|  |  |
| --- | --- |
|  | **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 Тимофеевна

Группа РК6-42Б

Тип задания Лабораторная работа №3

Вариант 23S

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](#_Toc162871810)

[**Постановка задачи** 3](#_Toc162871811)

[**Основания для разработки** 3](#_Toc162871812)

[**Требования к программе** 3](#_Toc162871813)

[**Стадии и этапы разработки** 3](#_Toc162871814)

[**Порядок контроля и приемки** 4](#_Toc162871815)

[**Описание алгоритма** 4](#_Toc162871816)

[**Приложение 1. Код программы** 4](#_Toc162871817)

[**Приложение 2. Результаты тестирования** 8](#_Toc162871818)

[**Список литературы** 8](#_Toc162871819)

# **Введение**

Настоящий документ определяет техническое задание на разработку графического приложения для поиска прямоугольника с минимальным отношением меньшей и большей стороны (далее по тексту – LAB4) в формате системы программирования C.

# **Постановка задачи**

Разработать программу поиска минимального по отношению меньшей и большей сторон в заданном наборе изотетичных прямоугольников, которые произвольно расположены в ее графическом окне ограниченного размера без перекрытий. Требуемый набор формируется и изменяется путем построения новых или удаления существующих прямоугольников. Для построения прямоугольников должна использоваться техника резиновой рамки, которая изображается пунктиром и управляется перемещением курсора мыши в границах графического окна, пока нажата ее левая кнопка. Удаление любого прямоугольника должно производиться щелчком правой кнопки мыши, когда курсор указывает на его изображение. Эти изменения должны синхронно отражаться в динамическом массиве, где прямоугольники упорядочены по отношению сторон, с закраской в графическом окне прямоугольника, у которого оно минимально. Завершение программы должно обеспечивать нажатие комбинации клавиш CTRL-A на клавиатуре. При разработке этой программы необходимо предусмотреть соответствующую обработку событий и изображений в ее графическом окне, используя библиотечные функции программного интерфейса Xlib из X Window System, a также реализовать сортировку прямоугольников по отношению сторон.

# **Основания для разработки**

Программа LAB4 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.

1.3 Прямоугольник, имеющий минимальное отношение меньшей и большей сторон, должен быть выделен цветом.

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

**2. Требования к надежности**

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

**3. Условия эксплуатации**

3.1. Программа LAB4 должна быть ориентирована на эксплуатацию в среде OS UNIX

3.2. Программа LAB4 должна быть реализована в виде выполняемого файла с именем LAB4, по которому она должна вызываться средствами любого командного процессора OS UNIX.

3.3. Программа LAB4 должна эксплуатироваться в режиме графического окна, которое обрабатывает события мыши и клавиатуры.

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

4.1. При разработке программы LAB4 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 LAB4 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. Минимальный по искомому условию прямоугольник должен быть выделен цветом.
2. Для приемки программы LAB4 должен быть организован вызов выполняемого файла LAB4 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.

# **Приложение 1. Код программы**

/\* Sort Rectangle \*/

#include <X11/Xresource.h>

#include <X11/Xlib.h>

#include <X11/Xos.h>

#include <X11/Xutil.h>

#include <stdlib.h>

#include <stdio.h>

int checkOverlap(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; // Rectangles overlap

}

else return 0; // Rectangles do not overlap

}

int max(int a, int b) {return (a > b) ? a : b;}

int min(int a, int b) {return (a < b) ? a : b;}

int compareRectangles(const void \*a, const void \*b){

XRectangle rect1 = \*(XRectangle \*)a;

XRectangle rect2 = \*(XRectangle \*)b;

double metricA = min(rect1.width, rect1.height) / max(rect1.width, rect1.height);

double metricB = min(rect2.width, rect2.height) / max(rect2.width, rect2.height);

return ( metricA - metricB );

}

int rekey(XEvent\* ev) {

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

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

return 0;

}

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

XClearWindow(dpy, win);

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);

if (i == 0) XSetForeground(dpy, gc, cl[2].pixel);

XFillRectangle(dpy, win, gc, rectangleList[i].x + 1, rectangleList[i].y + 1, rectangleList[i].width - 1, rectangleList[i].height - 1);

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

}

}

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

int flag = 0; // exit flag

int flagRMB; // RMB 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: // redraw while initialize

if ((event.xexpose.window == win) && (amountRectangles > 0)) XSetFunction(dpy, gc, GXcopy);

XClearWindow(dpy, win);

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

break;

case ButtonPress: // press lmb

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);

}

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: // lmb

// refine new rectangle object

x2 = event.xbutton.x;

y2 = event.xbutton.y;

if (x2 < x1){

rect.x = x2;

rect.width = x1 - x2;

}

else{

rect.x = x1;

rect.width = x2 - x1;

}

if (y2 < y1){

rect.y = y2;

rect.height = y1 - y2;

}

else{

rect.y = y1;

rect.height = y2 - y1;

}

if (amountRectangles == 0) rectangleList = calloc((1), sizeof(XRectangle));

else rectangleList = realloc(rectangleList, (amountRectangles + 1) \* sizeof(XRectangle));

rectangleList[amountRectangles++] = rect;

int der = 0;

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

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

if (i != j && checkOverlap(rectangleList[i], rectangleList[j])) der++;

}

}

if (der) amountRectangles--;

qsort(rectangleList, amountRectangles, sizeof(XRectangle), compareRectangles);

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

tracking = 0;

break;

case Button3: // rmb

flagRMB = 0;

int temp = amountRectangles;

int delRectIndex;

// check if the cursor in rectangle

while((!flagRMB) && (temp-- > 0)){

if ((event.xbutton.x > rectangleList[temp].x) && (event.xbutton.x < (rectangleList[temp].x + rectangleList[temp].width))

&& (event.xbutton.y > rectangleList[temp].y) && (event.xbutton.y < (rectangleList[temp].y + rectangleList[temp].height))){

flagRMB = 1;

delRectIndex = temp; // remember the rectangle

}

}

if(flagRMB){ // delere rectangle

for (int i = delRectIndex; i < amountRectangles; i++) rectangleList[i] = rectangleList[i + 1]; // resort

amountRectangles--;

flagRMB = 0;

}

XSetFunction(dpy, gc, GXcopy);

XClearWindow(dpy, win);

qsort(rectangleList, amountRectangles, sizeof(XRectangle), compareRectangles);

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

break;

} // close butttom3

}

break; // close butttom release

case MotionNotify:

if (tracking){

XSetFunction(dpy, gc, GXcopy);

XClearWindow(dpy, win);

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

xm = event.xmotion.x;

ym = event.xmotion.y;

if (xm < x1){

rect.x = xm;

rect.width = x1 - xm;

}

else{

rect.x = x1;

rect.width = xm - x1;

}

if (ym < y1){

rect.y = ym;

rect.height = y1 - ym;

}

else{

rect.y = y1;

rect.height = ym - y1;

}

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: // exit by CTRL+A

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 graph 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);

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, текст, Прямоугольник, программное обеспечение

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).