Санкт-Петербургский политехнический университет Петра Великого
Институт прикладной математики и механики
Высшая школа прикладной математики и вычислительной физики

КУРСОВАЯ РАБОТА

Решение задачи task4

по дисциплине «Алгоритмы и алгоритмические языки»

Выполнил
студент гр.3630102/00002 К.И. Желудев

Преподаватель А.Б. Григорьев

Санкт-Петербург

2020

Введение

Условие задачи следующее.

Поле шахматной доски определяется парой натуральных чисел, каждое из которых не превосходит восьми: первое число - номер вертикали (при счете слева направо), второе - номер горизонтали (при счете снизу вверх). Даны натуральные числа k, l, m, n, каждое не превосходит восьми. Требуется: Выяснить, можно ли с поля (k,l) одним ходом ферзя попасть на поле (m,n). Если нет, указать, как это можно сделать за 2 хода (указать поле, на которое придется первый ход). Решение отобразить графически.

Координаты фигур считываются по клику мышки на соответствующей клетке доски.

Ход решения

Создадим проект в среде разработки Code::Blocks opengl проект. По умолчанию он выводит вращающийся разноцветный треугольник на черном фоне. На рисунке 1 он показан.

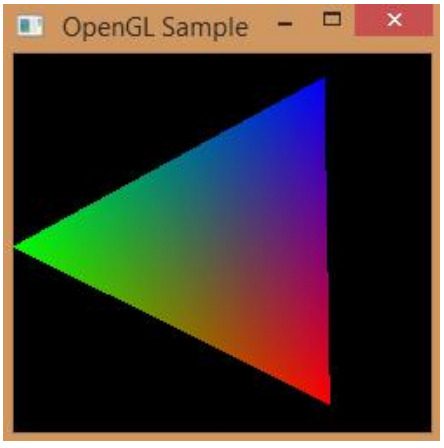

Рисунок 1 OpenGl Sample

Увеличим выводящее окно до 800*800 пикселей, изменив аргументы у функции CreateWindowEx

```
CreateWindowEx(0,
                    "GLSample",
                    "OpenGL Sample",
                    WS_OVERLAPPEDWINDOW,
                    CW_USEDEFAULT,
```

```
CW_USEDEFAULT,
800,
800,
NULL,
NULL,
hInstance,
NULL);
```

Рассмотрим OpenGL animation код, он отмечен соответствующим комментарием. Оставляем черный фон каким он и был. Ставим толщину всех отрезков за 10 функцией glLineWidth(10). Далее glLoadIdentity() загружает единичную матрицу преобразований, glTranslatef(-1, -1, 0) и glScalef(0.25, 0.25, 1) переводит центр системы координат по оси x и y на одну единицу влево и уменьшаем масштаб в 4 раза (при этом ось z не трогаем, поскольку она не нужна в задаче) для того, чтобы координаты окна были от 0 до 8. Рисуем доску процедурой DrawDesk().

DrawDesk, рисует шахматную доску на всю окно. Поскольку фон весь черный, то достаточно нарисовать только белые клетки.

С помощью двух вложенных циклов for нам удаётся нарисовать 16 белых клеток. Квадраты (в общем случае выпуклые многоугольники) создаются так: начало и конец рисования обрамляются функциями glBegin(GL_POLYGON) и glEnd(). Между ними объявляется цвет точки glColor3f(1, 1, 1) и сама точка glVertex2f(i, j). Так как нужен белый квадрат, то и 4 точки будут белыми.

Код процедуры:

```
void DrawDesk()
{
   double i, j;
   for (i = 0; i < 8; i += 2)
      for (j = 8; j > 0; j -= 2)
      {
      glBegin(GL_POLYGON);
         glColor3f(1, 1, 1); glVertex2f(i, j);
         glColor3f(1, 1, 1); glVertex2f(i + 1, j);
         glColor3f(1, 1, 1); glVertex2f(i + 1, j - 1);
         glColor3f(1, 1, 1); glVertex2f(i, j - 1);
      glEnd();
      }
```

```
        for (i = 1; i < 8; i += 2)
            for (j = 7; j > 0; j -= 2)
        {
        glBegin(GL_POLYGON);
            glColor3f(1, 1, 1); glVertex2f(i, j);
            glColor3f(1, 1, 1); glVertex2f(i + 1, j);
            glColor3f(1, 1, 1); glVertex2f(i + 1, j - 1);
            glColor3f(1, 1, 1); glVertex2f(i, j - 1);
        glEnd();
        }
}
```

Рисунок 2 показывает, что на данном этапе уже есть шахматная доска.

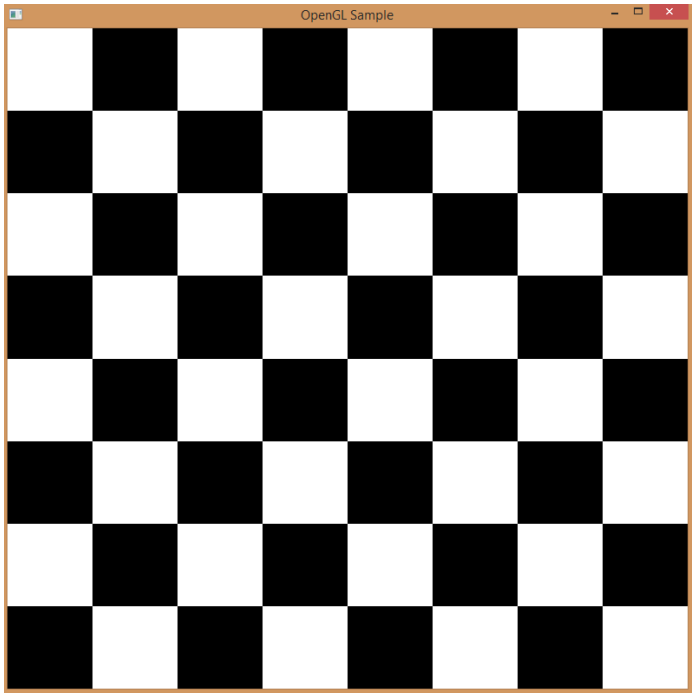

Рисунок 2 Шахматная доска

Необходимо реализовать систему считывания мышкой координат клеток. Создаем переменные int isItFirstClick = 1, isItSecondClick = 1; Поскольку координаты клика мыши и нашего окна отличны, то нужно привести первые координаты ко вторым. Для этого создадим функцию ScreeToOpenGL(HWND hwnd, int x, int y, float* ox, float* oy). В ней переменная rect структуры RECT присваивается контексту устройства hwnd структуры HWND функцией GetClientRect(hwnd, &rct). Далее координаты преобразовываются:

```
void ScreeToOpenGL (HWND hwnd, int x, int y, float* ox, float* oy)
{
   RECT rct;
   GetClientRect(hwnd, &rct);
```

```
    *ox = x / (float)rct.right * WIDTH;
    *oy = HEIGHT - y / (float)rct.bottom * HEIGHT;
}
```

HEIGHT и WEIGHT определяются как 8.

Заглянем в функцию LRESULT CALLBACK WindowProc(HWND hwnd, UINT uMsg, WPARAM wParam, LPARAM lParam) и найдем в ней case WM_LBUTTONDOWN: Если это первый клик, то с меняется флаг isItFirstClick = 0 и присваиваются переменным x1, y1 переведенные координаты клика левой кнопкой мыши.

Аналогично со вторым кликом, только меняется уже флаг isItSecondClick = 0 и присваиваются переменным x2, y2 переведенные координаты клика левой кнопкой мыши.

Итак, у нас есть приведенные координаты первой и второй точек. Далее, после рисовки доски, если оба флага равны нулю, т.е. когда уже получены все координаты, мы начинаем рисовать пути. Нарисуем начальный синий квадрат и конечный бронзовый квадрат. Алгоритм рисовки полигона был объяснен выше.

Далее, нам понадобится функция модуля числа. Она реализуется элементарно функцией abs без импорта всей библиотеки math.h

```
int abs(int x)
{
    if (x < 0)
        return -x;
    else
        return x;
}
```

Так если абсолютная разность координат x и y клеток совпадают, то ферзем можно добраться за один ход как слон, поэтому рисуем бирюзовый отрезок GL_LINES толщиной 10. На рисунке 3 показана реализация данного случая.

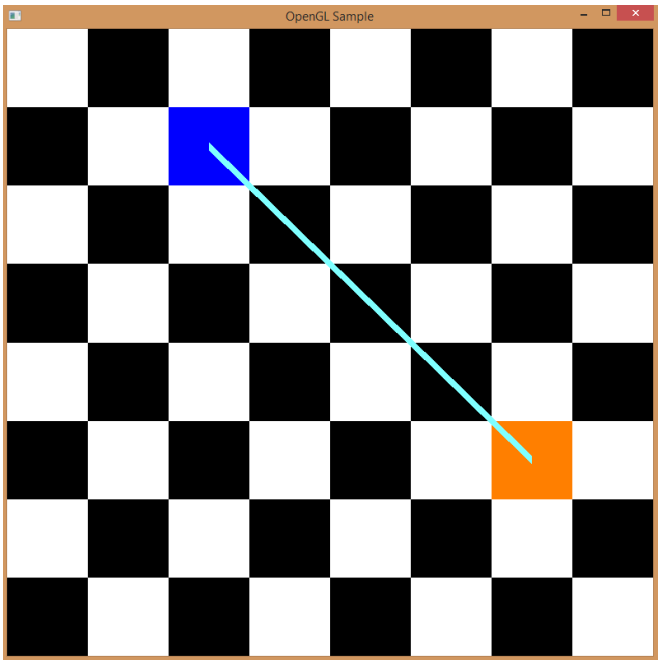

Рисунок 3 Диагональный случай

В остальных случаях ферзь ходит как ладья за один или два хода, поэтому рисуем ломаную GL_LINE_STRIP тем же бирюзовым цветом. На рисунке 4 показана реализация данного случая. На рисунке 5 показано, что выведет программа если абсолютная разность каких-то ординат или абсцисс равна нулю, т. е. когда ломаная вырождается в отрезок.

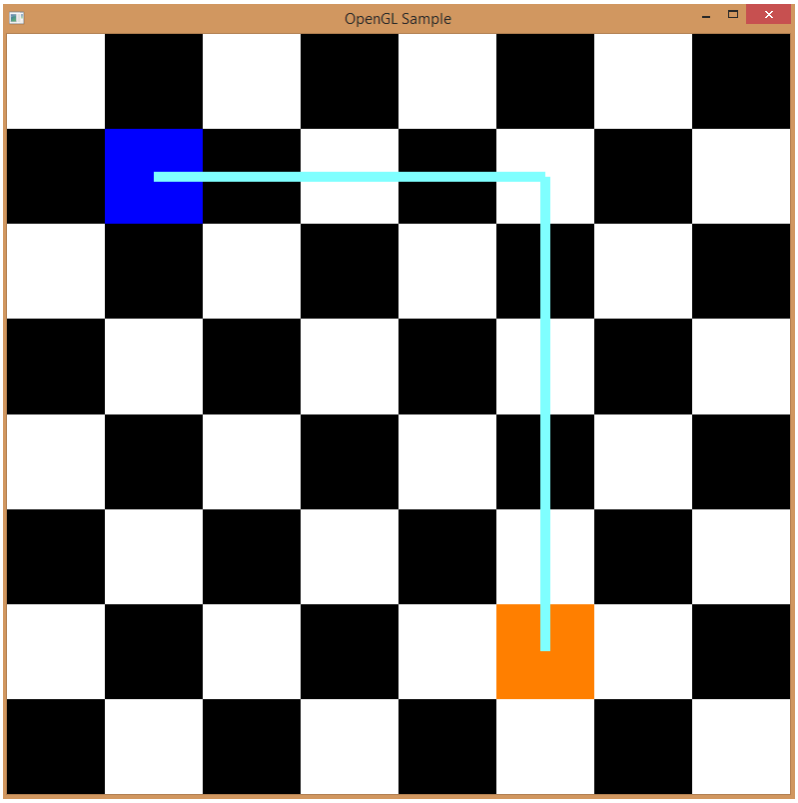

Рисунок 4 Ломаная

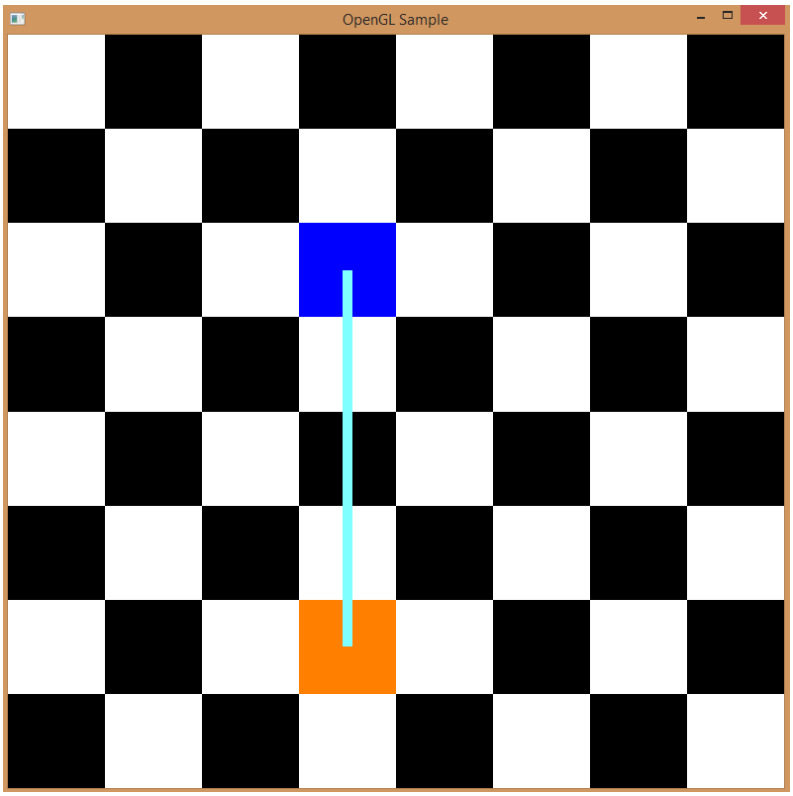

Рисунок 5 Вырожденная ломаная

```
if ((isItFirstClick == 0)&&(isItSecondClick == 0))
{
   glBegin(GL_POLYGON); // начальный квадрат - синий
   glColor3f(0, 0, 1); glVertex2f(x1, y1);
   glColor3f(0, 0, 1); glVertex2f(x1 + 1, y1);
   glColor3f(0, 0, 1); glVertex2f(x1 + 1, y1 + 1);
   glColor3f(0, 0, 1); glVertex2f(x1, y1 + 1);
   glEnd();

   glBegin(GL_POLYGON); // конечный квадрат - бронзовый
   glColor3f(1, 0.5, 0); glVertex2f(x2, y2);
   glColor3f(1, 0.5, 0); glVertex2f(x2 + 1, y2);
   glColor3f(1, 0.5, 0); glVertex2f(x2 + 1, y2 + 1);
   glColor3f(1, 0.5, 0); glVertex2f(x2, y2 + 1);
   glEnd();

   if (abs(x1 - x2) == abs(y1 - y2))
   {
   glBegin(GL_LINES);
   glColor3f(0.5, 1, 1); glVertex2f(x1 + OFF, y1 + OFF);
   glColor3f(0.5, 1, 1); glVertex2f(x2 + OFF, y2 + OFF);
   glEnd();
   }
   else
   {
   glBegin(GL_LINE_STRIP);
   glColor3f(0.5, 1, 1); glVertex2f(x1 + OFF, y1 + OFF);
   glColor3f(0.5, 1, 1); glVertex2f(x2 + OFF, y1 + OFF);
   glColor3f(0.5, 1, 1); glVertex2f(x2 + OFF, y2 + OFF);
```

```
            glEnd();
            }
        }
```

OFF = 0.5. Он отвечает за рисовку в центре клетки.

В конце концов отображается буфер, в котором мы рисовали, с помощью функции SwapBuffers(hDC).

Весь выше код находится в цикле while (!bQuit), поэтому он никогда не завершится (bQuit = 0), пока не будет нажата кнопка Esc. Это происходит потому, что есть в LRESULT CALLBACK WindowProc case VK_ESCAPE, который вызывает завершение программы.

Заключение

Данная задача познакомила меня с разработкой графического оконного приложения с помощью функций библиотеки gl, познакомила со структурами из библиотеки windows.h. Также я закрепил знания об указателях на практике.

Приложение 1. Код

```
#include <windows.h>
#include <gl/gl.h>

#define HEIGHT 8
#define WIDTH 8
#define OFF 0.5

int x1, y1, x2, y2;
int isItFirstClick = 1, isItSecondClick = 1;

LRESULT CALLBACK WindowProc(HWND, UINT, WPARAM, LPARAM);
void EnableOpenGL(HWND hwnd, HDC*, HGLRC*);
void DisableOpenGL(HWND, HDC, HGLRC);

int abs(int x)
{
    if (x < 0)
        return -x;
    else
        return x;
}

void DrawDesk()
```

```
{
   double i, j;
   for (i = 0; i < 8; i += 2)
      for (j = 8; j > 0; j -= 2)
      {
      glBegin(GL_POLYGON);
         glColor3f(1, 1, 1); glVertex2f(i, j);
         glColor3f(1, 1, 1); glVertex2f(i + 1, j);
         glColor3f(1, 1, 1); glVertex2f(i + 1, j - 1);
         glColor3f(1, 1, 1); glVertex2f(i, j - 1);
      glEnd();
      }
      for (i = 1; i < 8; i += 2)
         for (j = 7; j > 0; j -= 2)
      {
      glBegin(GL_POLYGON);
         glColor3f(1, 1, 1); glVertex2f(i, j);
         glColor3f(1, 1, 1); glVertex2f(i + 1, j);
         glColor3f(1, 1, 1); glVertex2f(i + 1, j - 1);
         glColor3f(1, 1, 1); glVertex2f(i, j - 1);
      glEnd();
      }
}

void ScreeToOpenGL (HWND hwnd, int x, int y, float* ox, float* oy)
{
   RECT rct;
   GetClientRect(hwnd, &rct);
   *ox = x / (float)rct.right * WIDTH;
   *oy = HEIGHT - y / (float)rct.bottom * HEIGHT;
}

int WINAPI WinMain(HINSTANCE hInstance,
                   HINSTANCE hPrevInstance,
                   LPSTR lpCmdLine,
                   int nCmdShow)
{
   WNDCLASSEX wcex;
   HWND hwnd;
   HDC hDC;
   HGLRC hRC;
   MSG msg;
   BOOL bQuit = FALSE;

   /* register window class */
   wcex.cbSize = sizeof(WNDCLASSEX);
   wcex.style = CS_OWNDC;
   wcex.lpfnWndProc = WindowProc;
   wcex.cbClsExtra = 0;
   wcex.cbWndExtra = 0;
   wcex.hInstance = hInstance;
   wcex.hIcon = LoadIcon(NULL, IDI_APPLICATION);
```

```
wcex.hCursor = LoadCursor(NULL, IDC_ARROW);
wcex.hbrBackground = (HBRUSH)GetStockObject(BLACK_BRUSH);
wcex.lpszMenuName = NULL;
wcex.lpszClassName = "GLSample";
wcex.hIconSm = LoadIcon(NULL, IDI_APPLICATION);;


if (!RegisterClassEx(&wcex))
   return 0;

/* create main window */
hwnd = CreateWindowEx(0,
                      "GLSample",
                      "OpenGL Sample",
                      WS_OVERLAPPEDWINDOW,
                      CW_USEDEFAULT,
                      CW_USEDEFAULT,
                      800,
                      800,
                      NULL,
                      NULL,
                      hInstance,
                      NULL);

ShowWindow(hwnd, nCmdShow);

/* enable OpenGL for the window */
EnableOpenGL(hwnd, &hDC, &hRC);

/* program main loop */
while (!bQuit)
{
   /* check for messages */
   if (PeekMessage(&msg, NULL, 0, 0, PM_REMOVE))
   {
      /* handle or dispatch messages */
      if (msg.message == WM_QUIT)
      {
         bQuit = TRUE;
      }
      else
      {
         TranslateMessage(&msg);
         DispatchMessage(&msg);
      }
   }
   else
   {
      /* OpenGL animation code goes here */

      glClearColor(0, 0, 0, 0);
      glLineWidth(10);
```

```
        glClear(GL_COLOR_BUFFER_BIT);

        glLoadIdentity();
        glTranslatef(-1, -1, 0);
        glScalef(0.25, 0.25, 1);

        DrawDesk();

        if ((isItFirstClick == 0)&&(isItSecondClick == 0))
        {
           glBegin(GL_POLYGON); // начальный квадрат - синий
           glColor3f(0, 0, 1); glVertex2f(x1, y1);
           glColor3f(0, 0, 1); glVertex2f(x1 + 1, y1);
           glColor3f(0, 0, 1); glVertex2f(x1 + 1, y1 + 1);
           glColor3f(0, 0, 1); glVertex2f(x1, y1 + 1);
           glEnd();

           glBegin(GL_POLYGON); // конечный квадрат - бронзовый
           glColor3f(1, 0.5, 0); glVertex2f(x2, y2);
           glColor3f(1, 0.5, 0); glVertex2f(x2 + 1, y2);
           glColor3f(1, 0.5, 0); glVertex2f(x2 + 1, y2 + 1);
           glColor3f(1, 0.5, 0); glVertex2f(x2, y2 + 1);
           glEnd();

           if (abs(x1 - x2) == abs(y1 - y2))
           {
           glBegin(GL_LINES);
           glColor3f(0.5, 1, 1); glVertex2f(x1 + OFF, y1 + OFF);
           glColor3f(0.5, 1, 1); glVertex2f(x2 + OFF, y2 + OFF);
           glEnd();
           }
           else
           {
           glBegin(GL_LINE_STRIP);
           glColor3f(0.5, 1, 1); glVertex2f(x1 + OFF, y1 + OFF);
           glColor3f(0.5, 1, 1); glVertex2f(x2 + OFF, y1 + OFF);
           glColor3f(0.5, 1, 1); glVertex2f(x2 + OFF, y2 + OFF);
           glEnd();
           }
        }

        SwapBuffers(hDC);
     }
}

/* shutdown OpenGL */
DisableOpenGL(hwnd, hDC, hRC);

/* destroy the window explicitly */
DestroyWindow(hwnd);

return msg.wParam;
```

```
}

LRESULT CALLBACK WindowProc(HWND hwnd, UINT uMsg, WPARAM wParam,
LPARAM lParam)
{
   switch (uMsg)
   {
      case WM_CLOSE:
         PostQuitMessage(0);
      break;

      case WM_DESTROY:
         return 0;

      case WM_LBUTTONDOWN:
      {
      if ((isItFirstClick == 1)&&(isItSecondClick == 1))
         {
            POINTFLOAT pf;
            ScreeToOpenGL(hwnd, LOWORD(lParam), HIWORD(lParam), &pf.x, &pf.y);
            isItFirstClick = 0;
            x1 = (int)pf.x;
            y1 = (int)pf.y;
         }
      else if ((isItFirstClick == 0)&&(isItSecondClick == 1))
         {
            POINTFLOAT pf;
            ScreeToOpenGL(hwnd, LOWORD(lParam), HIWORD(lParam), &pf.x, &pf.y);
            x2 = (int)pf.x;
            y2 = (int)pf.y;
            isItFirstClick = 0;
            isItSecondClick = 0;
         }
      }

      case WM_KEYDOWN:
      {
         switch (wParam)
         {
            case VK_ESCAPE:
               PostQuitMessage(0);
            break;
         }
      }
      break;

      default:
         return DefWindowProc(hwnd, uMsg, wParam, lParam);
   }

   return 0;
}
```

```
void EnableOpenGL(HWND hwnd, HDC* hDC, HGLRC* hRC)
{
   PIXELFORMATDESCRIPTOR pfd;

   int iFormat;

   /* get the device context (DC) */
   *hDC = GetDC(hwnd);

   /* set the pixel format for the DC */
   ZeroMemory(&pfd, sizeof(pfd));

   pfd.nSize = sizeof(pfd);
   pfd.nVersion = 1;
   pfd.dwFlags = PFD_DRAW_TO_WINDOW |
            PFD_SUPPORT_OPENGL | PFD_DOUBLEBUFFER;
   pfd.iPixelType = PFD_TYPE_RGBA;
   pfd.cColorBits = 24;
   pfd.cDepthBits = 16;
   pfd.iLayerType = PFD_MAIN_PLANE;

   iFormat = ChoosePixelFormat(*hDC, &pfd);

   SetPixelFormat(*hDC, iFormat, &pfd);

   /* create and enable the render context (RC) */
   *hRC = wglCreateContext(*hDC);

   wglMakeCurrent(*hDC, *hRC);
}

void DisableOpenGL (HWND hwnd, HDC hDC, HGLRC hRC)
{
   wglMakeCurrent(NULL, NULL);
   wglDeleteContext(hRC);
   ReleaseDC(hwnd, hDC);
}
```