

**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 *Робототехники и комплексной автоматизации*

КАФЕДРА *Системы автоматизированного проектирования (РК-6)*

# ОТЧЕТ О ВЫПОЛНЕНИИ РУБЕЖНОГО КОНТРОЛЯ

Студент Журавлев Н.В.

Группа РК6-42Б

Тип задания Рубежный контроль №2

Студент

**Журавлев Н.В.**

*подпись, дата фамилия, и.о.*

Преподаватель

**Волосатова Т.М.**

*подпись, дата фамилия, и.о.*

*Москва, 2021*

**Текст задания**

Разработать программу, которая отображает синхронное вращение двух стрелок в графическом окне. Каждая стрелка должна иметь форму отрезка прямой линии с треугольным наконечником, а их длины и угловые скорости вращения должны быть обратно пропорциональны. Вращение обеих стрелок должно происходить в одинаковом направлении в плоскости графического окна вокруг его центра, с которым совпадает общая начальная точка обеих стрелок, выделенная круглым пятном. Такая привязка должна сохраняться неизменной, а длины стрелок должны пропорционально увеличиваться или уменьшаться при любых реконфигурациях графического окна. Начальное угловое положение обеих стрелок должно передаваться программе во временном формате аргументом командной строки её вызова. Вращение стрелок должно начинаться сразу после появления на экране графического окна программы. Нажатие левой или правой кнопки мыши должно обеспечивать индивидуальную остановку большей или меньшей стрелки. Если левая и правая кнопка мыши одновременно удерживаются в нажатом состоянии, то обе стрелки должны быть неподвижны. В любом случае после отпускания нажатой левой и/или правой кнопки мыши вращение соответствующей стрелки должно возобновиться в прежнем направлении. Изменение направления вращения обеих стрелок одновременно, должен обеспечивать щелчок средней кнопки мыши. При этом программа должна реагировать на все перечисленные управляющие нажатия кнопок мыши, когда её курсор находится в любой точке графического окна. Завершение программы должно обеспечивать одновременное нажатие клавиш CTRL+END 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.

**Описание алгоритма**

В качестве основной структуры стрелки используется

typedef struct {

int A;

int dA;

XPoint c[2];

int LengthMax;

int Length;

} XArrow;

A – угловая координата в градусах;

dA – изменения угловой координаты;

c – массив точек, где с[0] положение начала отрезка, а c[1] конца;

Length – длина основной линии стрелки;

LengthMax – Максимальная длина.

Координаты конца отрезка рассчитываются исходя из параметрического задания эллипса.

Где, a – большая полуось эллипса, b – малая. Так как в нашем случает окружность, то берём a и b равные радиусу.

Центр “часов” изображается с использованием функции библиотеки Xlib XFillArc().

Отрисовка стрелки производится с помощью функции XDrawLine. Поворот на угол производится с помощью функции amod2pi, которая изменяет текущее значение угла на заданную величину.

В графическом модуле программы производится установка начальных параметров дисплея, окна и положения стрелок. Устанавливается маска ButtonReleaseMask и ButtonPressMask для возможности обработки события нажатия и отжатия кнопок. Когда одна из кнопок нажата вызывается функция для движения определённой стрелки, а для другой производится временная приостановка. При нажатии колёсика мыши изменяется направление дальнейшего движения обоих стрелок, путём изменения значения dA на отрицательное у обоих фигур.

**Описание входных данных**

Данные во временном формате, иначе задаётся стандартное значение.

**Описание выходных данных**

На выходе создается графическое окно с двумя вращающимися стрелками.

**Текст программы**

clock.h

#ifndef CLOCK\_H

#define CLOCK\_H

typedef struct {

int A;

int dA;

XPoint c[2];

int LengthMax;

int Length;

} XArrow;

int maxisize(XArrow\*, char\*);

int reset\_one(XArrow\*, int value, int);

int reset\_two(XArrow\*, int value, int);

int decent(XArrow\*);

int redraw(XEvent\*, GC, XArrow\*, XArrow\*);

int amod2pi(XArrow\*);

int twist\_all(Display\*, Window, GC, XArrow\*, XArrow\*);

int twist\_only\_one(Display\*, Window, GC, XArrow\*, XArrow\*);

int twist\_only\_two(Display\*, Window, GC, XArrow\*, XArrow\*);

int rep5355(Display\*, int);

int rapid(XEvent\*, int);

int overlap(XEvent\*);

#endif //CLOCK\_H

clock21.c

#include <X11/Xlib.h>

#include <X11/keysym.h>

#include <X11/keysymdef.h>

#include <string.h>

#include "clock.h"

#include <math.h>

#include <stdio.h>

#include <stdlib.h>

int maxisize(XArrow\* pr, char\* R0xN) {

int R0;

int N;

int empty;

XParseGeometry(R0xN, &empty, &empty, &R0, &N);

if(((pr->Length = R0) < 1) || (N < 1))

N = R0 = 0;

return(pr->LengthMax = 2\*R0\*N);

}

int decent(XArrow\* pr) {

int w = 2\*(pr->LengthMax + pr->Length) + (8 + 8); /\* R = R0 = dR now \*/

pr->c[0].x = w/2 - (pr->Length/2);

pr->c[1].x = pr->c[0].x + pr->Length; /\* = w/2 + (pr->R/2); \*/

pr->c[0].y = pr->c[1].y = w/2 + 8;

return(w);

}

int reset\_one(XArrow\* pr, int value, int length) {

pr->A = (30 \* value \*64);

pr->dA = (1\*64);

pr->Length = length;

return(2\*(pr->c[0].y));

}

int reset\_two(XArrow\* pr, int value, int length) {

pr->A = (30 \* value \* 64);

pr->dA = (6\*64);

pr->Length = length;

return(2\*(pr->c[0].y));

}

int redraw(XEvent\* ev, GC gc, XArrow\* pr, XArrow\* pr2) {

int y;

XArrow r;

XArrow r2;

static XRectangle clip[32];

static int n = 0;

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);

r = \*pr;

r2 = \*pr2;

XWindowAttributes attr;

XGetWindowAttributes(ev->xexpose.display, ev->xexpose.window, &attr);

pr->Length = attr.height / 5;

pr2->Length = attr.height / 5;

decent(pr);

decent(pr2);

reset\_one(pr, 0, attr.width / 4);

reset\_two(pr2, 0, attr.width / 3);

pr->c[0].x = attr.width / 2;

pr->c[0].y = attr.height / 2;

pr2->c[0].x = attr.width / 2;

pr2->c[0].y = attr.height / 2;

reset\_one(&r, 0, attr.width / 4);

y = reset\_two(&r2, 0, attr.width / 3) - 8;

while(twist\_all(ev->xexpose.display, ev->xexpose.window, gc, &r, &r2) < pr->Length);

r.dA = (pr->A - r.A);

r2.dA = (pr2->A - r2.A);

twist\_all(ev->xexpose.display, ev->xexpose.window, gc, &r, &r2);

XDrawString(ev->xexpose.display, ev->xexpose.window, gc,

8, y, "Ctrl+End", 8);

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

return(n=0);

}

int amod2pi(XArrow \* pr) {

pr -> A += (pr -> dA);

if (pr -> A == (360 \* 64))

return (pr -> A = (0 \* 64));

if (pr -> A == (0 \* 64))

pr -> A = (360 \* 64);

return (pr -> A);

}

int twist\_all(Display \* dpy, Window win, GC gc, XArrow \* arrow\_one, XArrow \* arrow\_two) {

XClearWindow(dpy,win);

float x1= cos(3.14 \* (arrow\_one->A + 90 \* 64) / (180 \* 64)) \* (arrow\_one->Length);

float y1= sin(3.14 \* (arrow\_one->A + 90 \* 64) / (180 \* 64)) \* (arrow\_one->Length);

float xs1= cos(3.14 \* (arrow\_one->A + 100 \* 64) / (180 \* 64)) \* (arrow\_one->Length - 20);

float ys1= sin(3.14 \* (arrow\_one->A + 100 \* 64) / (180 \* 64)) \* (arrow\_one->Length - 20);

float xs2= cos(3.14 \* (arrow\_one->A + 80 \* 64) / (180 \* 64)) \* (arrow\_one->Length - 20);

float ys2= sin(3.14 \* (arrow\_one->A + 80 \* 64) / (180 \* 64)) \* (arrow\_one->Length - 20);

XFillArc(dpy, win, gc, arrow\_one->c[0].x - arrow\_one->Length / 10, arrow\_one->c[0].y - arrow\_one->Length / 10, arrow\_one->Length / 5, arrow\_one->Length / 5, 0 \* 64, 360 \* 64);

XDrawLine(dpy, win, gc, arrow\_one->c[0].x , arrow\_one->c[0].y, arrow\_one->c[0].x - x1 , arrow\_one->c[0].y - y1);

XDrawLine(dpy, win, gc, arrow\_one->c[0].x - x1 , arrow\_one->c[0].y - y1, arrow\_one->c[0].x - xs1 , arrow\_one->c[0].y - ys1);

XDrawLine(dpy, win, gc, arrow\_one->c[0].x - x1 , arrow\_one->c[0].y - y1, arrow\_one->c[0].x - xs2 , arrow\_one->c[0].y - ys2);

XDrawLine(dpy, win, gc, arrow\_one->c[0].x - xs1 , arrow\_one->c[0].y - ys1, arrow\_one->c[0].x - xs2 , arrow\_one->c[0].y - ys2);

x1= cos(3.14 \* (arrow\_two->A + 90 \* 64) / (180 \* 64)) \* (arrow\_two->Length);

y1= sin(3.14 \* (arrow\_two->A + 90 \* 64) / (180 \* 64)) \* (arrow\_two->Length);

xs1= cos(3.14 \* (arrow\_two->A + 100 \* 64) / (180 \* 64)) \* (arrow\_two->Length - 20);

ys1= sin(3.14 \* (arrow\_two->A + 100 \* 64) / (180 \* 64)) \* (arrow\_two->Length - 20);

xs2= cos(3.14 \* (arrow\_two->A + 80 \* 64) / (180 \* 64)) \* (arrow\_two->Length - 20);

ys2= sin(3.14 \* (arrow\_two->A + 80 \* 64) / (180 \* 64)) \* (arrow\_two->Length - 20);

XDrawLine(dpy, win, gc, arrow\_two->c[0].x , arrow\_two->c[0].y, arrow\_two->c[0].x - x1 , arrow\_two->c[0].y - y1);

XDrawLine(dpy, win, gc, arrow\_two->c[0].x - x1 , arrow\_two->c[0].y - y1, arrow\_two->c[0].x - xs1 , arrow\_two->c[0].y - ys1);

XDrawLine(dpy, win, gc, arrow\_two->c[0].x - x1 , arrow\_two->c[0].y - y1, arrow\_two->c[0].x - xs2 , arrow\_two->c[0].y - ys2);

XDrawLine(dpy, win, gc, arrow\_two->c[0].x - xs1 , arrow\_two->c[0].y - ys1, arrow\_two->c[0].x - xs2 , arrow\_two->c[0].y - ys2);

XFlush(dpy);

amod2pi(arrow\_one);

amod2pi(arrow\_two);

return (arrow\_one -> Length);

}

int twist\_only\_one(Display \* dpy, Window win, GC gc, XArrow \* arrow\_one, XArrow \* arrow\_two) {

XClearWindow(dpy,win);

float x1= cos(3.14 \* (arrow\_one->A + 90 \* 64) / (180 \* 64)) \* (arrow\_one->Length);

float y1= sin(3.14 \* (arrow\_one->A + 90 \* 64) / (180 \* 64)) \* (arrow\_one->Length);

float xs1= cos(3.14 \* (arrow\_one->A + 100 \* 64) / (180 \* 64)) \* (arrow\_one->Length - 20);

float ys1= sin(3.14 \* (arrow\_one->A + 100 \* 64) / (180 \* 64)) \* (arrow\_one->Length - 20);

float xs2= cos(3.14 \* (arrow\_one->A + 80 \* 64) / (180 \* 64)) \* (arrow\_one->Length - 20);

float ys2= sin(3.14 \* (arrow\_one->A + 80 \* 64) / (180 \* 64)) \* (arrow\_one->Length - 20);

XFillArc(dpy, win, gc, arrow\_one->c[0].x - arrow\_one->Length / 10, arrow\_one->c[0].y - arrow\_one->Length / 10, arrow\_one->Length / 5, arrow\_one->Length / 5, 0 \* 64, 360 \* 64);

XDrawLine(dpy, win, gc, arrow\_one->c[0].x , arrow\_one->c[0].y, arrow\_one->c[0].x - x1 , arrow\_one->c[0].y - y1);

XDrawLine(dpy, win, gc, arrow\_one->c[0].x - x1 , arrow\_one->c[0].y - y1, arrow\_one->c[0].x - xs1 , arrow\_one->c[0].y - ys1);

XDrawLine(dpy, win, gc, arrow\_one->c[0].x - x1 , arrow\_one->c[0].y - y1, arrow\_one->c[0].x - xs2 , arrow\_one->c[0].y - ys2);

XDrawLine(dpy, win, gc, arrow\_one->c[0].x - xs1 , arrow\_one->c[0].y - ys1, arrow\_one->c[0].x - xs2 , arrow\_one->c[0].y - ys2);

x1= cos(3.14 \* (arrow\_two->A + 90 \* 64) / (180 \* 64)) \* (arrow\_two->Length);

y1= sin(3.14 \* (arrow\_two->A + 90 \* 64) / (180 \* 64)) \* (arrow\_two->Length);

xs1= cos(3.14 \* (arrow\_two->A + 100 \* 64) / (180 \* 64)) \* (arrow\_two->Length - 20);

ys1= sin(3.14 \* (arrow\_two->A + 100 \* 64) / (180 \* 64)) \* (arrow\_two->Length - 20);

xs2= cos(3.14 \* (arrow\_two->A + 80 \* 64) / (180 \* 64)) \* (arrow\_two->Length - 20);

ys2= sin(3.14 \* (arrow\_two->A + 80 \* 64) / (180 \* 64)) \* (arrow\_two->Length - 20);

XDrawLine(dpy, win, gc, arrow\_two->c[0].x , arrow\_two->c[0].y, arrow\_two->c[0].x - x1 , arrow\_two->c[0].y - y1);

XDrawLine(dpy, win, gc, arrow\_two->c[0].x - x1 , arrow\_two->c[0].y - y1, arrow\_two->c[0].x - xs1 , arrow\_two->c[0].y - ys1);

XDrawLine(dpy, win, gc, arrow\_two->c[0].x - x1 , arrow\_two->c[0].y - y1, arrow\_two->c[0].x - xs2 , arrow\_two->c[0].y - ys2);

XDrawLine(dpy, win, gc, arrow\_two->c[0].x - xs1 , arrow\_two->c[0].y - ys1, arrow\_two->c[0].x - xs2 , arrow\_two->c[0].y - ys2);

XFlush(dpy);

amod2pi(arrow\_one);

return (arrow\_one -> Length);

}

int twist\_only\_two(Display \* dpy, Window win, GC gc, XArrow \* arrow\_one, XArrow \* arrow\_two) {

XClearWindow(dpy,win);

float x1= cos(3.14 \* (arrow\_one->A + 90 \* 64) / (180 \* 64)) \* (arrow\_one->Length);

float y1= sin(3.14 \* (arrow\_one->A + 90 \* 64) / (180 \* 64)) \* (arrow\_one->Length);

float xs1= cos(3.14 \* (arrow\_one->A + 100 \* 64) / (180 \* 64)) \* (arrow\_one->Length - 20);

float ys1= sin(3.14 \* (arrow\_one->A + 100 \* 64) / (180 \* 64)) \* (arrow\_one->Length - 20);

float xs2= cos(3.14 \* (arrow\_one->A + 80 \* 64) / (180 \* 64)) \* (arrow\_one->Length - 20);

float ys2= sin(3.14 \* (arrow\_one->A + 80 \* 64) / (180 \* 64)) \* (arrow\_one->Length - 20);

XFillArc(dpy, win, gc, arrow\_one->c[0].x - arrow\_one->Length / 10, arrow\_one->c[0].y - arrow\_one->Length / 10, arrow\_one->Length / 5, arrow\_one->Length / 5, 0 \* 64, 360 \* 64);

XDrawLine(dpy, win, gc, arrow\_one->c[0].x , arrow\_one->c[0].y, arrow\_one->c[0].x - x1 , arrow\_one->c[0].y - y1);

XDrawLine(dpy, win, gc, arrow\_one->c[0].x - x1 , arrow\_one->c[0].y - y1, arrow\_one->c[0].x - xs1 , arrow\_one->c[0].y - ys1);

XDrawLine(dpy, win, gc, arrow\_one->c[0].x - x1 , arrow\_one->c[0].y - y1, arrow\_one->c[0].x - xs2 , arrow\_one->c[0].y - ys2);

XDrawLine(dpy, win, gc, arrow\_one->c[0].x - xs1 , arrow\_one->c[0].y - ys1, arrow\_one->c[0].x - xs2 , arrow\_one->c[0].y - ys2);

x1= cos(3.14 \* (arrow\_two->A + 90 \* 64) / (180 \* 64)) \* (arrow\_two->Length);

y1= sin(3.14 \* (arrow\_two->A + 90 \* 64) / (180 \* 64)) \* (arrow\_two->Length);

xs1= cos(3.14 \* (arrow\_two->A + 100 \* 64) / (180 \* 64)) \* (arrow\_two->Length - 20);

ys1= sin(3.14 \* (arrow\_two->A + 100 \* 64) / (180 \* 64)) \* (arrow\_two->Length - 20);

xs2= cos(3.14 \* (arrow\_two->A + 80 \* 64) / (180 \* 64)) \* (arrow\_two->Length - 20);

ys2= sin(3.14 \* (arrow\_two->A + 80 \* 64) / (180 \* 64)) \* (arrow\_two->Length - 20);

XDrawLine(dpy, win, gc, arrow\_two->c[0].x , arrow\_two->c[0].y, arrow\_two->c[0].x - x1 , arrow\_two->c[0].y - y1);

XDrawLine(dpy, win, gc, arrow\_two->c[0].x - x1 , arrow\_two->c[0].y - y1, arrow\_two->c[0].x - xs1 , arrow\_two->c[0].y - ys1);

XDrawLine(dpy, win, gc, arrow\_two->c[0].x - x1 , arrow\_two->c[0].y - y1, arrow\_two->c[0].x - xs2 , arrow\_two->c[0].y - ys2);

XDrawLine(dpy, win, gc, arrow\_two->c[0].x - xs1 , arrow\_two->c[0].y - ys1, arrow\_two->c[0].x - xs2 , arrow\_two->c[0].y - ys2);

XFlush(dpy);

amod2pi(arrow\_two);

return (arrow\_two -> Length);

}

int rep5355(Display \* dpy, int r) {

XKeyboardControl kbval;

kbval.key = XKeysymToKeycode(dpy, XK\_KP\_Add);

kbval.auto\_repeat\_mode = r;

XChangeKeyboardControl(dpy, (KBKey | KBAutoRepeatMode), & kbval);

kbval.key = XKeysymToKeycode(dpy, XK\_KP\_Subtract);

XChangeKeyboardControl(dpy, (KBKey | KBAutoRepeatMode), & kbval);

return (r);

}

int rapid(XEvent \* ev, int t) {

char sym[1];

KeySym code[1];

XLookupString((XKeyEvent \* ) ev, NULL, 0, code, NULL);

switch (code[0]) {

case XK\_plus:

case XK\_KP\_Add:

if (t > 1)

t >>= 1;

break;

case XK\_minus:

case XK\_KP\_Subtract:

if (t < (1 << 30))

t <<= 1;

break;

case 0:

t = (-32);

break;

case XK\_End:

if (ev -> xkey.state & ControlMask)

t = 0;

break;

default:

break;

}

return (t);

}

int overlap(XEvent \* ev) {

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

return (-32);

return (0);

}

Main.c

#include <X11/Xlib.h>

#include <X11/Xutil.h>

#include <stdio.h>

#include "clock.h"

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

Display \* dpy;

Window win;

GC gc[2];

XArrow arrow\_one;

XArrow arrow\_two;

{

/\* Display Block \*/

unsigned long tone;

XFontStruct \* fnptr;

dpy = XOpenDisplay(NULL);

DefaultScreen(dpy);

win = DefaultRootWindow(dpy);

DefaultScreen(dpy);

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

gc[1] = XCreateGC(dpy, win, 0, 0);

tone = 0x0FFF0F;

XSetForeground(dpy, gc[0], tone);

if ((fnptr = XLoadQueryFont(dpy, "9x15")) != NULL)

XSetFont(dpy, gc[0], fnptr -> fid);

} /\* Display block \*/

{

/\* Window block \*/

unsigned w, h;

XSetWindowAttributes attr;

XGCValues gval;

unsigned long amask;

Window root = win;

XSizeHints hint;

Atom wdw[1];

if (argc < 2)

argv[1] = "6:9";

int value\_1, value\_2;

if (sscanf( argv[1],"%d:%d", &value\_1, &value\_2) != 2) {

value\_1 = 6;

value\_2 = 9;

}

maxisize(&arrow\_one, "16x8");

decent(&arrow\_one);

maxisize(&arrow\_two, "16x8");

w = decent(&arrow\_two);

reset\_two(&arrow\_two, value\_2, 90);

h = reset\_one(&arrow\_one, value\_1, 120);

fflush(stdout);

amask = (CWOverrideRedirect | CWBackPixel);

XGetGCValues(dpy, gc[1], GCBackground, & gval);

attr.background\_pixel = gval.background;

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 = w;

hint.min\_height = h;

hint.max\_width = 785;

hint.max\_height = 785;

XSetNormalHints(dpy, win, & hint);

XStoreName(dpy, win, "clock");

wdw[0] = XInternAtom(dpy, "WM\_DELETE\_WINDOW", True);

XSetWMProtocols(dpy, win, wdw, 1);

XMapWindow(dpy, win);

} /\* window block \*/

{

/\* Multi Block \*/

unsigned long emask;

XEvent event;

int freeze\_one = 0;

unsigned delay\_one = (1 << 12);

int multi\_one = (1 << 12);

int count\_one = 0;

int freeze\_two = 0;

unsigned delay\_two = (1 << 12);

int multi\_two = (1 << 12);

int count\_two = 0;

int g = 0;

emask = (ExposureMask | KeyPressMask | FocusChangeMask |

VisibilityChangeMask | ButtonReleaseMask | ButtonPressMask);

XSelectInput(dpy, win, emask);

while (multi\_one != 0 || multi\_two != 0) {

event.type = 0;

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

switch (event.type) {

case Expose:

redraw(&event, gc[0], &arrow\_one, &arrow\_two);

break;

case VisibilityNotify:

freeze\_one = overlap(&event);

freeze\_two = overlap(&event);

break;

case ButtonRelease:

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

delay\_one = multi\_one = (1 << 12);

freeze\_one = 0;

} else {

if (event.xbutton.button == Button3) {

delay\_two = multi\_two = (1 << 12);

freeze\_two = 0;

}

}

break;

case ButtonPress:

case KeyPress:

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

if ((multi\_one = rapid(&event, delay\_one)) < 0) {

freeze\_one = -1;

}

} else {

if (event.xbutton.button == Button3) {

if ((multi\_two = rapid(&event, delay\_two)) < 0) {

freeze\_two = -1;

}

} else {

if (event.xbutton.button == Button2) {

arrow\_one.dA = -arrow\_one.dA;

arrow\_two.dA = -arrow\_two.dA;

} else {

if ((multi\_one = rapid(&event, delay\_one)) == 0)

multi\_two = 0;

}

}

}

break;

default:

break;

}

if (count\_two++ < delay\_two || count\_one++ < delay\_one) {

continue;

}

if ((freeze\_one < 0) && (freeze\_two < 0)) {

continue;

}

if ((freeze\_one < 0)) {

count\_two = 0;

twist\_only\_two(dpy, win, gc[g], &arrow\_one, &arrow\_two);

continue;

}

if ((freeze\_two < 0)) {

count\_one = 0;

twist\_only\_one(dpy, win, gc[g], &arrow\_one, &arrow\_two);

continue;

}

count\_one = 0;

count\_two = 0;

twist\_all(dpy, win, gc[g], &arrow\_one, &arrow\_two);

}

}

{

rep5355(dpy, AutoRepeatModeOn);

XDestroyWindow(dpy, win);

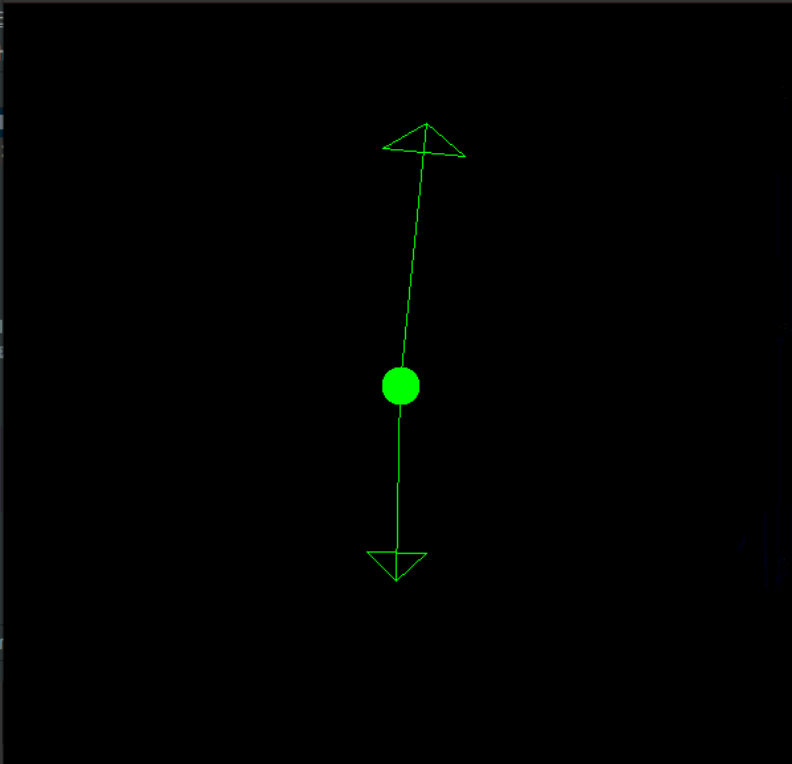
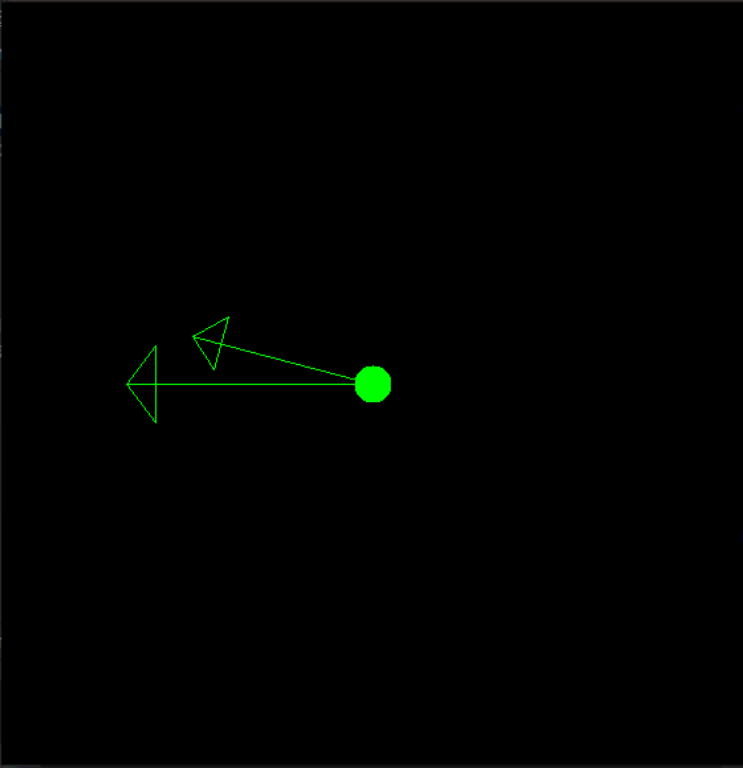
XCloseDisplay(dpy);

return (0);

} /\* exit block \*/

} /\* main \*/

**Тестирование**



**Список литературы**

1. Материалы учебного плана по предмету «Программирование графических приложений»

2. Волосатова Т.М., Родионов С. В., Беломойцев Д.Е. «Разработка графических приложений в среде X Window System»

3. Документация по библиотеке Xlib https://tronche.com/gui/x/xlib/ http://www.asvcorp.ru/tech/linux/xwinprg