

**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 №4 по курсу**

**«Программирование графических приложений»**

**Студент: Сергеева Диана**

**Группа:** РК6-46Б

**Преподаватель:** Волосатова Т.М.

Проверил:

Дата:

2021 год

**13M вариант**

**Задание:**

Разработать программу пульсирующей окружности в графическом окне фиксированного размера. Габариты графического окна должны передаваться программе аргументом командной строки её вызова. В каждом цикле пульсации окружность сначала расширяется из центра графического окна до максимального размера, допустимого его габаритами,а затем снова сжимается в центральную точку.Изменение размера окружности должно происходить равномерно в визуально различимом темпе и непрерывно продолжаться, пока графическое окно программы остаётся полностью видимым. Когда оно хотя бы частично перекрывается окнами других клиентов текущей графической сессии, процесс пульсации окружности должен быть приостановлен. Как только графическое окно программы снова становится полностью видимым, сжатие или расширение окружности должно быть возобновлено. Следует также предусмотреть возможность приостановки и возобновления процесса пульсации при полностью видимом графическом окне программы нажатием клавиш PAUSE и ENTER на клавиатуре. Завершение программы должно обеспечивать нажатие комбинации клавиш CTRL-С 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.

**Структурный состав программы:**

Исходный код программы составляют 2 модуля прикладных функций: основной диспетчер (spiral2.c) и фигурный модуль (spiral1.c). Их информационную связь обеспечивает заголовочный файл “spiral.h”, который подключается директивой include в начале каждого модуля. В него входит декларация структуры геометрической фигуры, определение массива графических контекстов (GC) для их изображения. Хранит координаты и георазмеры.

**Основной управляющий модуль (spiral2.c)**

Display\* dpy - адрес дисплейной структуры

GC gc[] - массив цветных графических контекстов

Window Win - идентификатор корневого и основного окна

Прикладные части:

Display block – задаёт цвета и адрес дисплейной стуктуры

Window block – определяет атрибуты окна

Multi Block – определяет события нажатия клавиш

Exit block – удаляет окно и дисплей

**Фигурный модуль** **(spiral1.c)**

В фигурном модуле сосредоточен исходный код прикладных функций. Требуемый тип обозначается XFig и определён в начале модуля директивой typedef.

Прикладные фигурные функции:

**Исходный код:**

configure\_circle – задаёт начальную конфигурацию структуры

draw\_circle – рисует 1 шаг окружности

redraw – перерисовываает потерянное изображение в окне

reverse - изменяет направление изменения радиуса окружности

step\_radius – увеличивает или уменьшает радиус пульсирующей окружности

overlap - проверяет окно на частичное или полное заслонение, для торможения отображения окружности

key\_check - отвечает на события нажатия на клавиши клавиатуры

**spiral.h**

/\* структура пульсирующщей окружности \*/

typedef struct {

int R; //радиус окружности (текущий)

int dR; //величина увеличения радиуса

int Rm; //максимальный радиус возможный для данной конфигурации окна

int w; //ширина окна

int h; //высота окна

XPoint c; //точка центра окружности

} XArrow;

/\* объявление необходимых функций \*/

int configure\_circle(XArrow \*, int , int ); /\* задание начальной конфигурации структуры \*/

int reverse(XArrow\* ); /\*изменение направления изменения радиуса окружности\*/

int redraw(XEvent\*, GC, XArrow\*); /\*перерисовка потерянного изображения в окне\*/

int step\_radius(XArrow \*pr); /\*увеличение/уменьшение радиуса пульсирующей окружности\*/

int draw\_circle(Display \*, Window , GC , XArrow \*); /\*рисование одного шага окружности\*/

int key\_check(XEvent \*); /\* ответ на события нажатия на клавиши клавиатуры \*/

int overlap(XEvent\* ); /\* проверка окна на частичное или полное заслонение, для торможения отображения окружности\*/

**spiral1.c**

/\* 2 centred spiral implementation \*/

#include <X11/Xlib.h>

#include <X11/keysym.h>

#include <X11/keysymdef.h>

#include "spiral.h"

#define MAX(a,b) (((a)>(b))?(a):(b))

#define MIN(a,b) (((a)>(b))?(b):(a))

/\* задание начальной конфигурации структуры \*/

int configure\_circle(XArrow\* pr, int w, int h) {

pr->R = 1;

pr->c.x = w / 2;

pr->c.y = h / 2;

pr->dR = 1;

pr->h = h;

pr->w = w;

pr->Rm = MIN(h, w) / 2;

return 0;

}

/\*рисование одного шага окружности\*/

int draw\_circle(Display \*dpy, Window win, GC gc, XArrow \*pr){

XClearWindow(dpy, win);

XFillArc(dpy, win, gc, pr->c.x - pr->R, pr->c.y - pr->R, pr->R \* 2, pr->R \* 2, 0, 360 \* 64);

XFlush(dpy);

return step\_radius(pr);

}

/\*перерисовка потерянного изображения в окне\*/

int redraw(XEvent\* ev, GC gc, XArrow\* pr) {

static XRectangle clip[32]; /\* clip rectangles buffer \*/

static int n = 0; /\* clip rectangles number \*/

clip[n].x = ev->xexpose.x;

clip[n].y = ev->xexpose.y;

clip[n].width = ev->xexpose.width;

clip[n].height = ev->xexpose.height;

n++;

if((ev->xexpose.count > 0) && (n < 32))

return(0);

XSetClipRectangles(ev->xexpose.display, gc, 0, 0, clip, n, Unsorted);

draw\_circle(ev->xexpose.display, ev->xexpose.window, gc, pr);

XSetClipMask(ev->xexpose.display, gc, None);

return(n=0);

} /\* redraw \*/

/\*изменение направления изменения радиуса окружности\*/

int reverse(XArrow\* pr) {

pr->dR = -pr->dR;

return 0;

} /\* reverse \*/

/\*увеличение/уменьшение радиуса пульсирующей окружности\*/

int step\_radius(XArrow\* pr) {

pr->R += pr->dR;

if (pr->R == 0) {

reverse(pr);

}

else

if (pr->R == pr->Rm) {

reverse(pr);

}

return 0;

} /\* step\_radius \*/

/\* проверка окна на частичное или полное заслонение, для торможения отображения окружности\*/

int overlap(XEvent\* ev) {

if(ev->xvisibility.state != VisibilityUnobscured)

return(-32);

return(0);

} /\* overlap \*/

/\* ответ на события нажатия на клавиши клавиатуры \*/

int key\_check(XEvent\* ev) {

static int state = 1;

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

if(ks == XK\_Break)

return state = 2;

if(ks == XK\_Return)

return state = 1;

if((ks == XK\_C) && (ev->xkey.state == ControlMask))

return 0; /\*код выхода\*/

return state;

} /\* rapid \*/

**spiral2.c**

/\* 2 centred spiral main function \*/

#include <X11/Xlib.h>

#include <X11/Xutil.h>

#include <stdio.h>

#include "spiral.h"

/\* Main function \*/

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

Display\* dpy; /\* Graphic Display \*/

Window win; /\* programm main window \*/

GC gc; /\* Black & White Graphic Context \*/

int scr; /\* screen number = 0 \*/

XArrow r; /\* spiral structure \*/

{ /\* Display Block \*/

unsigned long tone; /\* Light drawing tone \*/

dpy = XOpenDisplay(NULL);

scr = DefaultScreen(dpy);

win = DefaultRootWindow(dpy);

scr = DefaultScreen(dpy);

gc = XCreateGC(dpy, win, 0, 0);

//работа с цветом

XColor pix;

Colormap cmap = DefaultColormap(dpy, scr); /\* экранная палитра \*/

XParseColor(dpy, cmap, "#FFFFFF", &pix); /\* −> RGB \*/

XAllocColor(dpy, cmap, &pix);

XSetForeground(dpy, gc, pix.pixel); //цвет переднего фона

XParseColor(dpy, cmap, "#000000", &pix); /\* −> RGB \*/

XAllocColor(dpy, cmap, &pix);

XSetBackground(dpy, gc, pix.pixel); //цвет заднего фона

//работа с цветом

} /\* Display block \*/

{ /\* Window block \*/

unsigned w, h; /\* main window width & height \*/

XSetWindowAttributes attr; /\* window attributes structure \*/

XGCValues gval; /\* GC structure \*/

unsigned long amask; /\* window attributes mask \*/

Window root=win; /\* Display root window \*/

XSizeHints hint; /\* Geometry WM hints \*/

if(argc < 2)

argv[1] = "640x480";

int empty; /\* for x & y \*/

XParseGeometry(argv[1], &empty, &empty, &w, &h); //получение исходной конфигурации окна из строки

if((h < DisplayHeight(dpy, scr)) && (w < DisplayWidth(dpy, scr)) && (h >= 480) && (w >= 480))

configure\_circle(&r, w, h); //конфигурация структуры

else

{

argv[1] = "640x480";

XParseGeometry(argv[1], &empty, &empty, &w, &h); //получение исходной конфигурации окна из строки

configure\_circle(&r, w, h);

}

printf("Now: %s %s (RxN)\n", argv[0], argv[1]); fflush(stdout);

amask = (CWOverrideRedirect | CWBackPixel);

XGetGCValues(dpy, gc, GCBackground, &gval);

attr.background\_pixel = gval.background; /\* = 0x0 \*/

attr.override\_redirect = False;

win = XCreateWindow(dpy, root, 0, 0, w, h, 1, CopyFromParent,

InputOutput, CopyFromParent, amask, &attr);

hint.flags = (PMinSize | PMaxSize);

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

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

XSetNormalHints(dpy, win, &hint);

XStoreName(dpy, win, "spiral");

XMapWindow(dpy, win);

} /\* window block \*/

{ /\* Multi Block \*/

unsigned long emask; /\* window event mask \*/

XEvent event; /\* graphic event structure \*/

int freeze = 1; /\* window visibility stop state \*/

unsigned delay=(1<<15); /\* multi delay period = 2^rate \*/

int multi=(1<<12); /\* multi code \*/

int count = 0; /\* delay count \*/

emask = ( KeyPressMask | ExposureMask | VisibilityChangeMask);

XSelectInput(dpy, win, emask);

while(multi != 0) { /\* Async dispatch event with multic ground \*/

event.type = 0;

XCheckWindowEvent(dpy, win, emask, &event);

switch (event.type) {

case Expose:

redraw(&event, gc, &r);

break;

case VisibilityNotify:

freeze = overlap(&event);

break;

case KeyPress:

multi = key\_check(&event);

break;

default:

break;

} /\* switck \*/

if((freeze < 0) || (multi == 2)) /\* окно заслонено или была нажата клавиша pouse \*/

continue;

if(count++ < delay) /\* увеличение и проверка счетчика для отрисовки \*/

continue;

count = 0; /\* сброс счетчика \*/

draw\_circle(dpy, win, gc, &r); /\*отрисовка нового положения окружности\*/

} /\* while event \*/

} /\* multi block \*/

{ /\* Exit block \*/

XDestroyWindow(dpy, win);

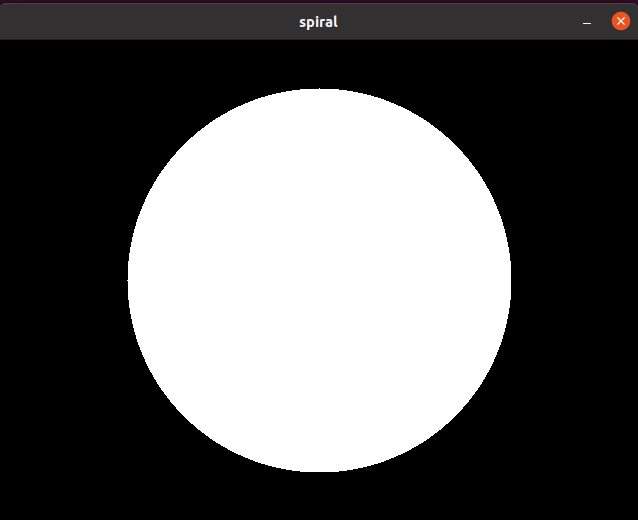
XCloseDisplay(dpy);

return(0);

} /\* exit block \*/

} /\* main \*/

**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 %B2%D0%B0%D0%BD%D0%B8%D0%B5\_%D0%B2\_X\_Window\_System.htm