|  |  |
| --- | --- |
|  | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ Робототехника и комплексная автоматизация (РК)

КАФЕДРА Системы автоматизированного проектирования (РК-6)

**ОТЧЕТ О ВЫПОЛНЕНИИ ЛАБОРАТОРНОЙ РАБОТЫ**

**по курсу программирование графических приложений**

Студент Долженко Анастасия Тимофееевна

Группа РК6-42Б

Тип задания Лабораторная работа №3

Вариант R23

Студент **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Долженко А.Т.**

*подпись, дата фамилия, и.о.*

Преподаватель **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Князева С.В.**

*подпись, дата фамилия, и.о.*

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Москва, 2024 г.*

**Оглавление**

[Введение 3](#_Toc162871062)

[Постановка задачи 3](#_Toc162871063)

[Основания для разработки 3](#_Toc162871064)

[Требования к программе 3](#_Toc162871065)

[Стадии и этапы разработки 3](#_Toc162871066)

[Порядок контроля и приемки 4](#_Toc162871067)

[Описание алгоритма 4](#_Toc162871068)

[Приложение 1. Код программы 4](#_Toc162871069)

[Приложение 2. Результаты тестирования 9](#_Toc162871070)

# Введение

Настоящий документ определяет техническое задание на разработку графического приложения для отрисовки прямоугольников с выделением их точек пересечния. (далее по тексту – LAB3) в формате системы программирования C.

# Постановка задачи

Разработать программу построения пересекающихся прямоугольников в графическом окне, которая реализует технику «резиновой рамки». Опорный угол «резиновой рамки» фиксируется по координатам курсора в любой точке графического окна программы при нажатии левой кнопки мыши. После этого при любых перемещениях мыши с нажатой левой кнопкой начальную и текущую позиции курсора должен соединять прямоугольный контур, который отображается пунктиром и обозначает «резиновую рамку». Её размеры и/или ориентация должны синхронно изменяться при движении курсора, пока нажата левая кнопка мыши. при этом должна быть автоматически блокирована возможность выхода курсора за границы графического окна программы. После отпускания нажатой левой кнопки мыши «резиновая рамка» фиксируется изображением прямоугольника со сплошным контуром. При этом каждая точка пересечения его с другими прямоугольниками должна быть выделена круглым пятном, размер которого превосходит толщину линий их контуров. Кроме того, необходимо предусмотреть возможность удалить любой прямоугольник щелчком правой кнопки мыши, когда курсор находится в зоне изображения его контура. Завершение программы должно обеспечивать нажатие клавиш CTRL+R на клавиатуре. При разработке программы необходимо предусмотреть соответствующую обработку событий и изображений в её графическом окне, используя библиотечные функции программного интерфейса Xlib из состава X Window System.

# Основания для разработки

Программа LAB3 разрабатывается в рамках лабораторной работы по курсу "Программирование графических приложений" для практического изучения создания резиновых объектов.

# Требования к программе

**1. Требования к функциональным характеристикам**

1.1. При этом положение каждой вершины должно фиксироваться ее координатами в условных единицах, пропорциональных размеру графического окна программы.

1.2. При нажатии правой кнопки мыши на область, ограниченную границей прямоугольника, он должен быть удален.

1.4 Должен быть реализован выход из программы по нажатию клавиш CTRL+R

**2. Требования к надежности**

Должно быть ограничение минимального размера прямоугольника. При построении курсор должен быть захвачен.

**3. Условия эксплуатации**

3.1. Программа LAB3 должна быть ориентирована на эксплуатацию в среде OS UNIX

3.2. Программа LAB3 должна быть реализована в виде выполняемого файла с именем LAB3, по которому она должна вызываться средствами любого командного процессора OS UNIX.

3.3. Программа LAB3 должна эксплуатироваться в режиме графического окна, которое обрабатывает события мыши и клавиатуры.

**4. Требования к информационной и программной совместимости**

4.1. При разработке программы LAB3 необходимо использовать библиотеку lX11 для языка Си для работы с оконной системой X Window System, обеспечивающей стандартные инструменты и протоколы построения графического интерфейса пользователя.

# Стадии и этапы разработки

В процессе разработки программы LAB4 средствами языка Си и библиотеки lX11 необходимо выполнить следующие этапы:

* подготовить файл с исходным кодом программы LAB4, используя любой текстовый редактор OS UNIX, например, xedit;
* построить по исходному коду выполняемый файл LAB4 программы LAB4, вызвав C-компилятор OS UNIX и подключив стандартную библиотеку оконной системы X Window System libX11.a следующим образом:  **$ gcc \*.c -lX11 -lm**

# Порядок контроля и приемки

1. Для контроля функционирования программы LAB3 необходимо запустить её и убедиться, что при нажатии кнопки мыши появляется резиновый объект прямоугольник и при отпускании левой кнопки мыши создается закрашенный прямоугольник. При нажатии правой кнопки мыши удаляется прямоугольник в зоне его изображения.
2. Для приемки программы LAB3 должен быть организован вызов выполняемого файла LAB3 в консольном режиме работы OS UNIX или режиме эмуляции терминала операционной среды X Window System.

# Описание алгоритма

Реализация алгоритма программы включает в себя следующие шаги:

1. Определение начальной точки выделения. Это делается путем отслеживания события нажатия левой кнопки мыши и запоминания текущих координат курсора.
2. Отображение границ прямоугольника. Когда пользователь перемещает курсор, программа рисует геометрическую фигуру между начальной точкой и текущим положением курсора. Эта фигура обновляется в реальном времени, по мере перемещения курсора.
3. Определение конечной точки выделения. Когда пользователь отпускает левую кнопку мыши, приложение определяет конечную точку выделения, которая соответствует текущим координатам курсора.
4. Обработка выделенной области. После того, как конечная точка выделения определена, программа обрисовывает четкий контур фигуры, делает ее заливку и выделяет точки пересечения с другими прямоугольниками.

# Приложение 1. Код программы

#include <X11/Xresource.h>

#include <X11/Xlib.h>

#include <X11/Xos.h>

#include <X11/Xutil.h>

#include <stdlib.h>

#include <cstdio>

int intersectTheRectangles(XRectangle rect1, XRectangle rect2){

if (rect1.x < rect2.x + rect2.width &&

rect1.x + rect1.width > rect2.x &&

rect1.y < rect2.y + rect2.height &&

rect1.y + rect1.height > rect2.y)

return 1; // intersect

else

return 0; // do not intersect

}

int intersectPoints(XRectangle rect1, XRectangle rect2, int\* x1\_, int\* y1\_, int\* x2\_, int\* y2\_, int\* x3\_, int\* y3\_, int\* x4\_, int\* y4\_){

int x1, y1, x2, y2, x3, y3, x4, y4, ab, bc, cd, da;

/\*

A\_\_\_\_\_\_\_\_\_B

| |

|\_\_\_\_\_\_\_\_\_|

D C

\*/

x1 = rect1.x;

x3 = rect2.x;

da = (x1 > x3)? x1 : x3;

x2 = rect1.x + rect1.width;

x4 = rect2.x + rect2. width;

bc = (x2 < x4) ? x2 : x4;

y1 = rect1.y;

y3 = rect2.y;

ab = (y1 > y3)? y1 : y3;

y2 = rect1.y + rect1.height;

y4 = rect2.y + rect2.height;

cd = (y2 < y4) ? y2 : y4;

int a[4] = {0,0,0,0};

int c\_points[4][2];

c\_points[0][0] = da;

c\_points[0][1] = ab;

c\_points[1][0] = bc;

c\_points[1][1] = ab;

c\_points[2][0] = bc;

c\_points[2][1] = cd;

c\_points[3][0] = da;

c\_points[3][1] = cd;

int f = 0;

for(int i = 0; i < 4; i++){

if( ! ((c\_points[i][0] == x1 && c\_points[i][1] == y1) ||

(c\_points[i][0] == x2 && c\_points[i][1] == y1) ||

(c\_points[i][0] == x1 && c\_points[i][1] == y2) ||

(c\_points[i][0] == x2 && c\_points[i][1] == y2) ||

(c\_points[i][0] == x3 && c\_points[i][1] == y3) ||

(c\_points[i][0] == x4 && c\_points[i][1] == y3) ||

(c\_points[i][0] == x3 && c\_points[i][1] == y4) ||

(c\_points[i][0] == x4 && c\_points[i][1] == y4)) )

{

if(f == 0){

\*x1\_ = c\_points[i][0];

\*y1\_ = c\_points[i][1];

f++;

}

else if(f == 1){

\*x2\_ = c\_points[i][0];

\*y2\_ = c\_points[i][1];

f++;

}

else if(f == 2){

\*x3\_ = c\_points[i][0];

\*y3\_ = c\_points[i][1];

f++;

}

else if(f == 3){

\*x4\_ = c\_points[i][0];

\*y4\_ = c\_points[i][1];

f++;

}

}

}

return 0;

}

void redraw(Display \*dpy, Window win, GC gc, XRectangle rectangleList[], XColor cl[], int amountRectangles){

for (int i = 0; i < amountRectangles; i++){

XDrawRectangle(dpy, win, gc, rectangleList[i].x, rectangleList[i].y, rectangleList[i].width, rectangleList[i].height);

XSetForeground(dpy, gc, cl[1].pixel);

XFillRectangle(dpy, win, gc, rectangleList[i].x, rectangleList[i].y, rectangleList[i].width, rectangleList[i].height);

XSetForeground(dpy, gc, cl[0].pixel);

}

//перечесечения

if(amountRectangles > 1){

for (int i = 0; i < amountRectangles; i++){

for (int j = 0; j < amountRectangles; j++){

if(i == j) continue;

// ifrectangles intersect

if (intersectTheRectangles(rectangleList[i], rectangleList[j])){

// Calculate intersection points

int x1 = -1, y1 = -1, x2 = -1, y2 = -1, x3 = -1, y3 = -1, x4 = -1, y4 = -1;

int i\_points[4];

intersectPoints(rectangleList[i], rectangleList[j], &x1, &y1, &x2, &y2, &x3, &y3, &x4, &y4);

// draw points

XSetForeground(dpy, gc, cl[2].pixel);

if(x1 != -1) XFillArc(dpy, win, gc, x1 - 3, y1 - 3, 6, 6, 0, 360 \* 64);

if(x2 != -1) XFillArc(dpy, win, gc, x2 - 3, y2 - 3, 6, 6, 0, 360 \* 64);

if(x3 != -1) XFillArc(dpy, win, gc, x3 - 3, y3 - 3, 6, 6, 0, 360 \* 64);

if(x4 != -1) XFillArc(dpy, win, gc, x4 - 3, y4 - 3, 6, 6, 0, 360 \* 64);

}

}

}

}

}

int rekey(XEvent\* ev) {

KeySym ks = XLookupKeysym((XKeyEvent\*) ev, 1);

if((ks == XK\_R) && (ev->xkey.state == ControlMask)) return('R'); //CTRL-R to return exit-code

return 0;

}

int dispatch(Display\* dpy, Window win, GC gc, XColor cl[4], unsigned long mask) {

int flag = 0; // exit flag

int tracking = 0; // check if lbm is pressed

int x1, y1; // coordinates of the basic point

int x2, y2; // coordinates of the point

int xm, ym; // cursor coordinates

int amountRectangles = 0; // amount of rectangles in window

XRectangle rect; // rectangle object

XRectangle\* rectangleList; // array of rectangles

XEvent event;

while (flag == 0){

XNextEvent(dpy, &event);

switch (event.type){

case Expose:

if ((event.xexpose.window == win) && (amountRectangles > 0))

XSetFunction(dpy, gc, GXcopy);

XClearWindow(dpy, win);

redraw(dpy, win, gc, rectangleList, cl, amountRectangles);

case ButtonPress: // press left mouse button

if (event.xbutton.button == Button1){

tracking = 1;

x1 = event.xbutton.x;

y1 = event.xbutton.y;

XGrabPointer(dpy, win, False, mask, GrabModeAsync, GrabModeSync, win, None, CurrentTime); //захватили курсор

}

break;

case ButtonRelease: // release lmb

x2 = event.xbutton.x;

y2 = event.xbutton.y;

XUngrabPointer(dpy, CurrentTime);

if (x1 != x2 && y1 != y2)

switch (event.xbutton.button){

case Button1: // press left mouse buttton

// refine new rectangle object

x2 = event.xbutton.x;

y2 = event.xbutton.y;

// fill the coordinates of top left angle of rectangle

rect.x = x1;

rect.y = y1;

// fill the height and width of rectangle

rect.width = x2 - x1;

rect.height = y2 - y1;

if (amountRectangles == 0) rectangleList = (XRectangle \*)calloc((1), sizeof(XRectangle)); // if it is the first rectangle

else rectangleList = (XRectangle \*)realloc(rectangleList, (amountRectangles + 1) \* sizeof(XRectangle)); // if it is not the first rectangle

// add the rectangle in tthe list

rectangleList[amountRectangles].x = rect.x;

rectangleList[amountRectangles].y = rect.y;

rectangleList[amountRectangles].width = rect.width;

rectangleList[amountRectangles].height = rect.height;

// increment the amount of rectangles

amountRectangles++;

// redraw the window with the new rectangle and inersection points

redraw(dpy, win, gc, rectangleList, cl, amountRectangles);

tracking = 0;

break;

case Button3: // if press the right mouse button - delete the rectangle

for (int i = amountRectangles; i >=0; i--){ // find the selected rectangle

// coordinates of the cursor

int x = event.xbutton.x;

int y = event.xbutton.y;

if (rectangleList[i].x <= x <= rectangleList[i].x + rectangleList[i].width &&

rectangleList[i].y <= y <= rectangleList[i].y + rectangleList[i].height){ // if it is the selected rectangle

for (int j = i + 1; j < amountRectangles; j++) rectangleList[j - 1] = rectangleList[j]; // shift the list after selected reectangle to the left

amountRectangles--; // decrement the amount of recangles

break;

}

}

XClearWindow(dpy, win);

// redraw the window with new conditions

redraw(dpy, win, gc, rectangleList, cl, amountRectangles);

}

break;

case MotionNotify:

if (tracking){ // if the left mouse button is pressed

XSetFunction(dpy, gc, GXcopy);

XClearWindow(dpy, win);

redraw(dpy, win, gc, rectangleList, cl, amountRectangles);

xm = event.xmotion.x;

ym = event.xmotion.y;

// define the attributes of

rect.x = x1;

rect.y = y1;

rect.width = xm - x1;

rect.height = ym - y1;

// set the atttributes for the line to draw the rectangle without intersections

XSetLineAttributes(dpy, gc, 1, LineOnOffDash, CapNotLast, JoinMiter);

XSetForeground(dpy, gc, cl[3].pixel);

// draw the rectangle

XDrawRectangle(dpy, win, gc, rect.x, rect.y, rect.width, rect.height);

XSetForeground(dpy, gc, cl[0].pixel); // обводка первого прямоугольника

XSetLineAttributes(dpy, gc, 1, LineSolid, CapNotLast, JoinMiter);

}

break;

case KeyPress: // close the program with the CTRL+R

flag = rekey(&event);

break;

default: break;

}

}

return(flag);

}

int main(int argc, char \*argv[]){

Display \*dpy; // graphic display

int src; // screen number

int depth; // screen depth

Colormap cmap; // color palate

GC gc; // graphic context

XSetWindowAttributes attr; // window attributes

unsigned long mask; // event mask

XColor exact; // returns the nearest RGB codes on PK

XColor cl[4]; // returrns exact color codes

Window root, win; // main & root windows id

// initialization

dpy = XOpenDisplay(NULL);

src = DefaultScreen(dpy);

depth = DefaultDepth(dpy, src);

cmap = DefaultColormap(dpy, src);

root = DefaultRootWindow(dpy);

// create grapth context

gc = DefaultGC(dpy, src);

XAllocNamedColor(dpy, cmap, "#000000", &cl[0], &exact); // line rectangles color

XAllocNamedColor(dpy, cmap, "#61E1FA", &cl[1], &exact); // rectangles color

XAllocNamedColor(dpy, cmap, "#FE0B16", &cl[2], &exact); // intersect point color

XAllocNamedColor(dpy, cmap, "#FFFFFF", &cl[3], &exact); // dashed line color

mask = CWOverrideRedirect | CWEventMask | CWBackPixel;

attr.override\_redirect = False;

attr.event\_mask = ButtonPressMask | ButtonReleaseMask | ExposureMask | ButtonMotionMask | KeyPressMask;

attr.background\_pixel = cl[3].pixel;

XAllocNamedColor(dpy, cmap, "#000000", &cl[3], &exact);

XAllocNamedColor(dpy, cmap, "#000000", &cl[3], &exact);

win = XCreateWindow(dpy, root, 0, 0, 640, 480, 1, depth, InputOutput, CopyFromParent, mask, &attr);

XMapWindow(dpy, win);

mask = ButtonReleaseMask | ButtonMotionMask;

dispatch(dpy, win, gc, cl, mask);

XDestroyWindow(dpy, win);

XCloseDisplay(dpy);

return 0;

}

# Приложение 2. Результаты тестирования

Изображение выглядит как снимок экрана, Прямоугольник, текст, диаграмма

Автоматически созданное описание

Изображение выглядит как снимок экрана, Прямоугольник, диаграмма, прямоугольный

Автоматически созданное описание

**Список литературы**

1. НИИСИ РАН. Графический стандарт X Window. Функции библиотеки Xlib, 2000.
2. Ниалл Мэнсфилд. The X Window System: A User's Guide (перевод: Система X Window: Руководство пользователя), 1999.
3. Программирование в среде X Window на основе библиотеки Xlib. Режим доступа: https://dims.petrsu.ru/posob/X/osnov.htm (дата обращения 30.03.2024).