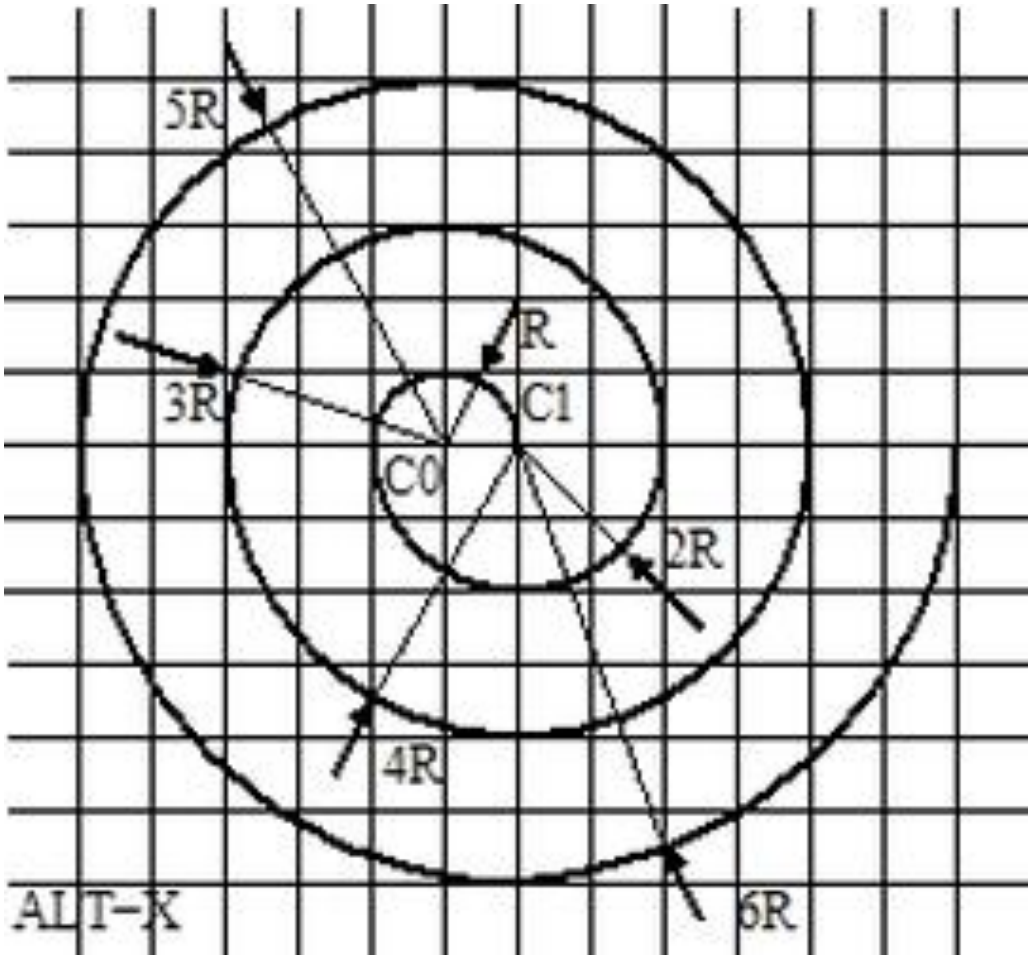
МУЛЬТИПЛИКАЦИЯ ЗАВИТКОВ СПИРАЛИ

В инженерной графике завитками называются плоские кривые из сопряженных дуг окружности, которые по форме приближаются к спирали. В технической практике они обычно применяются для проектирования спиральных пружин и удобны для компьютерной интерполяции и аппроксимации по точкам. Формы завитков и способы их построения зависят от числа центров сопряженных окружностей. Следующий рисунок иллюстрирует конфигурацию двухцентрового завитка, который требуется по заданию:

Рисунок конфигурации двухцентрового завитка спирали



Задание на разработку программы динамического построения спирали в графическом окне

Разработать программу динамического построения спирали в графическом окне фиксированного размера. Спиральная кривая должна формироваться по схеме двухцентрового завитка. В этом случае завитки спирали образуются парами сопряженных полуокружностей в диапазоне углов от 0 до 180 и от 180 до 360 градусов, X-координаты центров и радиусы которых отличаются на постоянную величину минимального радиуса. Во время выполнения программы изображение спирали должно периодически разворачиваться из центра до границ окна, а затем сворачиваться в точку в визуально различимом темпе, пока окно программы полностью видимо на экране. Указанный периодический процесс должен каждый раз приостанавливаться, когда окно программы оказывается частично или полностью перекрыто другими окнами текущей графической сессии. Кроме того, в программе должна быть предусмотрена интерактивная настройка темпа спиральных построений по нажатию клавиш "+" и "-" на цифровой клавиатуре.

Задание на разработку программы динамического построения спирали в графическом окне *(продолжение)*

Каждое нажатие указанных клавиш должно приводить, соответственно, к ускорению и замедлению темпа в 2 раза. Чтобы гарантировать однократную обработку по каждому физическому событию нажатия клавиш, нужно реализовать отключение режима автоповтора для них, когда окно программы получает фокус ввода. Когда фокус ввода передается любому другому окну графической сессии, режим автоповтора для них должен быть восстановлен. Завершение программы должно обеспечивать нажатие комбинации клавиш ALT-X на клавиатуре. Соответствующая подсказка должна постоянно отображаться в левом нижнем углу окна программы. При указанном корректном завершении программы с этой же целью должна быть исключена возможность принудительно закрыть окно программы для ее аварийного завершения. средствами оконного менеджера. При разработке программы следует предусмотреть асинхронное управление очередью событий от клавиатуры и видимости для графического окна программы, а также обеспечить обработку изображений в нем с помощью библиотечных функций программного интерфейса X Window System.

Код программы мультипликации завитка

Исходный код программы мультипликации завитка спирали составляют два модуля из прикладных функций и основной функции, а также заголовочный файл "spiral2x.h", который включается в них директивой include. Он содержит следующую декларацию структуры завитка XPiAr2, которая определятся директивой typedef:

```
typedef struct {
  int Rm;        /* Максимальный радиус полукруга завитка */
  int A;         /* Угловая координата 0 ≤ A ≤ 360 градусов */
  int R;                    /* Радиус полукруга 0 ≤ R ≤ Rm */
  int dA;                   /* Дуговой шаг завитка |dA| = 1 */
  int dR;                   /* Радиальный полушаг |dR| = R0 */
  XPoint C[2]; /* Координаты центров C[1].x – C[0].x = |dR| */
} XPiAr2;
```

Структура завитка XPiAr2

В этой завитковой структуре радиальное (R) и угловое (A) поля обозначают полярные координаты точки спирали. Их значения вычисляются относительно центров полукругов завитков, оконные координаты которых фиксирует пара графических структур XPoint центрового массива C[2]. Горизонтальная дистанция между ними имеет постоянное по модулю значение |dR|, которое задает радиальный полушаг спирали и равно половине зазора между соседними завитками. Из геометрии спирали оно также равно разности радиусов смежных завитков и радиусу полукруга начального завитка (R0 = |dR|). Дуговой шаг по завитку задает значение поля dA, которое сегментирует его полукруги дуговыми фрагментами. При этом пара (A, A+dA) определяет угловое положение такого фрагмента и, в частности, концевой дуги текущего изображения спирали.

Значения радиального и дугового (полу) шагов и их синхронизация друг с другом

Знаки значений радиального и дугового (полу)шагов синхронизируются друг с другом в циклах мультипликации завитков. В частности, для раскрутки спирали (против часовой стрелки) до максимального радиуса Rm устанавливаются положительные значения dR>0 и dA>0. При закрутке спирали (по часовой стрелке) в центральную точку, их значения должны быть оба отрицательны (dR<0 и dA<0). В периодическом процессе таких сжатий и расширений угловые координаты (A) меняются в диапазоне от 0 до 360 градусов с дискретностью дугового шага |dA| = 1. Радиальные координаты (R) меняются с дискретностью радиального полушага dR и остаются в пределах от 0 до Rm=(2*|dR|*N), где N обозначает число (за)витков. Рассмотренная завитковая структура XPiAr2 адресуется из основной функции main функциями прикладного модуля.

Превентивное объявление прототипов завершает заголовочный файл следующими спецификациями:

```
int maxisize(XPiAr2*, char*);              /* Размаха спирали */
int reset(XPiAr2*);           /* Начальная установка спирали */
int decent(XPiAr2*);        /* Координаты центров завитков */
int redraw(XEvent*, GC, XPiAr2*); /* перерисовка спирали */
int reverse(XPiAr2*);                /* Реверс мультипликации */
int amod2pi(XPiAr2*);      /* Ограничить угловой диапазон */
int recent(XPiAr2*);      /* Смена центра полукруга завитка */
int twist(Display*, Window, GC, XPiAr2*); /* Рас(за)крутка */
int rep5355(Display*, int);        /* Автоповтор клавиш + и − */
int rapid(XEvent*, int);              /* Темп мультипликаций */
int overlap(XEvent*);            /* Перекрытие окна спирали */
```

Стандартные заголовки X-графики

- Прикладной модуль составляют все завитковые и управляющие прикладные функции программы, которые вызывает ее основная функция main. Исходный код этого прикладного модуля начинается подключением стандартных заголовков X-графики и текстовой обработки символьных строк, а также прикладного заголовка с декларацией завитковой структуры, следующими директивами:

```
#include <X11/Xlib.h>
#include <X11/keysym.h>
#include <X11/keysymdef.h>
#include <X11/string.h>
#include "spiral2x.h"
```

Исходные данные программы

Исходными данными программы являются модуль полушага завитка, равный радиусу начального завитка R0 и число (за)витков N. Они передаются программе аргументом основной функции main в формате строки "R0xN" или "16x8" по умолчанию. В любом случае функция main, переадресует эту строку исходных данных вместе с завитковой структурой XPiAr2 для разбора в функцию maxisize. Она преобразует исходные данные в числовой формат, используя геометрическую утилиту XParseGeometry, и вычисляет по ним максимальный радиус завитка (Rm=2*N*R0). После автоконтроля своих величин полученные числовые результаты сохраняются в соответствующих полях Rm и R=dR=R0 завитковой структуры. Кроме того, значение максимального радиуса Rm возвращается в основную функцию main для возможности контроля максимального размаха спирали. Рассмотренная функция maxisize имеет следующий исходный текст.

Исходный текст функции maxisize ()

```
/* Контроль максимального радиуса внешнего завитка */

int maxisize(XPiAr2* pr, char* R0xN) {
int R0;                    /* Радиус внутреннего завитка */
int N;                          /* Число витков спирали */
int empty;          /* Пустое поле для координат x и y */
XParseGeometry(R0xN, &empty, &empty, &R0, &N);
if(((pr–>dR = pr->R = R0) < 1) || (N < 1))
  N = R0 = 0;
return(pr->Rm = 2*R0*N);     /* возврат max радиуса */
} /* maxisize */
```

Описание функции decent ()

Для продолжения инициализации функция main переадресует завитковую структуру XPiAr2 функции decent. Она вычисляет горизонтальный габарит графического окна спирали и фиксирует в нем координаты центров завитков. Для этого используются текущие значения полей Rm и R=dR=R0, куда функция maxisize записала радиусы внешнего и внутреннего завитков, который одновременно равен горизонтальному расстоянию между центрами завитков. При этом также обеспечивается симметричное расположение центров в окне и запас по 8 пикселей с каждого края. Полученные координаты записываются в центровые поля C[0] и C[1] завитковой структуры XPiAr2. Габаритный размер возвращается в функцию main, чтобы фиксировать ширину окна спирали. Исходный текст функции decent имеет следующий вид.

Исходный текст функции decent ()

Исходный текст функции decent имеет следующий вид.

```
int decent(XPiAr2* pr) {        /* Расчет центров завитков */
int w = 2*(pr−>Rm + pr−>R) + (8 + 8); /* R = R0 = dR now */
pr−>c[0].x = w/2 - (pr−>R/2);
pr−>c[1].x = pr−>c[0].x + pr−>R;      /* = w/2 + (pr−>R/2); */
pr−>c[0].y = pr−>c[1].y = w/2 + 8;
return(w);                    /* возврат ширины рамки спирали */
} /* decent */
```

Описание функции reset()

Чтобы начать построение спирали с внутреннего завитка вызывается функция reset. Ей адресуется структура завитка XPiAr2 для инициализации полярных координат A=R=0 в центре верхнего (левого) полукруга и задается дуговой шаг величиной dA=1 угловой градус. Кроме того, из алгоритмических соображений необходимо гарантировать значение радиального полушага dR>0. Функция reset вызывается из функции main для начальной инициализации спирали, а также функцией redraw для ее перерисовки при потере изображения. В любом случае она возвращает вертикальный габарит для окна спирали, который вычисляется по вертикальной координате из центрового поля структуры завитка XPiAr2.

Исходный код функции reset ()

Исходный код функции reset имеет следующий вид.

```
int reset(XPiAr2* pr) {    /* Начальная установка спирали */
pr−>A = (0*64);  pr−>dA = (1*64);  pr−>R = 0;
if(pr−>dR < 0)
  pr−>dR = -pr−>dR;
return(2*(pr−>c[0].y));  /* возврат высоты рамки спирали */
} /* reset */
```

Описание функции reverse ()

Рассмотренные прикладные функции maxisize, decent и reset вызываются для инициализации спирали. Другие завитковые функции reverse, amod2pi, recent и twist обеспечивают циклический процесс ее мультипликации. Самой простой из них является функция reverse, необходимая для контроля и обработки моментов, когда спираль расширяется до предельного радиуса (R=Rm) или сворачивается в точку (R=0). В обоих случаях функция reverse инвертирует знаки радиального dR и дугового dA (полу)шагов в соответствующих полях адресованной структуры завитка XPiAr2. Их текущие знаки кодируются значениями 0(+) и 1(−) при возврате в функцию main для индексации графических контекстов спирали.

Исходный текст функции reverse ()

Исходный текст функции reverse имеет следующий вид:

```
int reverse(XPiAr2* pr) {      /* Реверс мультипликаций */
int g = 2;     /* индекс графического контекста спирали */
if((pr−>R > pr−>Rm) || (pr−>R == 0)) {
  pr−>dR = -pr->dR;  pr−>dA = -pr−>dA;
  g = (pr−>dA > 0) ? 0 : 1;
} /* if */
return(g);
} /* reverse */
```

Определение индекса центра полукруга завитка

Функция recent адресует структуру завитка XPiAr2 для контроля центра его полукруга по угловой координате (A). Если ее значение кратно 180, происходит смена центра по угловой координате (A+dA) после дугового шага. При этом радиальная координата R также изменяется на величину радиального полушага dR. В любом случае возвращается индекс центра полукруга завитка 0 или 1, который определяется по следующей таблице:

Угловая координата (A)	0	180	180	360
Знак дугового шага (dA)	+	-	+	-
Индекс центра завитка (C)	0	0	1	1

Исходный текст функции recent ()

Исходный текст функции recent , который реализует эту таблицу, имеет следующий вид:

```
int recent(XPiAr2* pr) {/* Смена центра полукруга завитка */
if((pr−>A % (180*64)) != 0)
  return(pr->A / (180*64));
pr->R += pr->dR;
return((pr−>A + pr−>dA) / (180*64));
} /* recent */
```

Функция amod2pi ()

Функция amod2pi адресует структуру завитка XPiAr2 для циклического изменения угловой координаты (A) полукругов спирали на величину дугового шага dA в диапазоне результирующих значений (A+dA) от 0 до 360 угловых градусов. Результат вычислений передается через код возврата функции, а ее исходный текст имеет следующий вид.

```
 /* Ограничить дуговой угол диапазоном от 0 до 360 */
int amod2pi(XPiAr2* 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);
} /* amod2pi */
```

Описание функции twist ()

Обращение к функциям amod2pi и recent происходит из функции twist, которая вызывается для перерисовки и для мультипликации изображения спирали в графическом окне. Поэтому кроме завитковой структуры XPiAr2 ей передаются графические параметры адреса структуры дисплея (Dispay), идентификатор окна (Window) и графический контекст изображения (GC). При каждом вызове функция twist использует графический запрос XDrawArc, чтобы изменить дуговой фрагмент углового шага по внешнему завитку. При этом его полукруг идентифицирует возврат функции recent, а характер изменения определяет знак дугового шага dA, с которым согласован цвет изображения в графическом контексте. Если dA>0, спираль удлиняется на дугу углового шага, которая дорисовывается цветом своего изображения на фоне окна. Если dA<0, спираль сокращается на дугу углового шага, которая перерисовывается цветом фона окна и, следовательно, становится невидна, то есть стирается. В обоих случаях рисование дуги реализуется по запросу XDrawArc с параметрами полей структуры завитка XPiAr2 и соответствующими графическими аргументами вызова функции twist. После этого вызывается функция amod2pi, чтобы изменить угловую координату (A) изображения спирали на величину дугового шага dA, а его радиальная координата (R) возвращается для контроля размаха спирали.

Исходный текст функции twist ()

Исходный текст рассмотренной функции twist имеет следующий вид.

```
/* Отобразить 1 дуговой шаг за(раскрутки) спирали */
int twist(Display* dpy, Window win, GC gc, XPiAr2* pr) {
int i = recent(pr);        /* индекс центра полукруга завитка */
int R2 = (2*pr->R);      /* двойной радиус завитка */
XDrawArc(dpy, win, gc, pr->c[i].x - pr->R, pr->c[i].y - pr->R, R2, R2, pr->A, pr->dA);
XFlush(dpy);
amod2pi(pr);
return(pr->R);  } /* twist */
```

Кроме основного вызова в цикле мультипликаций, функция twist используется в функции redraw для перерисовки завитков спирали при потере изображения в графическом окне. Эта функция предусмотрена для отработки соответствующих событий типа Expose и адресует их стандартную структуру XEvent вместе с завитковой структурой XPiAr2 и графическим контекстом рисования изображения спирали GC(0).

Перерисовка спирали

Для перерисовки спирали сначала создается копия адресованной структуры завитков, чтобы получить координаты их центров и радиальный полушаг. Для копирования может быть использована стандартная библиотечная функция memcpy. Остальные поля копии инициализирует вызов функции reset , чтобы начать перерисовку спирали из центра до полярных координат (R,A) конечной точки потерянного изображения ее оригинала. Такую перерисовку реализует циклический вызов функции twist до возврата радиуса внешнего завитка оригинала и дорисовка по дуговой координате его полукруга за еще 1 вызов функции twist. В обоих случаях функция twist адресует завитковую структуру копии спирали и графический контекст рисования, а дисплейный адрес и идентификатор окна подставляются из соответствующих полей структуры события.

Кроме спирали, функция redraw осуществляет перерисовку подсказки строки выхода "ALT-X" в юго-западном углу окна по запросу XDrawString. Вертикальный габарит окна, который нужен для позиционирования подсказки, устанавливается по возврату функции reset при инициализации перерисовки копии спирали.

Оптимизация серийных перерисовок

Для оптимизации серийных перерисовок функция redraw использует технику отсечения графического вывода. Область отсечения формируется из объединения прямоугольников, которые покрывают часть окна, где потеряно изображение. Они получаются из структуры события Expose и накапливаются в массиве прямоугольных структур XRectangle для установки в графический контекст по запросу XSetClipRectangles в конце серии. Графический вывод в таком контексте ограничен пикселями его многоугольной области отсечения. Избыточная перерисовка за границей области отсечения физически не реализуется, что сокращает объем графического вывода без каких-либо специальных геометрических мероприятий. После завершения перерисовки область отсечения исключается из графического контекста по запросу XSetClipMask с параметром None. При возврате из функции redraw после перерисовки также обнуляется ее счетчик прямоугольников области отсечения. Рассмотренная функция redraw имеет следующий исходный код:

Перерисовка фрагмента спирали из центра

Рассмотренная функция redraw имеет исходный код:

```
int redraw(Display* dpy, XEvent* ev, GC gc, XPiAr2* pr) {
int y;                          /* y-смещение подсказки ALT-X */
XPiAr2 r;                               /* для копии спирали */
static XRectangle clip[32];                 /* буфер отсечения */
static int n = 0;                          /* счетчик отсечений */
clip[n].x = ev->xexpose.x;  clip[n].y = ev->xexpose.y;
clip[n].width = ev->xexpose.width;
clip[n++].height = ev->xexpose.height;
if((ev->xexpose.count > 0) && (n < 32))
  return(0);
XSetClipRectangles(dpy, gc, 0, 0, clip, n, Unsorted);
r = *pr;   /* = memcpy(&r, pr, (sizeof(XPiAr2))); */
y = reset(&r) - 8; .        /* инициализация копии спирали */
while(twist(dpy, ev->xexpose.window, gc, &r) < pr->R);
r.dA = (pr->A - r.A);
twist(dpy, ev->xexpose.window, gc, &r);
XDrawString(dpy, ev->xexpose.window, gc, 8, y, "ALT-X", 5);
XSetClipMask(dpy, gc, None);
return(n=0); } /* redraw */
```

Интерактивное управление циклом мультипликации

3 прикладные функции overlap, rapid и rep5355 обеспечивают интерактивное управление циклом мультипликации.

Самая короткая из них функция *overlap()* вызывается при любых изменениях статуса видимости графического окна программы по событию VisibilityNotify, структуру XEvent которого адресует ее аргумент. Для контроля указанных изменений проверяется поле состояния state этой структуры. Любая величина в нем, кроме визуальной константы VisibilityUnObscured , означает, что окно становится полностью или частично невидимо. В этом случае возвращается отрицательный стоп-код = (−32), чтобы приостановить мультипликации спирали. Когда окно станет полностью видимо, функции overlap адресуется структура XEvent, значение поля состояния которой равно VisibilityUnObscured. В этом случае предусмотрен возврат 0, чтобы сбросить стоп-код и возобновить мультипликации. Исходный текст функции имеет вид.

Исходный код функции overlap()

```
int overlap(XEvent* ev) {
/* Контроль перекрытий спирали */
return((ev−>xvisibility.state != VisibilityUnobscured) ? −32 : 0;);
} /* overlap */
```

Управляющая функция rep5355()

Другая управляющая функция rep5355 предназначена для переключения режима автоповтора клавиш "+" и "–" на клавиатуре по своему аргументу. Его значение фиксируется в структуре управления клавиатурой XKeyboardControl. Затем в нее поочередно записываются физические коды указанных клавиш, которые определяются по запросу XKeysymToKeycode для их логических кодов. После записи каждого физического кода клавиатурная структура XKeyboardControl адресуется запросу XChangeKeyboardControl для регистрации X-сервером, чтобы включить или выключить автоповтор соответствующей клавиши. Исходный текст функции rep5355 имеет вид.

Исходный текст функции rep5355()

```
int rep5355(Display* dpy, int r) {      /* Автоповтор +/− */
XKeyboardControl kbval; /* структура контроля клавиатуры */
unsigned long kbmask = (KBKey | KBAutoRepeatMode);
kbval.key = XKeysymToKeycode(dpy, XK_KP_Add);
kbval.auto_repeat_mode = r;
XChangeKeyboardControl(dpy, kbmask, &kbval);
kbval.key = XKeysymToKeycode(dpy, XK_KP_Subtract);
XChangeKeyboardControl(dpy, kbmask, &kbval);
return(r);
} /* rep5355 */
```

Клавиатурная функция rapid()

Еще 1 клавиатурная функция rapid реализует управление темпом мультипликаций. При вызове ей адресуется структура клавиатурного события XEvent и положительный темповый параметр. Его текущая величина изменяется и возвращается в основную функцию main для ускорения или замедления мультипликаций. Кроме того, предусматривается неположительный возврат для приостановки и прерывания их процесса в цикле диспетчеризации функции main. Выбор альтернативы обработки и возврата устанавливает логический код нажатой клавиши, который образуется по запросу XLookupString из структуры XEvent полученного клавиатурного события для сравнения с макроопределениями X-клавиш в системном заголовке <keysymdef.h>

Задача клавиатурной обработки функции rapid()

Основной задачей клавиатурной обработки функции rapid является регулировка темпа мультипликации клавишами "+" и "−" на цифровой или основной клавиатуре с логическими кодами XK_KP_Add и XK_KP_Subtract или, соответственно, XK_Plus и XK_Minus. Каждое нажатие любой их этих клавиш изменяет темп мультипликации в два раза, логическим сдвигом текущего значения его параметра из аргумента вызова функции rapid в диапазоне целых чисел без знака от 1 до (2^32−1). Полученное значение возвращается в функцию main, где используется для временной задержки дуговых шагов по завиткам спирали, которая обратно пропорциональна угловой скорости мультипликаций, а возврат больших значений практически останавливает

Дополнительная обработка функции rapid()

Кроме того, предусмотрен возврат отрицательного значения (−32) при нажатии клавиши пробела с логическим кодом XK_space. Такой отрицательный возврат используется в функции main для установки клавиатурного стоп-кода, чтобы принудительно зафиксировать спираль независимо от значения параметра темпа. Для возобновления мультипликаций в прежнем темпе функция rapid осуществляет возврат параметра темпа без изменений при нажатии любой другой клавиши на клавиатуре. Его гарантированно положительное значение используется функцией main, чтобы сбросить отрицательный клавиатурный стоп-код мультипликации.

Специальная обработка предусмотрена на случай одновременного нажатия ком клавиш X и ALT. Распознавание клавиатурной комбинации ALT-X обеспечивает маскировка поля состояния state структуры XEvent полученного события по альтернативе логических кодов XK_X(x) клавиши X. При соответствии значения этого поля маске Mod1Mask модификатора ALT функция rapid возвращает 0. Такой 0-возврат используется в функции main для установки нулевого значения мульти-кода, которое прерывает цикл обработки событий и мультипликаций, чтобы корректно завершить выполнение программы. Рассмотренная функция rapid имеет следующий

```
/* Клавиатурное управление темпом мультипликации */
int rapid(XEvent* ev, int t) {
char sym[1];                          /* ASCII код символа */
KeySym code[1];                       /* X-код клавиши */
XLookupString((XKeyEvent* ) ev, NULL, 0, code, NULL);
switch(code[0]) {  case XK_plus:
  /* ускорить темп мультипликаций */
  case XK_KP_Add: /* (уменьшить задержку) */
                  if(t > 1)
                    t >>= 1;
                    break;
  case XK_minus:      /* замедлить темп мультипликаций */
  case XK_KP_Subtract: /* (увеличить задержку) */
                       if(t < (1 << 30))
                         t <<= 1;
                       break;
  case XK_space: /* остановить спираль пробелом */
                 t = (-32);
                 break;
  case XK_x:
  case XK_X:     /* выход по ALT-X */
```

Основная функция main()

Рассмотренные прикладные функции вызывает основная функция main , которая одна занимает отдельный основной модуль. Он начинается подключением пары системных заголовков X-графики и прикладного заголовка спирали следующими директивами:

```
#include <X11/Xlib.h>
#include <X11/Xutil.h>
#include "spiral2x.h"
```

Остальную часть основного модуля занимает исходный код основной функции main. Он делится на логические блоки, информационную связь которых обеспечивают формальные параметры стандартных командных аргументов функции main(argc, argv[]) и автоматические переменные системных графических типов {Display, Window, GC}, а также прикладной завитковой структуры XPiAr2. Блочную организацию функции main иллюстрирует следующий макро-код:

Исходный код функции main()

```
int main(int argc, char* argv[]) {
Display *dpy;    /* адрес дисплейной структуры */
Window win;     /* корневое окно спирали */
GC gc[2];          /* черный и белый графические контексты
*/
XPiAr2 helix;    /* структура завитка спирали */
{ /* Дисплейный блок */ }
{ /* Оконный блок */ }
{ /* Мульти блок */ }
{ /* Блок выхода */ }
} /*main*/
```

Описание аргументов функции main()

При вызове программы формальные аргументы функции main argc и argv обеспечивают передачу требуемых параметров спирали из командной строки по адресу argv[1] в формате строки "R0xN", где значения R0 и N обозначают, соответственно, радиальный полушаг завитка спирали и число ее (за)витков. При вызове программы без аргументов (argc=1) принимаются значения по умолчанию.

4 автоматические переменные внешнего main-блока определяют адрес структуры графического дисплея (Display), идентификатор корневого и основного окна программы (Window), пару графических контекстов (GC) для рисования и стирания изображения спирали, а также структуру ее завитков XPiAr2. Значения этих автоматических переменных будут доступны всем блокам.

Дисплейный блок

Дисплейный блок начинается графическим запросом XOpenDisplay , чтобы установить контакт с X-сервером по дисплейному адресу Display из внешнего main-блока. К нему применяется дисплейный макрос DefaultRootWindow, чтобы определить идентификатор корневого окна по умолчанию (Window). Эти значения используются парой одинаковых запросов XCreateGC, чтобы создать 2 графических контекста с параметрами по умолчанию, а их идентификаторы сохраняются в массиве GC для рисования [0] и стирания [1] завитков спирали в циклах мультипликации.

В стандартной структуре графического контекста рисования переустанавливается белый цвет изображения по графическому макросу WhitePixel и запросу XsetForeground (для контраста с черным по умолчанию цветом фона, который имеет пиксельный код 0). В этом графическом контексте также загружается и устанавливается шрифт надписей с размером (и алиасом названия) 9x15. С этой целью применяются графические запросы XLoadFont и XSetFont. Как правило стандартный шрифт "9x15" имеется в дистрибутиве X Window System, поэтому соответствующая проверка результата его загрузки не производится. Первый в паре графический контекст стирания, который имеет по умолчанию черный цвет изображения и фона, не изменяется. Исходный текст рассмотренного дисплейного блока имеет следующий вид:

Дисплейный блок

```
{ /* Дисплейный блок */
Font fn;                    /* идентификатор шрифта надписи */
unsigned long tone;      /* пиксельный тон для белого цвета */
dpy = XOpenDisplay(NULL);       /* Контакт с X-сервером */
scr = DefaultScreen(dpy);    /* Номер экрана по умолчанию */
win = DefaultRootWindow(dpy);    /* Корневое окно экрана */
gc[0] = XCreateGC(dpy, win, 0, 0);       /* Пара графических */
gc[1] = XCreateGC(dpy, win, 0, 0);              /* контекстов */
tone = WhitePixel(dpy, scr);        /* Белый код = 0xFFFFFF; */
XSetForeground(dpy, gc[0], tone);    /* Задать белый цвет и */
fn = XLoadFont(dpy, "9x15");              /* Загрузить шрифт */
XSetFont(dpy, gc[0], fn);          /* Установить шрифт в GC */
} /* Display block */
```

Оконный блок

- В следующем оконном блоке создается графическое окно программы. Его габаритные размеры определяются по параметрам спирали в строке формального аргумента argv[1]. При вызове программы без аргумента, argv[1] переадресуется значением константной строки "16x8" по умолчанию. В любом случае для габаритной обработки вызываются прикладные функции maxisize, decent и reset, которые адресуют и инициализируют завитковую структуру XPiAr2 спирали. При этом функция maxisize возвращает максимальный радиус внешнего завитка для контроля величины и переинициализации с параметрами по умолчанию. Возвраты функций decent и reset устанавливают минимальный габарит графического окна, в котором могут быть симметрично размещены полукруги внешнего завитка с максимальным радиусом.
-
- Полученные значения передаются запросу XCreateWindow, по которому создается графическое окно необходимого размера для отображения спирали. Оно становится подокном корневого окна экрана, наследуя его визуальный класс и глубину цветовых плоскостей по паре параметров CopyFromParent. Кроме того, формально задаются нулевые координаты и толщина рамки 1, а через структуру оконных атрибутов XSetWindowAttributes адресуются пиксельный код фона окна в ее поле background_pixel и внешнее обрамление оконного менеджера, установкой значения False в ее поле override_redirect. При этом (черный по умолчанию) цвет фона определяется из структуры XGCValues любого графического контекста

Оконная переменная (Window) внешнего блока

Идентификатор созданного окна записывается в оконную переменную (Window) внешнего блока вместо идентификатора корневого окна экрана. Размеры внешнего обрамления окна записываются в структуру XSizeHints и фиксируются установкой свойства WM_NORMAL_HINTS оконного менеджера по запросу XSetNormalHints. Кроме этих геометрических рекомендаций, оконному менеджеру задается протокол его сообщений окну программы по свойству WM_PROTOCOLS. В протокольном наборе фиксируется только свойство WM_DELETE_WINDOW для посылки сообщения при попытке закрыть окно программы через его обрамление. Атомный номер указанного свойства по его имени возвращает утилита XInternAtom. Он адресуется утилите XSetWMProtocols для записи в протокол сообщений окну программы, чтобы исключить возможность интерактивно закрыть его без завершающих действий выходного блока. При этом окно не закрывается, а сообщение просто игнорируется при обработке событий в мульти-блоке. Оконный блок завершается отображением окна спирали по запросу XmapWindow. Рассмотренный оконный блок имеет следующий исходный код.

Исходный код оконного блока

```
{ /* Оконный блок */
unsigned w, h;                          /* габариты окна спирали */
XSetWindowAttributes attr;       /* структура атрибутов окна */
XGCValues gval;         /* структура графического контекста */
unsigned long amask;                    /* маска атрибутов окна */
Window root = win;          /* идентификатор корневого окна */
XSizeHints hint;                /* Геометрические свойства WM */
Atom wdw[1];                    /* WM-атом удаления окна по X */
if(argc < 2)
  argv[1] = "16x8";       /* Параметры спирали по умолчанию */
while(1) { /* Установка габарита окна и параметров спирали */
  if(maxisize(&r, argv[1]) == 0)
    maxisize(&r, argv[1] = "16x8");
  w = decent(&r);
  if((h = reset(&r)) < DisplayHeight(dpy, scr))
    break;
  argv[1] = "16x8";
} /* while */
amask = (CWOverrideRedirect | CWBackPixel); /* фон и WM */
XGetGCValues(dpy, gc[1], GCBackground, &gval);
attr.background_pixel = gval.background;  /* Черный фон = 0 */
attr.override_redirect = False;             /* WM-контроль окна */
win = XCreateWindow(dpy, root, 0, 0, w, h, 1, CopyFromParent,
                        InputOutput, CopyFromParent, amask, &attr);
hint.flags = (PMinSize | PMaxSize); /* фиксировать габариты */
```

Мульти-блок функции main()

Мульти-блок функции main реализует цикл асинхронной обработки событий на фоне периодического процесса мультипликации завитков спирали в графическом окне программы. Допустимый набор событий устанавливается априорным запросом XSelectInput по заданной маске, которая разрешает обработку событий в окне программы от нажатия клавиш на клавиатуре (KeyPress), (де) фокусировки ввода (FocusIn и FocusOut), изменения визуального статуса окна (VisibilityNotify), а также потери (части) его изображения (Expose). Для их получения из очереди X-сервера в цикле обработки событий используется асинхронный запрос XCheckWindowEvent с маской указанных событий для окна программы. Он возвращает 0 без блокировки программы, если в очереди нет подходящих по маске событий. Каждое полученное по маске событие адресуется для прикладной обработки функциями управления мультипликацией спирали. Выбор альтернативы обработки определяется типом полученного события.

Мульти-блок функции main()

В частности, при потере изображения в окне по событию типа Expose, его структура XEvent адресуется функции redraw вместе с завитковой структурой XPiAr2 и графическим контекстом рисования (0). Она обеспечивает буферизованную перерисовку спирали из центра с отсечением областей, где сохранилось ее изображение.

Для обработки события VisibilityNotify вызывается функция overlap, которая адресует его структуру XEvent. Своим отрицательным возвратом она обеспечивает остановку мультипликаций спирали, пока окно программы не будет полностью видимо на экране. Как только это произойдет, новый вызов функции overlap сбросит ее стоп-код мультипликации freeze своим нулевым возвратом.

События FocusIn и FocusOut программа получает, когда меняется фокус ввода ее окна. Для их обработки вызывается функция rep5355 с аргументом AutoRepeatModeOff или AutoRepeatModeOn, чтобы выключить или включить режим автоповтора нажатия клавиш '+' и '−' на клавиатуре, когда окно программы получает или, соответственно, передает (теряет) фокус ввода.

Обработку клавиатурных событий типа KeyPress при нажатии клавиш реализует функция rapid , которой адресуется их структура XEvent и параметр темпа мультипликации. Ее возврат фиксирует управляющий мульти-код multi, который используется в зависимости от своей величины. Его равно 0 при выходе или больше 0 в процессе мультипликаций.

Мульти-блок функции main()

Положительное значение мульти-кода устанавливает степень задержки мультипликации delay по счетчику итераций count цикла обработки событий. Значение этого счетчика увеличивается на 1 в каждой итерации. Когда оно достигает уровня задержки, вызывается функция twist с графическими параметрами и завитковой структурой XPiAr2 для дугового шага по спирали. Следом за ней вызывается функция reverse, которой адресуется завитковая структура XPiAr2 для встроенного габаритного контроля спирали и реверса мультипликаций в предельном положении. Ее возврат (пере)индексирует графический контекст спирали. С ним сразу вызывается функция twist для дугового шага в обратном направлении, чтобы компенсировать радиальный проскок предельных положений спирали. Независимо от реверса счетчик итераций count обнуляется для отсчета задержки до следующего дугового шага по спирали через delay итераций цикла обработки событий.

Продолжительность задержки может быть переустановлена регулировкой положительного мульти кода возврата функции rapid по клавиатурным событиям нажатия клавиш '+' и '–' для ускорения или замедления темпа мультипликации, соответственно. Возврат отрицательного мульти-кода (по нажатию клавиши пробела) фиксирует спираль, аналогично событию VisibilityNotify. В обоих случаях требуемый эффект достигается обнулением счетчика итераций count на каждой итерации цикла обработки событий перед сравнением с задержкой delay. Это исключает вызов функции twist. Поэтому спираль не может изменяться до перезаписи мульти-кода положительным возвратом функции rapid при нажатии любой клавиши, чтобы возобновить мультипликацию. Нулевой мульти-код возврата функции rapid (при нажатии комбинации клавиш ALT-X на клавиатуре) прерывает цикл обработки событий в мульти-блоке функции main. Исходный текст мульти-блока имеет следующий вид.

Исходный текст мульти-блока

```
{ /* Мульти-блок */
unsigned long emask;                    /* маска событий */
XEvent event;                           /* структура событий */
int freeze = 0;                         /* визи-стоп спирали */
unsigned delay = (1<<12);       /* период мульти-задержки */
int multi = (1<<12);                    /* мульти-код */
int count = 0;                  /* счетчик мульти-задержки */
int g = 0;                      /* индекс графического контекста */

emask = (ExposureMask | KeyPressMask | FocusChangeMask |
       VisibilityChangeMask);       /* задать маску событий */
XSelectInput(dpy, win, emask);              /* для обработки */

while(multi != 0) {      /* асинхронная обработка событий */
  event.type = 0;     /* сбросить тип предыдущего события */
  XCheckWindowEvent(dpy, win, emask, &event);
  switch(event.type) {                  /* анализ типа события */
    case Expose: redraw(&event, gc[0], &r);  /* перерисовка */
                 break;             /*при потере изображения */
    case VisibilityNotify: freeze = overlap(&event);
                           break;   /* визи−стоп код спирали */
       case FocusIn: rep5355(dpy, AutoRepeatModeOff);
                     break;    /* отключить авто-повтор + и − */
       case FocusOut: rep5355(dpy, AutoRepeatModeOn);
                      break;          /* включить авто-повтор */
```

Выходной блок функции main

Выходной блок функции main реализует корректное завершение программы. Сначала в нем вызывается функция rep5355 с аргументом AutoRepeatModeOn, чтобы восстановить режим автоповтора нажатия клавиш '+' и '−' на клавиатуре. Затем по запросам XDestroyWindow и XCloseDisplay закрывается окно программы и разрывается ее связь с X-сервером. После этого программа завершается с кодом 0. Перечисленные инструкции составляют следующий исходный текст блока выхода функции main.

```
{ /* Блок выхода */
rep5355(dpy, AutoRepeatModeOn); /* автоповтор + и − */
XDestroyWindow(dpy, win);   /* Закрыть окно спирали */
XCloseDisplay(dpy);                 /* Разрыв X-контакта */
return(0);
} /* exit block */
```

Исходный текст прикладного и основного модулей рассмотренной программы мультипликации спирали сохраняется в текстовых файлах spiral21.c и spiral22.c в одном каталоге с их прикладным заголовком (spiral2x.h). Для их компиляции и компоновки с графической библиотекой требуется выполнить следующую команду:

```
$ cc -o xspiral2 spiral21.c spiral22.c –lX11
```

В результате будет получен выполняемый файл xspiral2, который может быть вызван, например, следующей командной строкой:

```
$ xspiral2 8x32 &
```

При таком формате вызова в графическом окне программы будет отображаться периодический процесс мультипликаций спирали, которую составляют 32 завитка с промежутками по 8 пикселей между ними. При вызове программы без аргумента устанавливаются параметры по умолчанию "16x8". В любом случае клавишами '+' и '−' на клавиатуре можно настроить желаемый темп мультипликации с учетом быстродействия своего компьютера.