Министерство образования и науки Российской Федерации

Московский государственный технический университет им. Н. Э. Баумана

Кафедра «РК6» Группа 46Б

**Отчет по лабораторной работе № 4**

По курсу «Программирование графических приложений»

**Вариант S10**

Выполнил: Петраков С. А.

Проверил: Родионов С. В.

Дата: 27.04.2021

Подпись: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Москва, 2021 г.

**Задание**

Разработать программу построения перекрывающихся прямоугольных ячеек в графическом окне, которая реализует технику «резиновой рамки». Опорный угол каждой «резиновой рамки» фиксируется по координатам курсора в любой свободной точке графического окна (вне ячеек) 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 штрих 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 новой прямоугольной ячейки со сплошной рамкой. При этом её части, которые перекрывают ранее построенные ячейки, должны быть видимы. Кроме того следует также предусмотреть возможность удалить любую ячейку щелчком правой кнопки мыши, когда курсор находится внутри видимой части её контура. При этом покрытые ей части других ячеек должны стать видимыми. 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 XwindowSystem.

**Структура программы**

Исходный код программы графической сортировки и поиска составляют 3 модуля прикладных функций: основной диспетчер (xsort2.c), резиновый редактор (xsort1.c) и фигурный (в данном случае прямоугольный) модуль (xrect.c). Их информационную связь обеспечивает заголовочный файл “xsort.h”, который подключается директивой include в начале каждого модуля. В него входит декларация объединения (union) X-типов геометрических фигур (XFragment), макроопределение цветных индексов массива графических контекстов (GC) для их изображения.

3 поля объединения XFragment поддерживают хранение массивов отрезков прямых, прямоугольников и эллипсов. Их оконные координаты и георазмеры задаются соответствующими стандартными Х-типами примитивов графического вывода (XSegment, XRectangle и XArc). Графические контексты предусматривают цветную дифференциацию фиксированных (FGC), резиновых (RGC) геофигур, а также фона (BGC).

Кроме того, заголовочный файл “xsort.h” содержит спецификации прототипов всех прикладных функций с их разделением по программным модулям. Прикладные функции модулей xsort[012] инвариантны по типам геофигур. Все фигурно зависимые функции сосредоточены в фигурном модуле, в данном случае xrect.

**Основной управляющий модуль (xsort2.c)**

Глобальные графические параметры (static):

Display\* dpy Адрес дисплейной структуры

GC gc[] Массив цветных графических контекстов

Window Win Идентификатор корневого и основного окна

Прикладные функции

resmng – управление цветными ресурсами. Создает базу цветных ресурсов программы по спецификациям в ресурсном файле .XSort, аргументах командной строки или из значений по умолчанию в своем статическом массиве. В любом случае спецификация цветов задается в Х-формате. Цвета из базы ресурсов распределяются в цветовую карту палитру (или идентифицируются в ней) по умолчанию. Затем создается массив граф. контекстов для распределенных цветов палитры в поле foreground их структуры и с одинаковым фоном в поле background.

gcing – корректировка граф. контекстов. Установка тонкого пунктира с функцией Xor для резинового граф. контекста. Для остальных контекстов устанавливается двойная сплошная линия с функцией copy.

canvas – создание основного граф. окна с начальным размером 640 х 480. Его фон должен совпадать с фоном в массиве графических контекстов. Также задается маска событий, которые будут обрабатываться диспетчером dispatch.

expo – перерисовка изображений геофигур в граф. окне вызовом функции refrag из резинового модуля (xsort1). Минимизацию перерисовок обеспечивает техника отсечения. Функция вызывается по событию Expose диспетчером dispatch при любой потере изображения в графокне программы.

dispatch – диспетчер событий. Чтение очереди событий для графокна программы в основном для их резиновой обработки.

rekey – обработка событий нажатия комбинации Ctrl-R для выхода.

main – основная функция для вызова всех прикладных функций управляющего модуля, а также функции pass1 из резинового модуля (xsort1) для адресации ему граф. параметров данного модуля

**Резиновый модуль (xsort1.c)**

Глобальные графические параметры (static)

Адресуются из управляющего модуля (xsort2.c)

Глобальные геометрические параметры (static)

XFragment\* frag – адрес динамического массива геофигуры

int nfrag – длина массива геофигур

XFragment ftmp[1] – шаблон резиновой геофигуры

Прикладные функции

pass1 – передача граф. параметров из управляющего модуля через свои аргументы соответствующих адресных типов.

XFixes – перерисовка заданного числа адресованных геофигур с заданным графконтекстом. Реализует циклич. вызов фиг. функции XFix из фигурного модуля

rubin – идентификация базовой точки резинового редактирования по координатам мыши вызовом фигурной функции frag0, если была нажата левая кнопка мыши.

rubout – удаление геофигуры по указателю мыши при нажатии правой и средней кнопки. Изобр. стирается вызовом фиг. функций XFix. Для исключения геофигуры из массива frag выз. функцию realloc.

near – поиск геофигуры по координатам в её аргументах. Вызывается функцией rubout для удаления геофигуры. Проверку близости к каждый геофигуре из их массива обеспечивает вызов фигурной функции fragon из фигурного модуля.

rerub – реализует деформацию резинового шаблона геофигуры ftmp по событию MotionNotify при перемещении мыши с нажатой левой кнопкой. Резиновое редакт. реализует вызов фигурных функций fragvar и XContour из фигурного модуля, чтобы изменить размеры резинового шаблона и изображение его контура (стереть прошлый габарит и нарисовать новый габарит в рез. графконтекст).

refrag – тотальная перерисовка геофигур. Производится явно после завершения каждой операции редактирования массива геофигур или неявно при обращении из функции expo основного модуля, которая вызывается диспетчером dispatch для обработки события Expose при любой потере изображения в графокно программы. Для перерисовки геофигур используются рез. функция XFixes.

widewin – расширение графокна программы, когда зафиксированное изображение новой геофигуры пересекает его правую или/и нижнюю границу. Вычисление максимальных координат геофигуры обеспечивают фигурные функции fragmaxix и fragmaxiy. Если эти значения превышают габариты графокна программы, оно расширяется, чтобы геофигура стала полностью видна. Функция widewin необходима только для геофигур с центральной базой, например, круг или овал. Для геофигур с угловой базой, напр., прямоуг. или отрезок прямой она не нужна.

cancet – отменить фиксацию шаблона резиновой геофигуры при его наложении на изображение любой геофигуры в графокне программы, или когда размеры шаблона меньше минимально допустимых значений. Для анализа этих ситуаций вызываются фигурные функции fragover и tinyfrag.

fix – фиксирует контур резинового шаблона в графокне программы и добавляет его в массив геофигур. Эти действия сопровождаются при необходимости расширением габаритов окна программы функцией widewin и сортировкой массива геофигур стандартной функцией qsort. Для сравнения геофигур ей адресуется функция fragcmp, которая обязана быть специфицирована в одном модуле с точкой вызова qsort. Расширение массива геофигур обеспечивает стандартная функция realloc с одновременным инкрементом его длины nfrag.

fragcmp – сравнение пары геофигур для стандартной функции qsort, которую вызывает функция fix для сортировки массива геофигур (см. выше). Парное сравнение геофигур обеспечивает вызов фигурной функции difrag. Такое косвенное сравнение геофигур является вынужденным, т.к. функция сравнения, адресуемая qsort не может быть специфицирована с другом, в частности, в фигурном модуле (см. выше fix)

miniwin – устанавливает минимально допустимые габариты графокна программы для оконного менеджера. Их определяют максимальные координаты контуров имеющихся геофигур. Тогда при любых интерактивных изменениях размеров графокна программы все геофигуры будут полностью видны. Для этих координатных и габаритных расчетов используются фигурные функции fragmaxix и fragmaxiy, которые должны быть вызваны для каждой геофигуры. Функция miniwin должна вызываться после каждой модификации массива геофигур.

allfree – очистка массива геофигур и изображения в графокне программы с обнулением всех геопараметров. Эта функция должна вызываться диспетчером событий dispatch из основного модуля при завершении программы по нажатию комбинации клавиш Ctrl-R.

**Фигурный (прямоугольный) модуль (xrectc)**

Определение и преобразование типов геометрических фигур

В фигурном модуле сосредоточен исходный код прикладных функций, который зависит от типа геофигур. Требуемый тип обозначается XFig и должен быть определён в начале модуля директивой typedef. При этом допустимый набор типов ограничен стандартным X-типами XSegment (отрезок), XRectangle (прямоугольник) и XArc (овал или круг) по Х-типам полей объединения XFragment (см. xsort.h), которое адресует геофигуры для инвариантной обработки в обоих типонезависимых модулях. Явное адресное преобразование (XFragment \*) => (XFig \*) для аргументов фиг. функций, которое обеспечит корректный доступ к полям необходимого Х типа, реализует макрос REFIG. Он макроопределён в начале фиг. модуля и присутствует во всех фиг. функциях с арг. типа (XFragment \*)

Прикладные фигурные функции

fragon: обеспечивает идентификацию геофигуры по заданным координатам (x, y). Возвращает 0 значение (FALSE), если точка не принадлежит области (или контуру) геофигуры. Иначе возвращается положит. код (TRUE), значение которого показывает, например, предельно-допустимую погрешность идентификации.

fragover: обеспечивает контроль взаимного перекрытия изображений для пары адресованных геофигур. Возвращает 0-код (FALSE), когда перекрытия нет и 1-код (TRUE), если геофигуры имеют общие точки.

difrag: оценка различия для адресованной пары геофигур. Вызывается при сортировке массива геофигур в коде резиновой функции fragcmp, которая адресуется qsort (см. xsort1.c)

fragsize: возвращает габаритный размер адресованной геофигуры.

tinyfrag: сравнение габаритов адресованной геофигуры с минимально допустимыми размерами, которые установлены в исходном тексте этой фигурной функции. Если габарит меньше предельно-допустимой величины, возвращает значение 1 (TRUE), иначе возвращается 0.

frag0: фиксирует базовую точку адресуемой геофигуры в начале операции резинового редактирования. Для круга или овала база совпадает с его центром. Для прямоугольника база фиксирует одну из его вершин, а для отрезка указывает его начало. В любом случае базовые (x, y)-координаты должны передаваться в функцию frag0 вместе с адресом геофигуры.

fragvar: обеспечивает вариацию контурных габаритов адресованной геофигуры в ходе её резинового редактирования. Габаритные размеры вычисляются по отклонению (x, y)-координат (курсора мыши) в аргументах этой фигурной функции от базы резинового контура геофигуры, которая была зафиксирована предшествующим вызовом фигурной функции frag0. В коде fragvar координаты базы получаются из Х-структуры геофигуры при её нулевых габаритах. Их значения сохраняют внутренние статические переменные для последующих оценок габаритных отклонений от базы.

fragmaxix: возврат максимальной координаты X в габарите адресованной геофигуры. Она необходима при выходе за границы окна контуров геофигур с центральной базой (овал или круг). Для геофигур с угловой базой (прямоугольник или отрезок) эта фигурная функция не является необходимой.

fragmaxiy: возврат максимальной координаты y в габарите адресованной геофигуры (см. fragmaxix).

XContour: изображение контура адресованной геофигуры с заданным обычно резиновым графконтекстом (RGC).

XFix: изображение адресованной геофигуры с дополнительной прорисовкой её контура (исключая отрезки) в заданном графконтексте.

**Код**

**xsort.h**

typedef union {  
 XSegment seg[1];  
 XRectangle rec[1];  
 XArc arc[1];  
} XFragment;   
  
#define NGC 3  
#define FGC 0  
#define RGC 1  
#define BGC 2  
  
#define MINSIZEX 4  
#define MINSIZEY 4  
  
#define STARTSIZEX 640  
#define STARTSIZEY 480  
  
int pass1(Display\*, Window, GC\*);   
int rubin(XEvent\*);  
int rubout(XEvent\*);  
int rerub(XEvent\*);  
int fix(XEvent\*);  
int widewin();  
int miniwin();  
int refrag();  
int near(int, int);  
int cancet();  
int reggc(int);  
  
int fragon(XFragment\*, int, int, int);  
int difrag(XFragment\*, XFragment\*);  
int tinyfrag(XFragment\*);  
int fragsize(XFragment\*);  
int frag0(XFragment\*, int, int);  
int fragvar(XFragment\*, int, int);  
int fragmaxix(XFragment\*);  
int fragmaxiy(XFragment\*);  
int fragover(XFragment\*, XFragment\*);  
  
int XFixes(Display\*, Window, GC, XFragment\*, int);  
int XContour(Display\*, Window, GC, XFragment\*);  
int XFix(Display\*, Window, GC, XFragment\*, GC);  
  
int resmng(int, char\*[]);  
int canvas();  
int gcing();  
int dispatch();  
int rekey(XEvent\*);

**xsort1.c**

#include <X11/Xlib.h>  
#include <X11/Xutil.h>  
#include <string.h>  
#include <stdlib.h>  
#include "xsort.h"  
  
#define FRAGSIZ sizeof(XFragment)  
  
static Display\* dpy;  
static Window win;  
static GC\* gc;  
  
static XFragment ftmp[1];  
static XFragment bak[2];  
static XFragment\* frag;  
static int nfrag=0;  
static int GGC=FGC;  
  
//static int building = 0;  
  
int pass1(Display\* d, Window w, GC\* g) {  
 dpy = d; win = w; gc = g;  
 return(0);  
}  
  
int XFixes(Display\* dpy, Window win , GC \_gc, XFragment\* f, int n) {  
 int i;  
 for(i = 0; i < n; i++, f++) XFix(dpy, win, \_gc, f, gc[BGC]);  
 return(0);  
}  
  
int rubin(XEvent\* ev) {  
 int i;  
 XGrabPointer(dpy, win, False, (ButtonReleaseMask | Button1MotionMask),   
 GrabModeAsync, GrabModeAsync, win, None, CurrentTime);  
 if(ev->xbutton.button != Button1) return(~Button1);  
 //if((i = near(ev->xbutton.x, ev->xbutton.y)) == -1) {  
 //building = 1;  
 frag0(ftmp, ev->xbutton.x, ev->xbutton.y);  
 //}  
 return(Button1);  
}  
  
int near(int x, int y) {  
 int i;  
 for(i = nfrag-1; i >= 0; --i)  
 if(fragon(frag+i, x, y, 0) > 0) break;  
 return(i);  
}  
  
int rubout(XEvent\* ev) {  
 int i;  
 if(nfrag < 1) return(0);  
 if((i = near(ev->xbutton.x, ev->xbutton.y)) == -1) return(nfrag);  
 XFix(dpy, win, gc[BGC], frag+i, gc[BGC]);  
 XFlush(dpy);  
 if(--nfrag > i) memmove((frag + i), (frag + i + 1), (nfrag - i)\*FRAGSIZ);  
 frag = realloc(frag, nfrag\*FRAGSIZ);  
 if(nfrag == 0) frag = NULL;  
 return(nfrag);  
}  
  
int rerub(XEvent\* ev) {  
 static int x, y;  
 //if(!building) return(nfrag);  
 XContour(dpy, win, gc[RGC], ftmp);  
 if(fragvar(ftmp, ev->xmotion.x, ev->xmotion.y) < 0) {  
 XContour(dpy, win, gc[RGC], ftmp);  
 return(XWarpPointer(dpy, None, win, 0, 0, 0, 0, x, y));  
 }  
 XContour(dpy, win, gc[RGC], ftmp);  
 x = ev->xmotion.x; y = ev->xmotion.y;  
 return(0);  
}  
  
int reggc(int g) {  
 return(GGC = g);  
}  
  
int refrag() {  
 XFixes(dpy, win, gc[FGC], frag, nfrag);  
 return(nfrag);  
}  
  
int widewin() {  
 int w, h;  
 XWindowAttributes attr;  
 XGetWindowAttributes(dpy, win, &attr);  
 w = fragmaxix(ftmp);   
 h = fragmaxiy(ftmp);  
 if((w < attr.width) && (h < attr.height)) return(0);  
 if(w < attr.width) w = attr.width;  
 if(h < attr.height) h = attr.height;  
 XResizeWindow(dpy, win, w, h);   
 return(0);  
}  
  
int cancet() {  
 int i;  
 if(tinyfrag(ftmp) > 0) return(1);  
 return(0);  
}  
  
int fix(XEvent\* ev) {  
 int w, h;  
 XUngrabPointer(dpy, CurrentTime);  
 if(ev->xbutton.button != Button1) return(nfrag);  
 //if(!building) return(nfrag);  
 //building = 0;  
 fragvar(ftmp, ev->xbutton.x, ev->xbutton.y);  
 XContour(dpy, win, gc[RGC], ftmp);  
 XFlush(dpy);  
 if(cancet() > 0) return(0);  
 widewin(w, h);  
 frag = realloc(frag, (nfrag + 1)\*FRAGSIZ);  
 memcpy((frag+nfrag), ftmp, FRAGSIZ);  
 ++nfrag;  
 return(nfrag);  
}  
  
int miniwin() {  
 XSizeHints hint;  
 int i=0;  
 unsigned w=128;  
 unsigned h=128;  
 int xm, ym;  
 for(i=0; i < nfrag; i++) {  
 if((xm = fragmaxix(frag+i)) > w) w = xm;  
 if((ym = fragmaxiy(frag+i)) > h) h = ym;  
 }  
 hint.min\_width = w; hint.min\_height = h;  
 hint.flags = PMinSize;  
 XSetNormalHints(dpy, win, &hint);  
 return(0);  
}

**xsort2.c**

#include <X11/Xlib.h>  
#include <X11/Xresource.h>  
#include <X11/keysym.h>  
#include <X11/keysymdef.h>  
#include <string.h>  
#include <stdio.h>  
#include "xsort.h"  
  
static Display\* dpy;  
static GC gc[NGC];  
static Window win;  
  
int resmng(int argc, char\* argv[]) {  
 int scr;  
 Window root;  
 Colormap cmap;  
 XColor rgb, exact;  
 int i;  
 XrmDatabase rdb=NULL;  
 char restype[64];  
 char\* rtype = restype;  
 XrmValue resval;  
 XrmValue\* rval = &resval;  
 static char\* rname[] = {  
 "xsort.foreground",  
 "xsort.rubbground",  
 "xsort.background"  
 };  
 static char\* rdef[] = {  
 "black",  
 "grey",  
 "white"  
 };  
 static XrmOptionDescRec rtab[] = {  
 {"-fg ", ".foreground", XrmoptionSepArg, NULL},  
 {"-rg ", ".rubbground", XrmoptionSepArg, NULL},  
 {"-bg ", ".background", XrmoptionSepArg, NULL},  
 {"-xrm ", NULL, XrmoptionResArg, NULL}  
 };  
  
 dpy = XOpenDisplay(NULL);  
 scr = DefaultScreen(dpy);  
 cmap = DefaultColormap(dpy, scr);  
 root = DefaultRootWindow(dpy);   
  
 XrmInitialize();   
 rdb = XrmGetFileDatabase("xsort.res");  
 XrmParseCommand(&rdb, rtab, NGC+1, "xsort", &argc, argv);  
  
 puts("\nColor resources:");  
 for(i=0; i < NGC; i++) {  
 if(XrmGetResource(rdb, rname[i], NULL, &rtype, rval) == False)  
 rval->addr = rdef[i];  
 if(XParseColor(dpy, cmap, rval->addr, &rgb) == 0)  
 if(XLookupColor(dpy, cmap, (rval->addr = rdef[i]), &rgb, &exact) == 0)  
 memcpy(&rgb, &exact, sizeof(XColor));  
 printf("%s(%s): %s\n", rname[i], rtab[i].option, rval->addr); /\* Echo \*/  
 fflush(stdout);  
 XAllocColor(dpy, cmap, &rgb);  
 gc[i] = XCreateGC(dpy, root, 0, 0);  
 XSetForeground(dpy, gc[i], rgb.pixel);  
 }  
 if(rdb != NULL) XrmDestroyDatabase(rdb);  
  
 for(i=0; i < NGC; i++) XSetBackground(dpy, gc[i], rgb.pixel);  
 return(0);  
}  
  
int gcing() {  
 XGCValues gval;  
 unsigned long gmask = GCLineWidth;  
 int i;   
 gval.line\_width = 2;  
 for(i=0; i < NGC; i++) XChangeGC(dpy, gc[i], gmask, &gval);  
 gmask = (GCFunction | GCLineWidth | GCLineStyle);  
 gval.line\_width = 1;   
 gval.function = GXxor;  
 gval.line\_style = LineOnOffDash;  
 XChangeGC(dpy, gc[RGC], gmask, &gval);  
 return(0);  
}  
  
int canvas() {  
 XSetWindowAttributes attr;  
 unsigned long amask;  
 unsigned long emask;   
 Window root;  
 XGCValues gval;  
 XGetGCValues(dpy, gc[BGC], GCBackground, &gval);  
 attr.background\_pixel = gval.background;  
 attr.override\_redirect = False;  
 attr.bit\_gravity = NorthWestGravity;  
 amask = (CWOverrideRedirect | CWBackPixel | CWBitGravity);  
 root = DefaultRootWindow(dpy);  
 win = XCreateWindow(dpy, root, 0, 0, STARTSIZEX, STARTSIZEY, 1, CopyFromParent,  
 InputOutput, CopyFromParent, amask, &attr);  
 XStoreName(dpy, win, "xsort");  
 emask = (ButtonPressMask | ButtonReleaseMask | Button1MotionMask |  
 ExposureMask | KeyPressMask);  
 XSelectInput(dpy, win, emask);  
 XMapWindow(dpy, win);  
 return(0);  
}  
  
int rekey(XEvent\* ev) {  
 KeySym ks = XLookupKeysym((XKeyEvent\*) ev, 1);  
 if((ks == XK\_R) && (ev->xkey.state == ControlMask)) return('R');  
 return(0);  
}  
  
int dispatch() {  
 int done = 0;  
 XEvent event;  
 while(done == 0) {  
 XNextEvent(dpy, &event);  
 switch(event.type) {  
 case Expose: if(event.xexpose.count > 1) break;  
 refrag();  
 break;  
 case ButtonPress:  
 if(rubin(&event) != Button1) rubout(&event);  
 break;  
 case MotionNotify: rerub(&event);  
 break;  
 case ButtonRelease: fix(&event);  
 refrag();  
 miniwin();  
 break;  
 case KeyPress: done = rekey(&event);   
 break;  
 default: break;  
 }  
 }  
 XDestroyWindow(dpy, win);  
 XCloseDisplay(dpy);  
 return(0);  
}  
  
int main(int argc, char\* argv[]) {  
 resmng(argc, argv);  
 gcing();  
 canvas();  
 pass1(dpy, win, gc);  
 dispatch();  
 return(0);  
}

**xrect.c**

#include <X11/Xlib.h>  
#include "xsort.h"  
  
typedef XRectangle XFig;  
  
#define REFIG(F) (F->rec)  
  
int fragon(XFragment\* ff, int x, int y, int t) {  
 XFig\* f = REFIG(ff);  
 if((x < (f->x - 1)) || (x > (f->x + f->width + 1)) ||  
 (y < (f->y - 1)) || (y > (f->y + f->height + 1)))  
 return(0);   
 return(1);  
}  
  
int fragover(XFragment\* ff1, XFragment\* ff2) {  
 XFig\* f1 = REFIG(ff1);  
 XFig\* f2 = REFIG(ff2);  
 if((f1->x > f2->x + f2->width) || (f2->x > f1->x + f1->width) ||  
 (f1->y > f2->y + f2->height) || (f2->y > f1->y + f1->height)) return(0);  
 else return(1);  
}  
  
int difrag(XFragment\* ff1, XFragment\* ff2) {   
 XFig\* f1 = REFIG(ff1);  
 XFig\* f2 = REFIG(ff2);  
 return(f1->width \* f1->height - f2->width \* f2->height);  
}  
  
int tinyfrag(XFragment\* ff) {  
 XFig\* f = REFIG(ff);  
 if((f->width < MINSIZEX) || (f->height < MINSIZEY))  
 return(1);  
 return(0);  
}  
  
int fragsize(XFragment\* ff) {  
 XFig\* f = REFIG(ff);  
 return(f->width \* f->height);  
}  
  
int frag0(XFragment\* ff, int x, int y) {  
 XFig\* f = REFIG(ff);  
 f->x = x; f->y = y;  
 f->width = f->height = 0;  
 return(0);  
}  
  
int fragvar(XFragment\* ff, int x, int y) {  
 XFig\* f = REFIG(ff);  
 static int fx0, fy0;  
   
 if(f->width == 0)  
 fx0 = f->x;  
 if(f->height == 0)  
 fy0 = f->y;  
   
 if(x < fx0) {  
 f->x = x;  
 f->width = fx0 - x;  
 } else {  
 f->x = fx0;  
 f->width = x - fx0;  
 }  
 if(y < fy0) {  
 f->y = y;  
 f->height = fy0 - y;  
 } else {  
 f->y = fy0;  
 f->height = y - fy0;  
 }  
   
 return(0);  
}  
  
int fragmaxix(XFragment\* ff) {  
 XFig\* f = REFIG(ff);  
 return(f->x + f->width);  
}  
  
int fragmaxiy(XFragment\* ff) {  
 XFig\* f = REFIG(ff);  
 return(f->y + f->height);  
}  
  
int XContour(Display\* dpy, Window win, GC gc, XFragment\* ff) {  
 XFig\* f = REFIG(ff);  
 XDrawRectangles(dpy, win, gc, f, 1);  
 return(0);  
}  
  
int XFix(Display\* dpy, Window win, GC gc, XFragment\* ff, GC fillGC) {  
 XFig\* f = REFIG(ff);  
 XFillRectangles(dpy, win, fillGC, f, 1);  
 XDrawRectangles(dpy, win, gc, f, 1);  
 return(0);  
}

**xrect.res**

# Green All fragments  
xsort.foreground: #000000  
  
# Blue Rubber contour   
xsort.rubbground: gray  
  
# Black background for rubber out fragment  
xsort.background: rgb:FF/FF/FF

**xrect.m**

fig=xrect  
  
EXEC=${fig}  
  
  
LIBS= -lX11  
  
INCLUDES=xsort.h  
  
CC=cc  
  
OBJECTS=${fig}.c xsort1.c xsort2.c  
  
all: $(EXEC)  
$(OBJECTS): $(INCLUDES)  
$(EXEC): $(OBJECTS) $(INCLUDES)  
 $(CC) -o $(EXEC) $(OBJECTS) $(LIBS)

**Литература**

1. Конспекты лекций

2. [http://bigor.bmstu.ru](http://bigor.bmstu.ru/)

3. https://ru.bmstu.wiki/X\_Window\_System

4. https://docs.freebsd.org/ru/books/handbook/x11/