Министерство образования и науки Российской Федерации
Московский Государственный Технический Университет (МГТУ)
имени Н. Э. Баумана
Факультет «Робототехника и комплексная автоматизация (РК)»
Кафедра «Системы автоматизированного проектирования (РК6)»

Отчет по рубежному контролю №2 по разработке графических приложений в среде X Window System

По курсу «Программирование графических приложений»

Вариант 5М

Выполнил: Студент Гусаров Аркадий Андреевич
Группа РК6-43Б

Проверил: _____
Дата _____
Подпись _____

Москва, 2021 г.

1. **Задание:**

   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 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 XIlib из состава X Window System.

2. **Цель работы:**

   Изучение базовых инструментальных средств разработки прикладного программного обеспечения в полиоконной системе X Window System. А также рассмотрение техники программирования фона графических окон с соответствующей обработкой событий и захватов указателя мыши в них библиотечными функциями программного интерфейса Xlib.

3. **Алгоритм работы:**

   В программе используется функция рисования в контексте главного окна, выполняющая отрисовку основываясь на массиве данных, которые могут изменяться в ходе работы программы.

   Далее будут кратко описаны функции, которые используются в программе, в том порядке, в котором они идут.

   Сначала получаем предварительные сведения из дефолтных настроек.

   XOpenDisplay - установление связи с графическим терминалом (X-сервером).

   DefaultRootWindow - макрос возвращает корневое окно для экрана по умолчанию.

   DefaultScreen - макрос возвращает номер экрана в подпрограмме XopenDisplay.

   DefaultDepth - макрос возвращает глубину цвета (количество разрядов, относящихся для кодирования цвета в окне) для указанного экрана.

DefaultVisual - макрос возвращает визуальные характеристики окна для указанного экрана.
Создаем графический образ.
XCreatePixmap - создает графический образ с указанными размерами и глубиной цветности, возвращая его идентификатор.
Имеет 5 аргументов:

- display - указатель на структуру, описывающую соединение с X-сервером;
- d - окно или графический образ для определения экрана сервера;
- width and height - размеры графического образа;
- depth - глубина цветности графического образа.

DefaultGC - возвращает графический контекст по умолчанию для корневого окна указанного экрана.
Сначала выполняется инициализация массива данных, затем при вызове события перерисовки функция отрисовки выполняет рисование треугольных элементов учитывая их смещение и положение диагонали в клетке между треугольниками.
XSetBackground(display, gc, background) - устанавливает фоновый цвет для графического контекста gc.
Background - индекс фонового цвета.
XSetForeground (display, gc, foreground) - устанавливает цвет графических примитив при выводе их через графический контекст gc.
Foreground - индекс цвета для вывода графических примитивов
XFillRectangle (display, d, gc, x, y, width, height) - рисует заполненный прямоугольник, левый верхний угол которого находится в координатах (x, y), а размеры равны (width, height).
XDrawLine (display, d, gc, x1, y1, x2, y2) - рисует прямую линию между двумя указанными точками (x1, y1), (x2, y2).
XDrawPoint (display, d, gc, x, y) - рисует точку в указанных координатах.Так как точка на графическом окне отображается очень мелко - была использована функция рисования дуги.
XFillArc (display, d, gc, x, y, width, height, angle1, angle2) - рисует дугу, образованную бесконечно тонким эллипсом, вписанным в прямоугольник.
Заполняем атрибуты окна.
С помощью маски меняем некоторые параметры из аргумента attributes, а именно:

- override_redirect - размер окна и его положение смогут меняться только с помощью менеджера окон, так как мы присваиваем ему значение False.
- background_pixmap - фон окна будет задаваться картинкой (картой пикселей).

Создаем графическое окно.
XCreateWindow - создает включенное окно-потомок для указанного окна-предка, возвращая идентификатор созданного окна.
Имеет 8 аргументов:

- display - указатель на структуру, описывающую соединение с X-сервером;
- parent - идентификатор родительского окна;
- x, y - координата левого верхнего угла;
- width and height - размеры графического образа;
- border_width - толщина рамки вокруг окна;
- depth - количество разрядов, отводящихся для кодирования цвета в окне;

• class - класс окна. Допустимые значения InputOutput, InputOnly, CopyFromParent;

• visual - визуальные характеристики окна;

• valuemask - маска, указывающая, какие параметры из аргумента attributes следует указывать;

• attributes - расширенные параметры окна.

XStoreName - устанавливает заголовок окна.

XMapWindow - делает указанное окно видимым.

XFlush - принудительная передача содержимого буфера вывода.

Потом прописываем маску событий и функцию инверсии фона.

Затем следует цикл while, реагирующий на изменение положение курсора в поле графического окна и за его пределами. При нажатии любой клавиши на клавиатуре происходит выход из графического окна.

XDestroyWindow - удаляет окно.

XCloseDisplay - прерывает соединение с X-сервером.

4. Код программы:

- Файл 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
```

- Файл 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);
}
```

- Файл main.c:

```
#include <X11/Xlib.h>
#include <X11/Xutil.h>
#include <stdio.h>
#include "clock.h"

/* Main function */

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);
        }
      } /* multi block */
      {
        /* Exit block */
        rep5355(dpy, AutoRepeatModeOn);
        XDestroyWindow(dpy, win);
        XCloseDisplay(dpy);
        return (0);
      } /* exit block */
    } /* main */
```

5. Список литературы:

1. O'Reilly & Associates, Inc. - Table of contents for Xlib Programming Manual (O'Reilly & Associates, Inc.): Режим доступа к ст. http://www.sbin.org/doc/Xlib/index_contents.html.
2. Adrian Nye - Volume One: Xlib Programming Manual: Режим доступа к ст. http://www.ac3.edu.au/SGI_Developer/books/XLib_PG/sgi_html.
3. Вадим Годунко - Xlib - интерфейс с X Window на языке C : Режим доступа к ст. http://motif.opennet.ru/book3.html..
4. Kluwer Academic Publishers - Fundamentals of X Programming GUI and Beyond.pdf Режим доступа http://ftp.homei.net.ua/index.
5. Kenton Lee - Technical Window System and Motif WWW Sites: Режим доступа к ст. http://www.rahul.net/kenton/xsites.html..
6. Robert W. Scheifler - RFC 1013 - X Window System Protocol. Режим доступа к ст. http://www.apps.ietf.org/rfc/rfc1013.html.
7. Theo Pavlidis - Fundamentals of X Programming GUI and Beyond. Режим доступа http://www.maives.ru/modules/news.