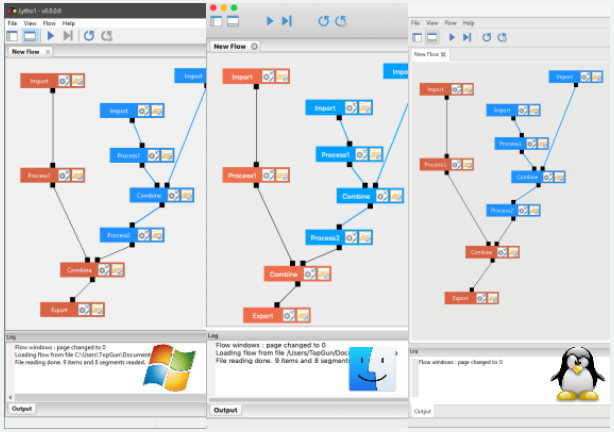
МИЭТ
Национальный исследовательский университет «МИЭТ»
Кафедра ПКИМС

Кроссплатформенная разработка программного обеспечения

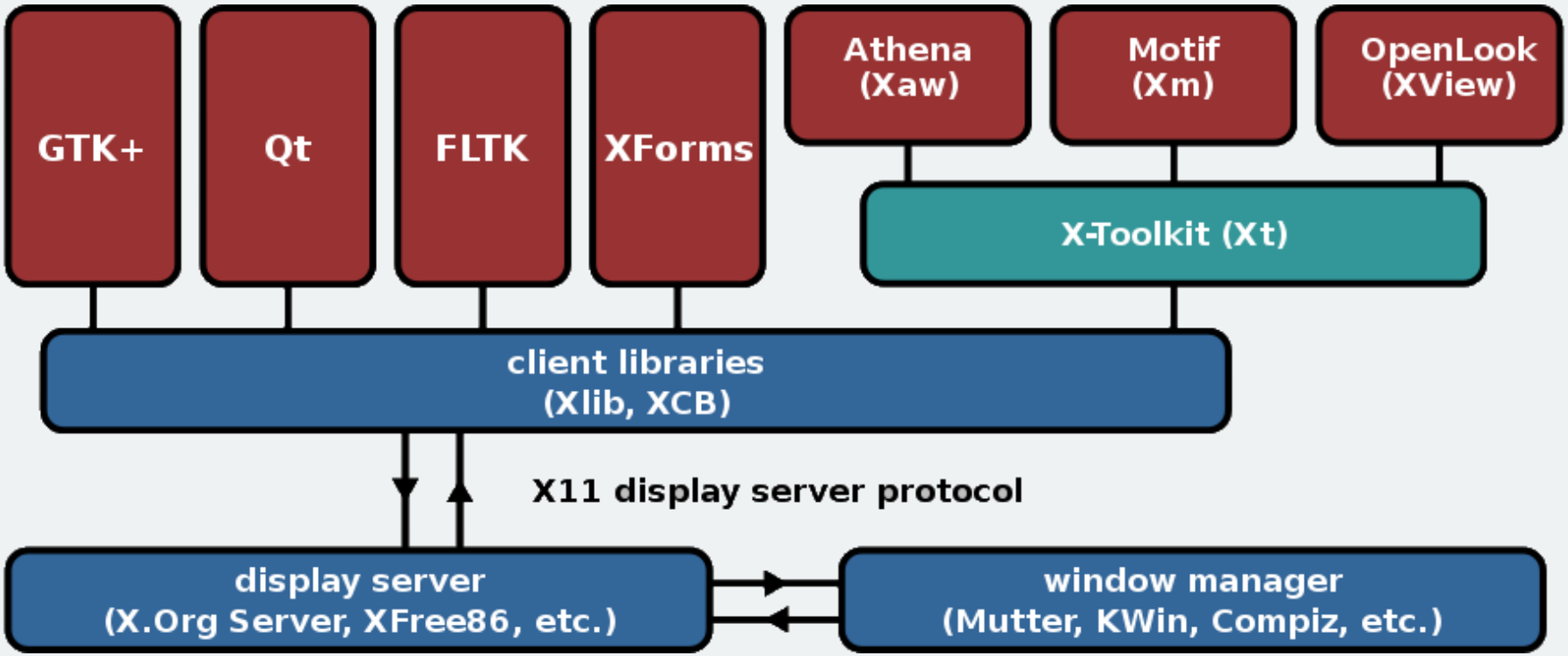


Лабораторная работа №2

Основы программирования под X Window System. Основы GTK.

Библиотека Xlib

Xlib – библиотека, написанная на C++, отвечающая за взаимодействие с X Server



Архитектура приложения с использованием Xlib: код

main_00.cpp

```
#include <X11/Xlib.h>
#include <stdio.h>
#include <stdlib.h>

int main(int, char *[]) {
  Display *p_display = XOpenDisplay(NULL);
  if (!p_display) {
    printf("Can’t open display.\n");
    return EXIT_FAILURE;
  }
  Window window = XCreateSimpleWindow(p_display,
                                      XDefaultRootWindow(p_display),
                                      100, 100, 200, 200,
                                      4, 0, 0);
  XMapWindow(p_display, window);
  XSelectInput(p_display, window, NoEventMask);
  XEvent event;
  for (;;) {
    XNextEvent(p_display, &event);
  }
  XDestroyWindow(p_display, window);
  XCloseDisplay(p_display);
  return EXIT_SUCCESS;
}
```

Маски событий

Какие события хотим получать:

`NoEventMask`	Никакие
`KeyPressMask`	Нажата клавиша
`KeyReleaseMask`	Отпущена клавиша
`ButtonPressMask`	Нажата кнопка мыши
`ButtonReleaseMask`	Отпущена кнопка мыши
`EnterWindowMask`	Курсор вошёл в зону окна
`LeaveWindowMask`	Курсор вышел из зоны окна
`PointerMotionMask`	Курсор движется над поверхностью окна
`Button1MotionMask`	Курсор движется, пока нажата кнопка 1
`Button2MotionMask`	Курсор движется, пока нажата кнопка 2
`Button3MotionMask`	Курсор движется, пока нажата кнопка 3
`Button4MotionMask`	Курсор движется, пока нажата кнопка 4
`Button5MotionMask`	Курсор движется, пока нажата кнопка 5
`ButtonMotionMask`	Курсор движется, когда нажата любая кнопка
`ExposureMask`	Нужно перерисовать окно
`StructureNotifyMask`	Поменялась информация о структуре окна
…	

Ловим события от X Server

```
for (;;) {
   XNextEvent(p_display, &event);

   if (event.type == KeyPress)
     printf("A key was pressed");
   if (event.type == ButtonPress)
     printf("Mouse button was pressed");
 }
```

Поля объединения XEvent

```
typedef union _XEvent {
    int type;
    XAnyEvent xany;
    XKeyEvent xkey;
    XButtonEvent xbutton;
    XMotionEvent xmotion;
    XCrossingEvent xcrossing;
    XFocusChangeEvent xfocus;
    XExposeEvent xexpose;
    XGraphicsExposeEvent xgraphicsexpose;
    XNoExposeEvent xnoexpose;
    XVisibilityEvent xvisibility;
    XCreateWindowEvent xcreatewindow;
    XDestroyWindowEvent xdestroywindow;
    XUnmapEvent xunmap;
    XMapEvent xmap;
    XMapRequestEvent xmaprequest;
    XReparentEvent xreparent;
    XConfigureEvent xconfigure;
    XGravityEvent xgravity;
    XResizeRequestEvent xresizerequest;
    XConfigureRequestEvent xconfigurerequest;
    XCirculateEvent xcirculate;
    XCirculateRequestEvent xcirculaterequest;
    XPropertyEvent xproperty;
    XSelectionClearEvent xselectionclear;
    XSelectionRequestEvent xselectionrequest;
    XSelectionEvent xselection;
    XColormapEvent xcolormap;
    XClientMessageEvent xclient;
    XMappingEvent xmapping;
    XErrorEvent xerror;
    XKeymapEvent xkeymap;
    long pad[24];
} XEvent;
```

```
typedef struct {
    int type;
    unsigned long serial;
    Bool send_event;
    Display *display;
    Window window;
    Window root;
    Window subwindow;
    Time time;
    int x, y;
    int x_root, y_root;
    unsigned int state;
    unsigned int keycode;
    Bool same_screen;
} XKeyEvent;
typedef XKeyEvent XKeyPressedEvent;
typedef XKeyEvent XKeyReleasedEvent;
```

Маски событий и события

Какие события маскируем и какие сообщения ловим:

`NoEventMask`		Никакие
`KeyPressMask`	`KeyPress`	Нажата клавиша
`KeyReleaseMask`	`KeyRelease`	Отпущена клавиша
`ButtonPressMask`	`ButtonPress`	Нажата кнопка мыши
`ButtonReleaseMask`	`ButtonRelease`	Отпущена кнопка мыши
`EnterWindowMask`	`EnterNotify`	Курсор вошёл в зону окна
`LeaveWindowMask`	`LeaveNotify`	Курсор вышел из зоны окна
`PointerMotionMask`	`MotionNotify`	Курсор движется над окном
`ButtonMotionMask`	`XPointerMovedEvent`	Курсор движется, когда нажата любая кнопка
`ExposureMask`	`Expose`	Нужно перерисовать окно

…

Использование графического контекста

main_01.cpp

```
...

XSelectInput(p_display, window, ExposureMask);

GC gc = XCreateGC(p_display, window, 0, NULL);

XEvent event;
for (;;) {
  XNextEvent(p_display, &event);

  if (event.type == Expose) {
    XClearWindow(p_display, window);
    XDrawLine(p_display, window, gc, 10, 60, 180, 20);
    XFlush(p_display);
  }
}

XFreeGC(p_display, gc);

...
```

Использование цветов: стандартные цвета

main_02.cpp

```
XColor red;
Colormap cmap = DefaultColormap(p_display, screen);
Status rc = XAllocNamedColor(p_display, cmap, "red", &red, &red);
if (!rc) {
  printf("'red' color allocation failure.\n");
  return EXIT_FAILURE;
}

...

XSetForeground(p_display, gc, red.pixel);
XDrawLine(p_display, window, gc, 10, 60, 180, 20);
```

Использование цветов: собственные цвета

```
XColor col;
col.red = 65535;
col.green = 32767;
col.blue = 0;

Colormap cmap = DefaultColormap(p_display, screen);

Status rc = XAllocColor(p_display, cmap, &col);
if (!rc) {
  printf("Color allocation failure.\n");
  return EXIT_FAILURE;
}

...

XSetForeground(p_display, gc, col.pixel);
XDrawLine(p_display, window, gc, 10, 60, 180, 20);
```

Простое приложение GTK

gtk_00.cpp

```
#include <gtk/gtk.h>

int main(int argc, char *argv[]) {
  gtk_init(&argc, &argv);

  GtkWidget *p_window = gtk_window_new(GTK_WINDOW_TOPLEVEL);

  g_signal_connect( p_window,
                    "destroy",
                    G_CALLBACK(gtk_main_quit),
                    NULL);

  gtk_widget_show(p_window);

  gtk_main();

  return 0;
}
```

Компиляция приложения GTK

```
g++ -pthread -I/usr/include/gtk-2.0 -I/usr/lib64/gtk-2.0/include
-I/usr/include/atk-1.0 -I/usr/include/cairo
-I/usr/include/gdk-pixbuf-2.0 -I/usr/include/pango-1.0
-I/usr/include/glib-2.0 -I/usr/lib64/glib-2.0/include
-I/usr/include/harfbuzz -I/usr/include/freetype2
-I/usr/include/pixman-1 -I/usr/include/libpng15
-I/usr/include/libdrm  -lgtk-x11-2.0 -lgdk-x11-2.0 -latk-1.0
-lgio-2.0 -lpangoft2-1.0 -lpangocairo-1.0 -lgdk_pixbuf-2.0 -lcairo
-lpango-1.0 -lfontconfig -lgobject-2.0 -lglib-2.0
-lfreetype   -o gtk_00 ./gtk_00.cpp
```

```
pkg-config --libs gtk+-2.0
```

```
g++ `pkg-config --cflags --libs gtk+-2.0` -o gtk_00 ./gtk_00.cpp
```

Обработка событий

```c
#include <gtk/gtk.h>

void on_destroy() {
  gtk_main_quit();
}

int main(int argc, char *argv[]) {
  gtk_init(&argc, &argv);

  GtkWidget *p_window = gtk_window_new(GTK_WINDOW_TOPLEVEL);

  g_signal_connect( p_window, "destroy",
                    G_CALLBACK(on_destroy), NULL);

  gtk_widget_show(p_window);

  gtk_main();

  return 0;
}
```

Рисование с cairo (1)

gtk_02.cpp

```
g_signal_connect(p_window, "destroy",
                 G_CALLBACK(gtk_main_quit), NULL);

GtkWidget *p_draw = gtk_drawing_area_new();

gtk_container_add(GTK_CONTAINER(p_window), p_draw);

g_signal_connect(G_OBJECT(p_draw),
                 "expose",
                 G_CALLBACK(on_draw),
                 NULL);
```

Рисование с cairo (2)

```
gboolean on_draw(GtkWidget *widget, GdkEventExpose *event, gpointer data
  cairo_t *c = gdk_cairo_create(gtk_widget_get_window(widget));
  cairo_move_to(c, 30, 30);
  cairo_show_text(c, "Text");

  cairo_set_line_width(c, 1.0);
  cairo_set_source_rgb(c, 0, 0, 0);
  cairo_rectangle(c, 10, 10, 100, 100);
  cairo_stroke(c);

  cairo_destroy(c);

  g_print("draw\n");
  return TRUE;
}
```

Создание меню в GTK

gtk_03.cpp

```
vbox = gtk_vbox_new(FALSE, 0);
gtk_container_add(GTK_CONTAINER(p_window), vbox);

GtkWidget *menubar = gtk_menu_bar_new();
GtkWidget *menu_file = gtk_menu_new();

GtkWidget *mi_file = gtk_menu_item_new_with_label("File");
GtkWidget *mi_quit = gtk_menu_item_new_with_label("Quit");

gtk_menu_item_set_submenu(GTK_MENU_ITEM(mi_file), menu_file);
gtk_menu_shell_append(GTK_MENU_SHELL(menu_file), mi_quit);
gtk_menu_shell_append(GTK_MENU_SHELL(menubar), mi_file);

gtk_box_pack_start(GTK_BOX(vbox), menubar, FALSE, FALSE, 0);
```

Задание на дом

С использованием библиотеки Gtk+ разработать программу, которая будет отображать на окне фигуру – прямоугольник.

У окна программы должно быть меню, в котором есть 3 пункта со следующими элементами.

1. Меню «File»
   - Элемент «Exit» - выход из программы
2. Меню «Brush»
   - Элемент «Red» - прямоугольник рисуется красным цветом заливки
   - Элемент «Yellow» - прямоугольник рисуется жёлтым цветом заливки
   - Элемент «Brown» - прямоугольник рисуется коричневым цветом заливки
3. Меню «Pen»
   - Элемент «Green» - контур прямоугольника рисуется зелёным цветом
   - Элемент «Blue» - контур прямоугольника рисуется синим цветом
   - Элемент «Pink» - контур прямоугольника рисуется розовым цветом