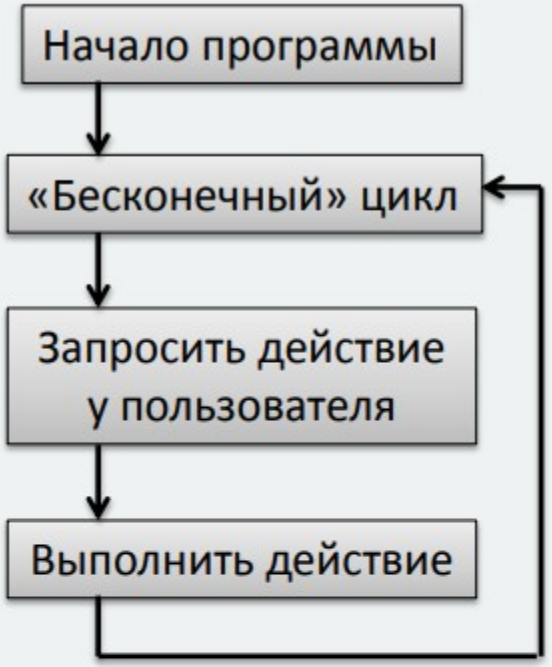
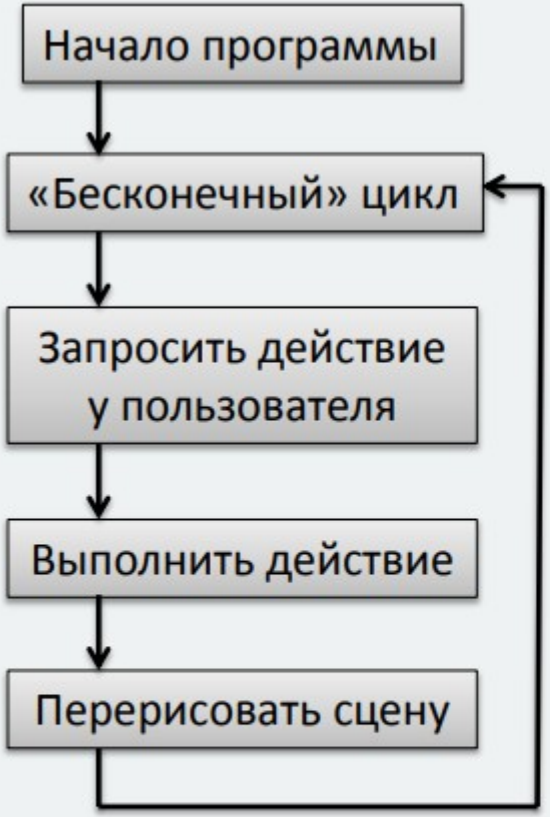
Основы программирования приложений с использованием Windows API.

Архитектура консольных программ

```c
#include <stdlib.h>
#include <stdio.h>
#include <time.h>

int main(int, char **) {
  srand(time(NULL));
  int r_val = rand() % 10;
  int user_answer = 0;
  while (1) {
    printf("Enter your guess : ");
    scanf("%d", &user_answer);
    if (user_answer == r_val)
      break;
    printf("Try one again\n");
  }
  printf("Congratulations!");
  return 0;
}
```



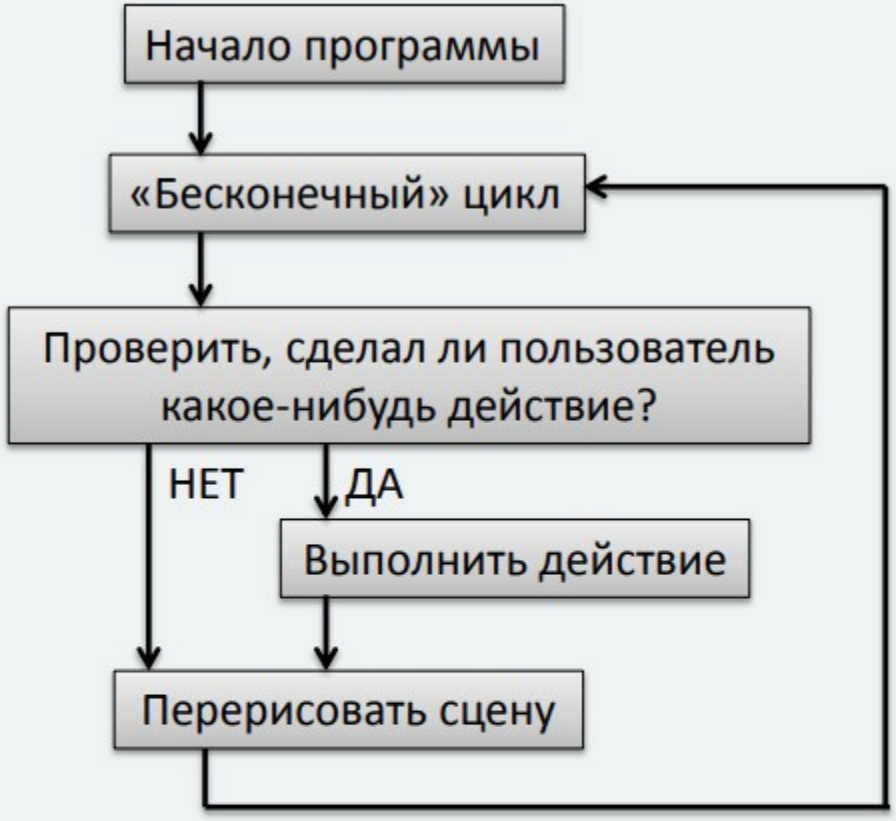
Как была реализована анимация?

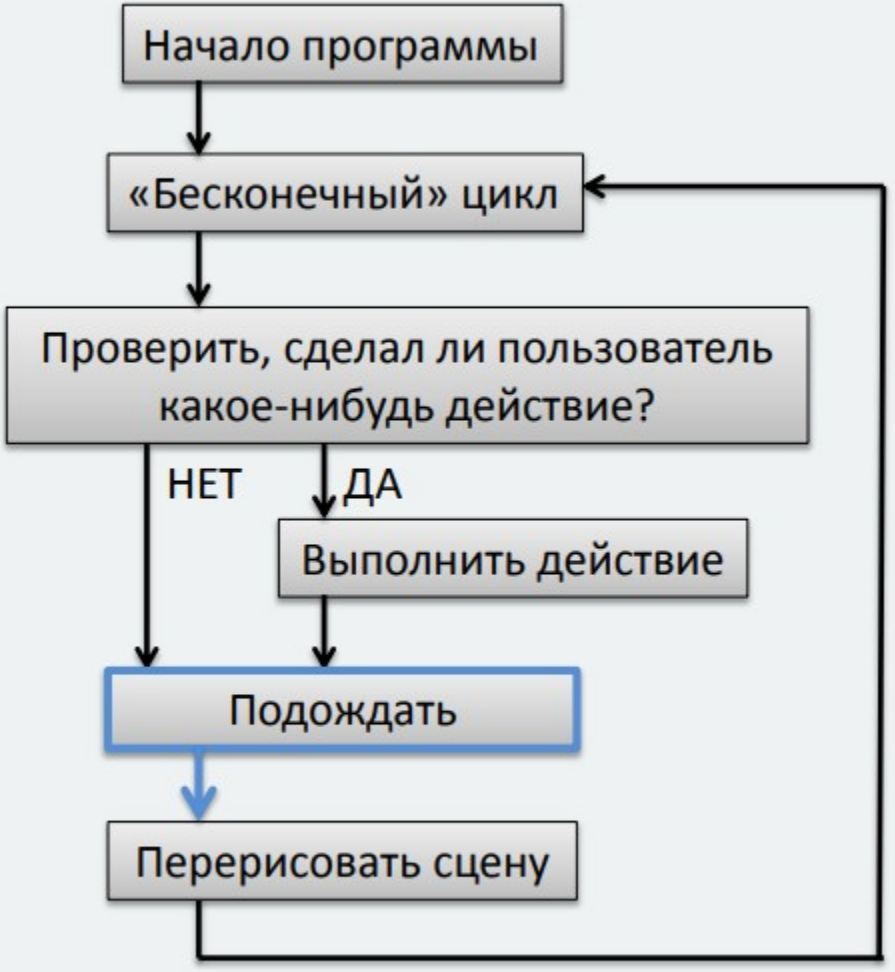


Подождать нажатия клавиш:
`getch()`

Проверить состояние клавиатуры:
`kbhit() + getch()`

Компенсация роста скорости работы ЭВМ





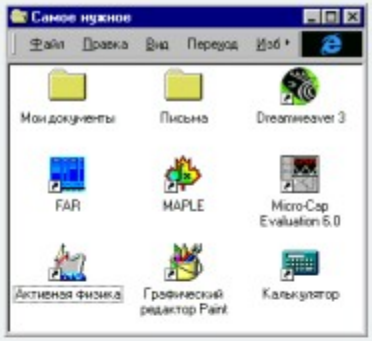

```
A = GetTickCount();
B = GetTickCount();
Δ = B – A;
if ( Δ < 1.0/FPS )
   sleep(1.0/FPS – Δ );
```

Сложность «оконных» программ – что нужно отслеживать

Система

Окно:
- активировано/деактивировано
- нажали кнопку NC области
- действие мыши в NC области
- надо рисовать

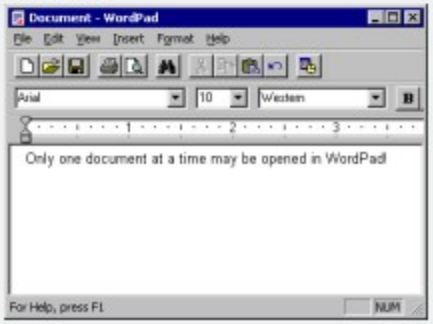
…



Программа

Один из «компонентов» - текст:
- выделили область
- скопировали/вырезали/вставили текст

…



События от таймера.
Межпроцессное взаимодействие.

…

Пользователь

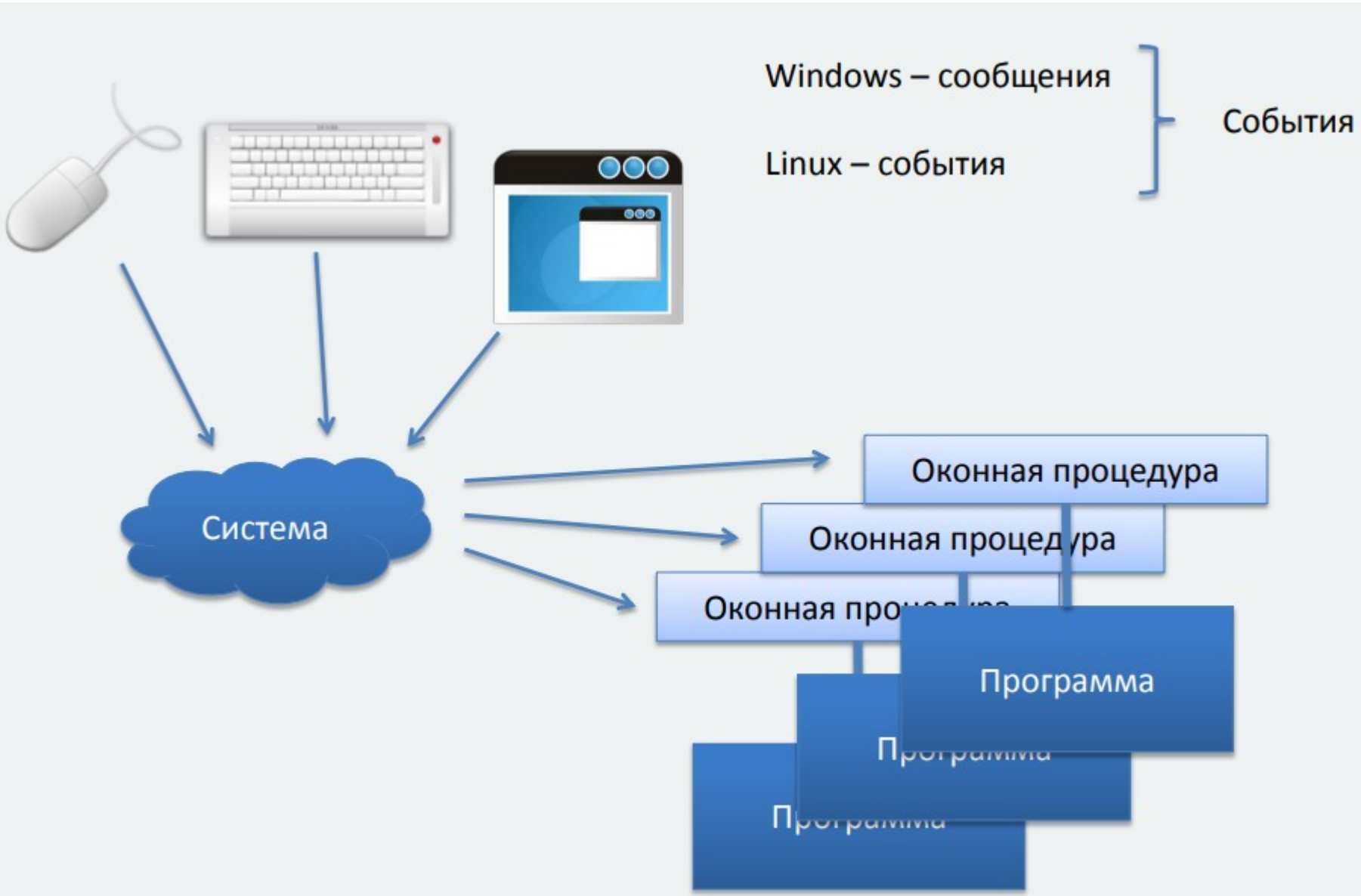
Данные от клавиатуры:
- клавиша нажата
- клавиша отпущена
- нажали клавишу

…

Данные от мыши:
- нажали кнопку
- отпустили кнопку
- кликнули 1 или 2 раза
- крутим скролл
- двигаем мышку

…

Архитектура ПО на основе обработки событий



Простейшая программа под Windows

```
#include <windows.h>
#include <string.h>

#define szWindowClass "MyWindow"
#define szTitle "A Simple Window"

int __stdcall WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow) {
  WNDCLASSEX wcex;

  wcex.cbSize = sizeof(WNDCLASSEX);
  wcex.style = CS_HREDRAW | CS_VREDRAW;
  wcex.lpfnWndProc = WndProc;
  wcex.cbClsExtra = 0;
  wcex.cbWndExtra = 0;
  wcex.hInstance = hInstance;
  wcex.hIcon = LoadIcon(hInstance, MAKEINTRESOURCE(IDI_APPLICATION));
  wcex.hCursor = LoadCursor(NULL, IDC_ARROW);
  wcex.hbrBackground = (HBRUSH)(COLOR_WINDOW + 1);
  wcex.lpszMenuName = NULL;
  wcex.lpszClassName = szWindowClass;
  wcex.hIconSm = LoadIcon(wcex.hInstance, MAKEINTRESOURCE(IDI_APPLICATION));

  if (!RegisterClassEx(&wcex)) {
    MessageBox(NULL, "Can't register window class!", "Win32 API Test", NULL);
    return 1;
  }

  HWND hWnd = CreateWindow(szWindowClass, szTitle, WS_OVERLAPPEDWINDOW, CW_USEDEFAULT,
    CW_USEDEFAULT, 500, 400, NULL, NULL, hInstance, NULL);

  if (!hWnd) {
    MessageBox(NULL, "Can't create window!", "Win32 API Test", NULL);
    return 1;
  }

  ShowWindow(hWnd, SW_SHOWNORMAL);
  UpdateWindow(hWnd);

  MSG msg;
  while (GetMessage(&msg, NULL, 0, 0)) {
    TranslateMessage(&msg);
    DispatchMessage(&msg);
  }

  return msg.wParam;
}
```

Инициализация параметров создаваемого окна

Регистрация окна с нашими параметрами в системе

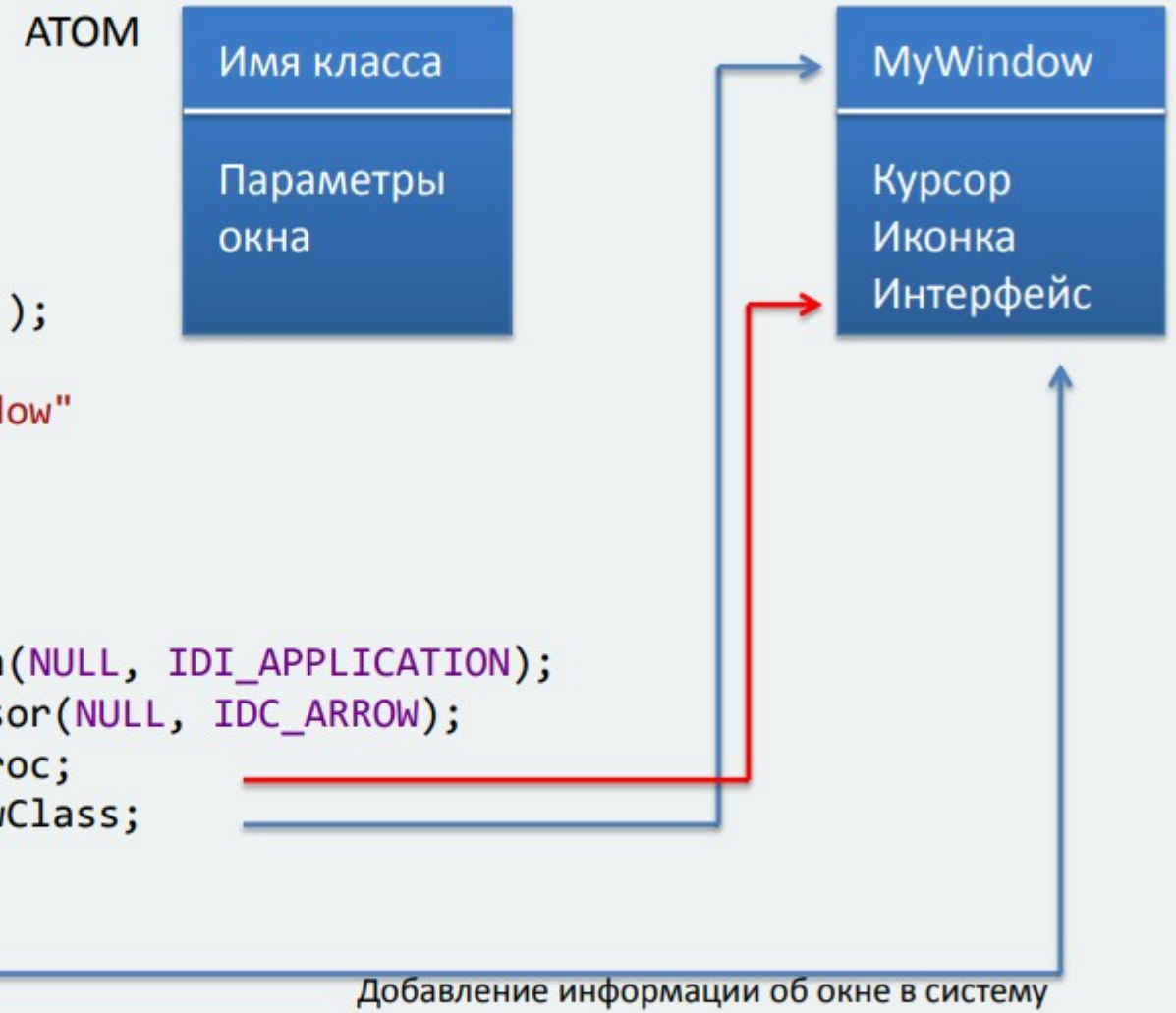
Создание экземпляра окна

Показать окно

Запуск обработчика событий

+ оконная процедура

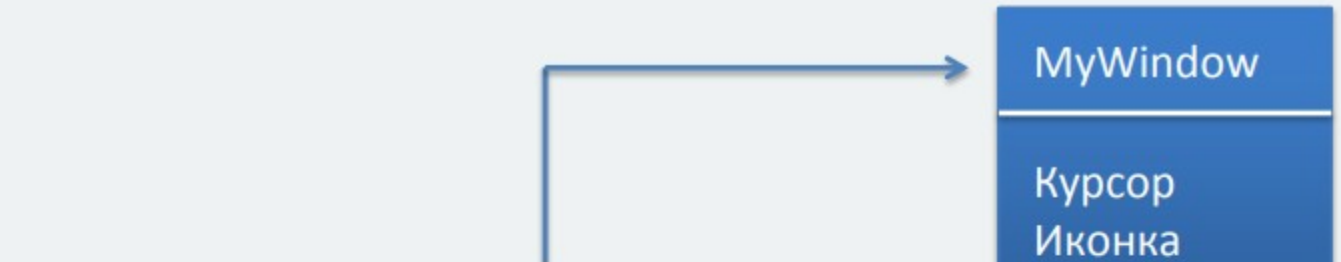
Регистрация окна в системе

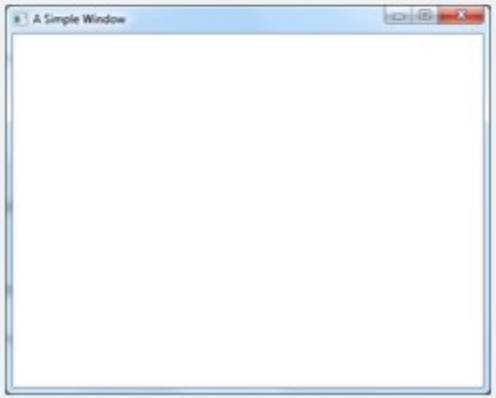


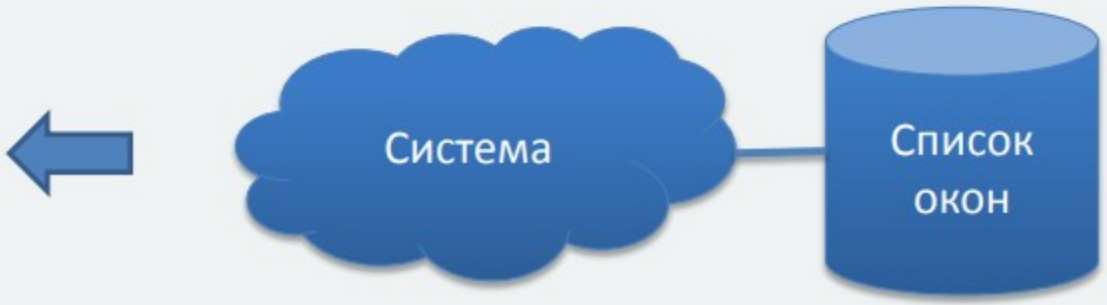

```
long __stdcall WindowProc(...);

#define szWindowClass "MyWindow"

int WinMain(...) {
  WNDCLASSEX wc;
  ...
  wc.hIcon         = LoadIcon(NULL, IDI_APPLICATION);
  wc.hCursor       = LoadCursor(NULL, IDC_ARROW);
  wc.lpfnWndProc   = WindowProc;
  wc.lpszClassName = szWindowClass;
  ...

  RegisterClassEx(&wc);

  ...
}
```

Создание экземпляра окна

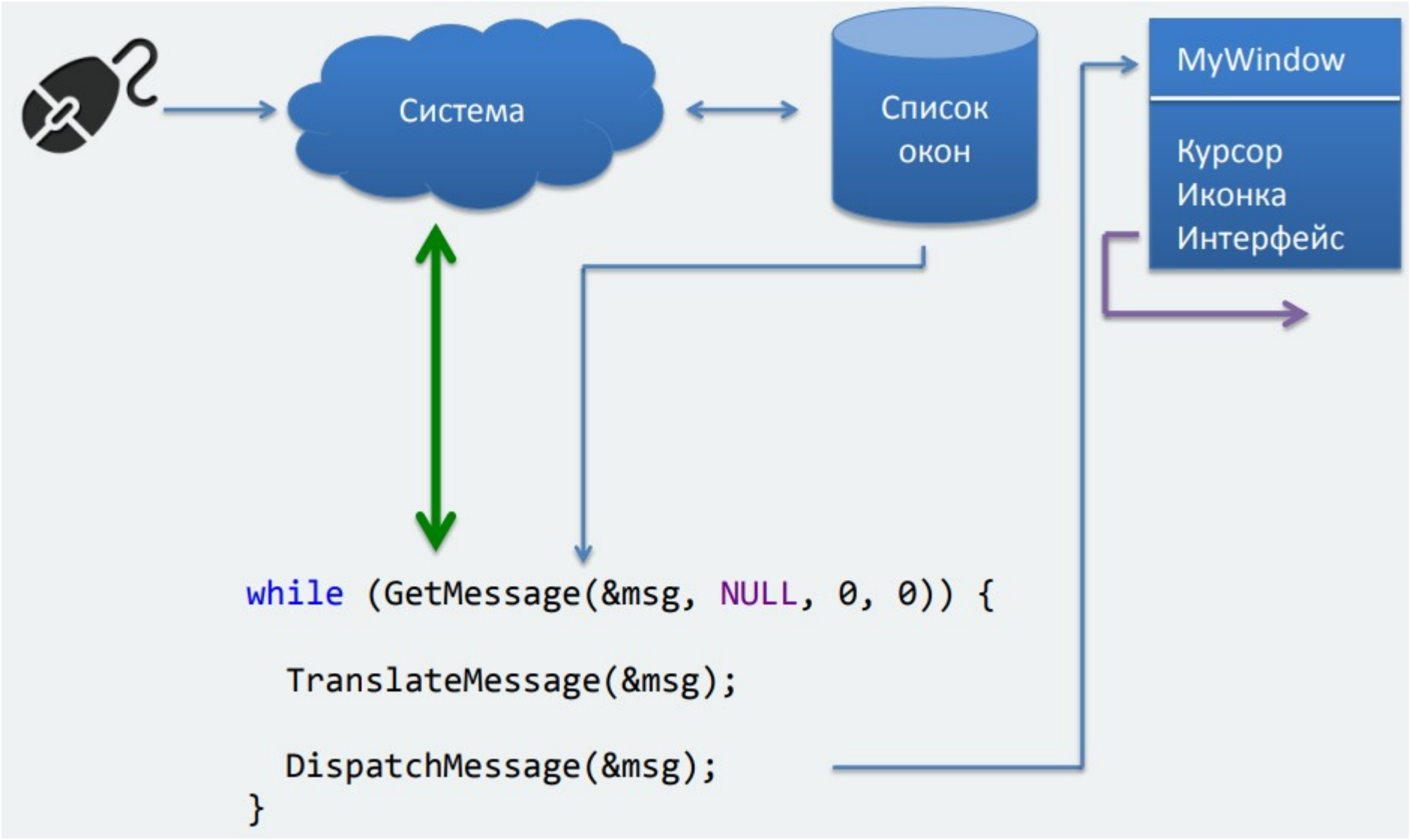
```
HWND hWnd = CreateWindow(szWindowClass,
                         "A Simple Window", ...);

if (!hWnd) {
  MessageBox(NULL, "Can't create window!", ...);
  return 1;
}
```







Ожидание сообщений от операционной системы



Информирование окна системой о происходящем с ним (1)

MyWindow

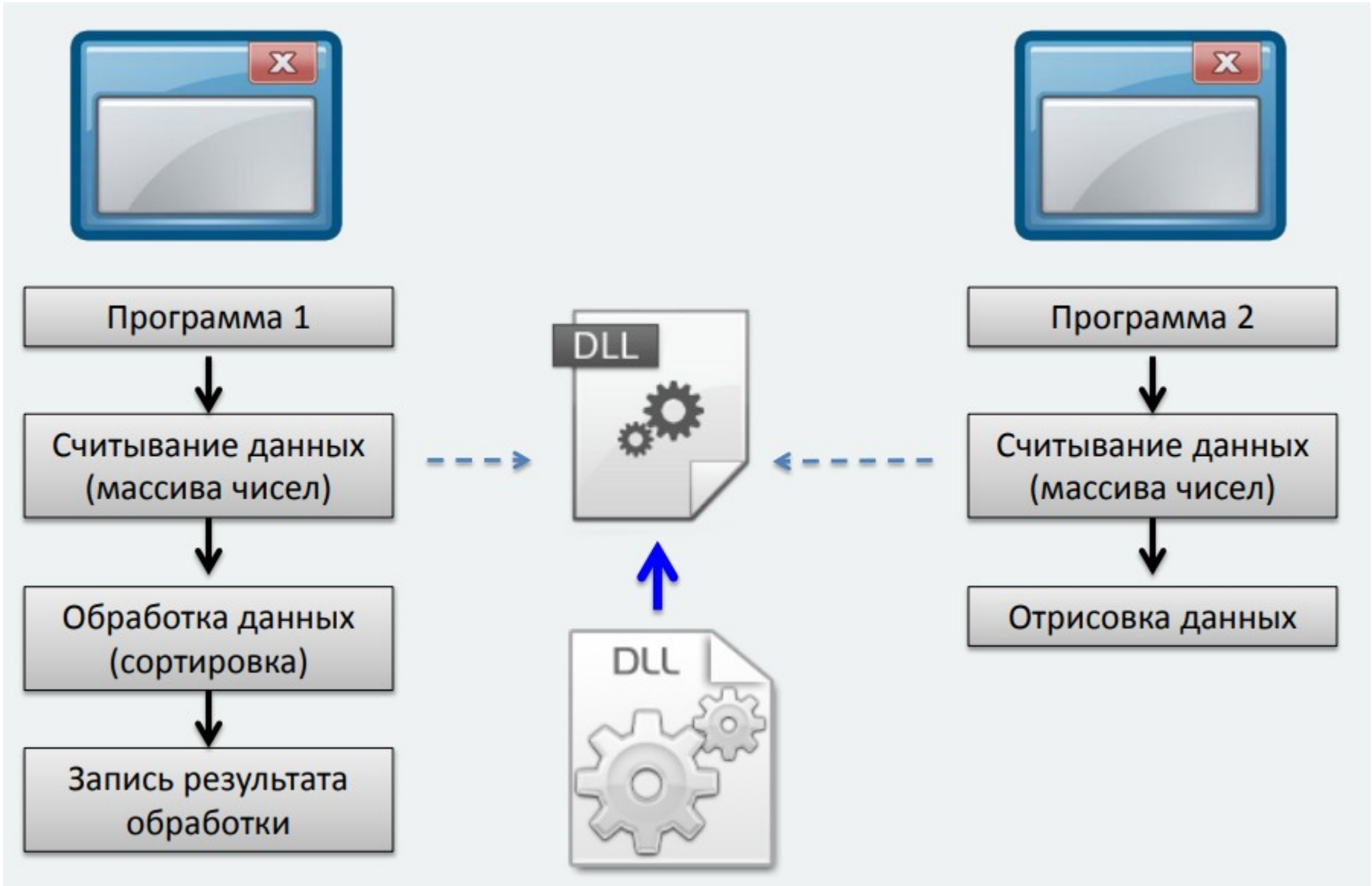
Курсор
Иконка
Интерфейс

```
long __stdcall WndProcedure(HWND   hWnd,
                            UINT   Msg,
                            WPARAM wParam,
                            LPARAM lParam) {
  switch (Msg) {
    case WM_DESTROY: PostQuitMessage(WM_QUIT);
      break;

    case WM_PAINT: ...

    case WM_LBUTTONDOWN: ...

    case WM_SIZE: ...

    default: return DefWindowProc(hWnd, Msg, wParam, lParam);
  }
  return 0;
}
```

Примеры WM_ сообщений (WinUser.h)

```
/*
 * Window Messages
 */

#define WM_NULL                         0x0000
#define WM_CREATE                       0x0001
#define WM_DESTROY                      0x0002
#define WM_MOVE                         0x0003
#define WM_SIZE                         0x0005

...

#define WM_NCCREATE                     0x0081
#define WM_NCDESTROY                    0x0082
#define WM_NCCALCSIZE                   0x0083
#define WM_NCHITTEST                    0x0084
#define WM_NCPAINT                      0x0085

...

#define WM_PAINT                        0x000F
#define WM_CLOSE                        0x0010
#define WM_QUERYENDSESSION              0x0011
```

Динамически загружаемые библиотеки



WinAPI

```
#include <windows.h>
#include <stdlib.h>
#include <string.h>
#include <tchar.h>

static TCHAR szWindowClass[] = _T("win32app");
static TCHAR szTitle[] = _T("A Simple Window");

  HINSTANCE hInst;

LRESULT CALLBACK WndProc(HWND, UINT, WPARAM, LPARAM);

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow) {
  WNDCLASSEX wcex;

  wcex.cbSize = sizeof(WNDCLASSEX);
  wcex.style = CS_HREDRAW | CS_VREDRAW;
  wcex.lpfnWndProc = WndProc;
  wcex.cbClsExtra = 0;
  wcex.cbWndExtra = 0;
  wcex.hInstance = hInstance;
  wcex.hIcon = LoadIcon(hInstance, MAKEINTRESOURCE(IDI_APPLICATION));
  wcex.hCursor = LoadCursor(NULL, IDC_ARROW);
  wcex.hbrBackground = (HBRUSH)(COLOR_WINDOW + 1);
  wcex.lpszMenuName = NULL;
  wcex.lpszClassName = szWindowClass;
  wcex.hIconSm = LoadIcon(wcex.hInstance, MAKEINTRESOURCE(IDI_APPLICATION));

  if (!RegisterClassEx(&wcex)) {
    MessageBox(NULL, _T("Call to RegisterClassEx failed!"), _T("Win32 Guided Tour"), NULL);
    return 1;
  }

  hInst = hInstance;
  HWND hWnd = CreateWindow(
    szWindowClass,
    szTitle,
    WS_OVERLAPPEDWINDOW,
    CW_USEDEFAULT, CW_USEDEFAULT,
    500, 100,
    NULL,
    NULL,
    hInstance,
    NULL
  );

  if (!hWnd) {
    MessageBox(NULL, _T("Call to CreateWindow failed!"), _T("Win32 Guided Tour"), NULL);
    return 1;
  }

  ShowWindow(hWnd, nCmdShow);
  UpdateWindow(hWnd);

  MSG msg;
  while (GetMessage(&msg, NULL, 0, 0)) {
    TranslateMessage(&msg);
    DispatchMessage(&msg);
  }

  return (int)msg.wParam;
}
```

```
long __stdcall WndProc(HWND hWnd, UINT message, WPARAM wParam, LPARAM lParam) {
  PAINTSTRUCT ps;
  HDC hdc;
  TCHAR greeting[] = _T("Hello, World!");

  switch (message)
  {
  case WM_PAINT:
    hdc = BeginPaint(hWnd, &ps);
    TextOut(hdc, 5, 5, greeting, strlen(greeting));
    EndPaint(hWnd, &ps);
    break;
  case WM_DESTROY:
    PostQuitMessage(0);
    break;
  default:
    return DefWindowProc(hWnd, message, wParam, lParam);
    break;
  }

  return 0;
}
```

API Xlib под X Window System (1)

```c
#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;
}
```

API Xlib под X Window System (2)

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

int main(int, char *[]) {

  ...

  XMapWindow(p_display, window);

  XSelectInput(p_display, window, NoEventMask);

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

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

  ...

}
```

Кросплатформенные ОО библиоти: Gtk

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

void destroy() {
  gtk_main_quit();
}

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

  gtk_init(&argc, &argv);

  GtkWidget *window = gtk_window_new(GTK_WINDOW_TOPLEVEL);

  gtk_signal_connect(GTK_OBJECT(window),
                     "destroy",
                     GTK_SIGNAL_FUNC(destroy),
                     NULL);

  gtk_widget_show(window);

  gtk_main();

  return 0;
}
```

GTK

Кросплатформенные ОО библиоти:

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

class Frame : public wxFrame {
public:
  Frame(const wxString& title);
};

Frame::Frame(const wxString& title)
  : wxFrame(NULL, wxID_ANY, title, wxDefaultPosition,
    wxSize(250, 150)) {

  Centre();
}

class SimpleApp : public wxApp {
public:
  virtual bool OnInit();
};

IMPLEMENT_APP(MyApp)

bool MyApp::OnInit() {
  SimpleApp *p_app = new SimpleApp(wxT("Simple"));
  p_app->Show(true);

  return true;
}
```

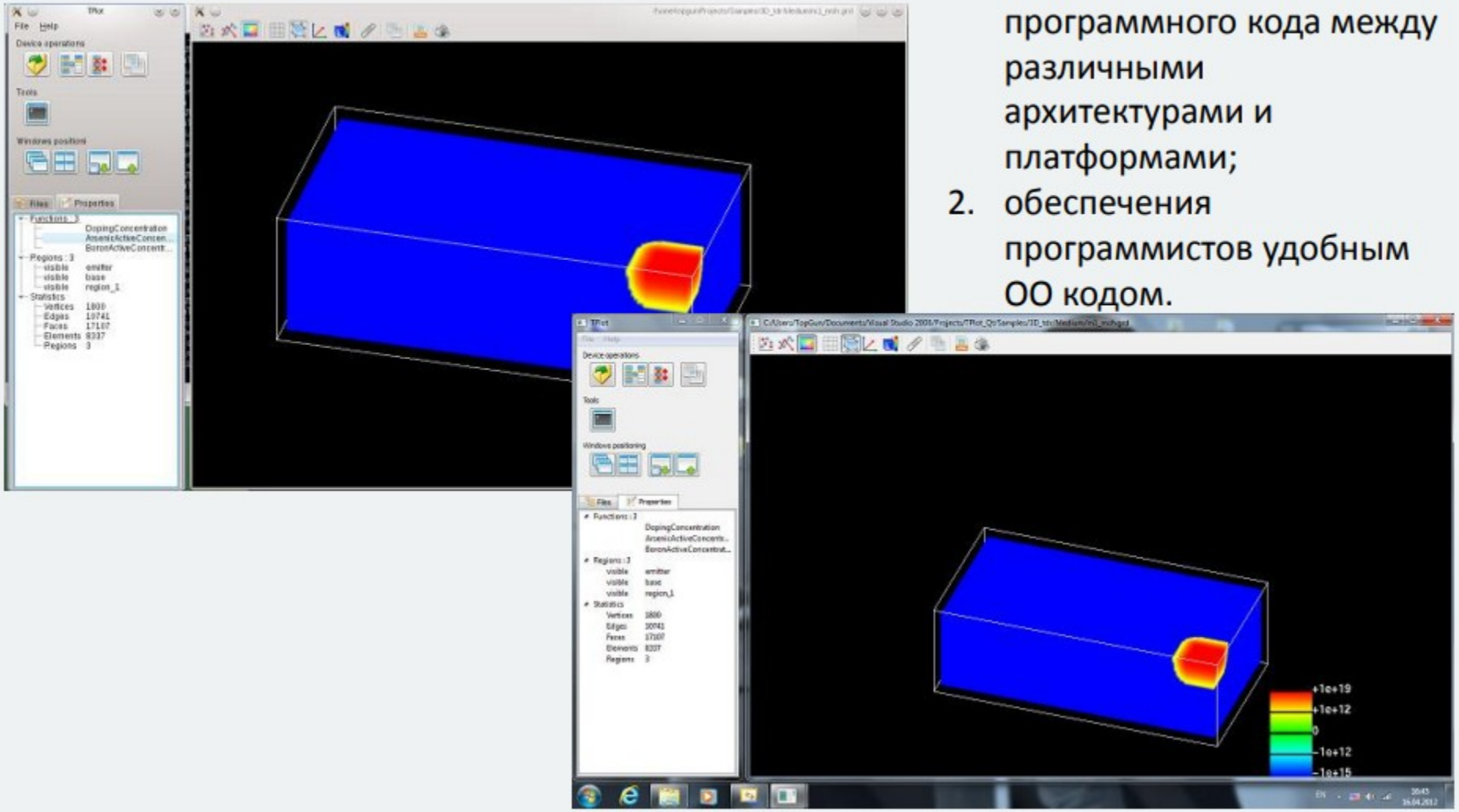
wxWidgets

Кросплатформенные ОО библиоти: Qt

```
#include <QMainWindow>
#include <QApplication>

class MyWindow : public QMainWindow {
public:
  MyWindow(QWidget *parent = 0);
};

MyWindow::MyWindow(QWidget *parent) : QMainWindow(parent) {
  setWindowTitle("Simple");
  resize(400, 200);
}

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

  QApplication app(argc, argv);

  MyWindow *win = new MyWindow;
  win->show();

  return app.exec();
}
```

Qt

Пример интерфейса программы, использующей Qt



Решают задачи:

1. переносимости программного кода между различными архитектурами и платформами;
2. обеспечения программистов удобным ОО кодом.