0%BD%D1%8B%D0%B8\_\_%D1%83%D1%87%D0 %D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5\_%D0%B2\_X\_Window\_System.htm