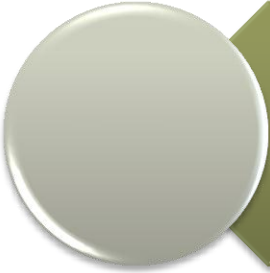


## Bài 2

# GIỚI THIỆU VISUAL C++ VÀ LẬP TRÌNH ĐỒ HỌA 3D VỚI OPENGL

# Nội dung



1 Giới thiệu Visual  
C++



2 Chương trình  
OpenGL đầu tiên



3 OpenGL: Các khái  
niệm cơ bản

# 1 GIỚI THIỆU VISUAL C++

# Giới thiệu Visual C++

- Microsoft Visual C++ (MSVC) là một Môi trường phát triển tích hợp (IDE) cho các ngôn ngữ lập trình C&C++
- Có các công cụ cho phát triển và gỡ lỗi mã nguồn C/C++
- MFC là một thư viện lập trình giao diện đồ họa cho các ứng dụng chạy trên Windows
- Visual C++ nằm trong bộ Visual Studio, trong đó bao gồm công cụ phát triển cho Visual Basic, Visual J#, Visual C#, Visual Web Developer...

# Các phiên bản của Visual C++

- Visual C++ 1.0, là phiên bản đầu tiên của Visual C++, ra đời năm 1992.
- Visual C++ 1.5, hỗ trợ thêm OLE 2.0 và ODBC cho MFC
- Visual C++ 2.0, là phiên bản đầu tiên chỉ dành riêng cho 32-bit
- Visual C++ 4.0, được thiết kế cho Windows 95, cũng như Windows NT
- Visual C++ 6.0, MFC 6.0, ra đời 1998, đã và đang được sử dụng rộng rãi cho các project lớn và nhỏ (ÊÊÊ)
- Visual C++ .NET 2002 (7.0) có một giao diện người dùng mới. Đây cũng chính là nguyên nhân tại sao Visual C++ 6.0 hiện vẫn còn được sử dụng rộng rãi
- Visual C++ .NET 2003 (7.1) là một phiên bản nâng cấp quan trọng, hỗ trợ chuẩn C++ của ISO tốt hơn rất nhiều so với các phiên bản trước đó
- Visual C++ 2008 (9.0) hỗ trợ các ứng dụng Unicode dễ dàng (nhất là cho tiếng Việt)
- Visual C++ 2010 (10.0) chính thức hỗ trợ ISO C++0x

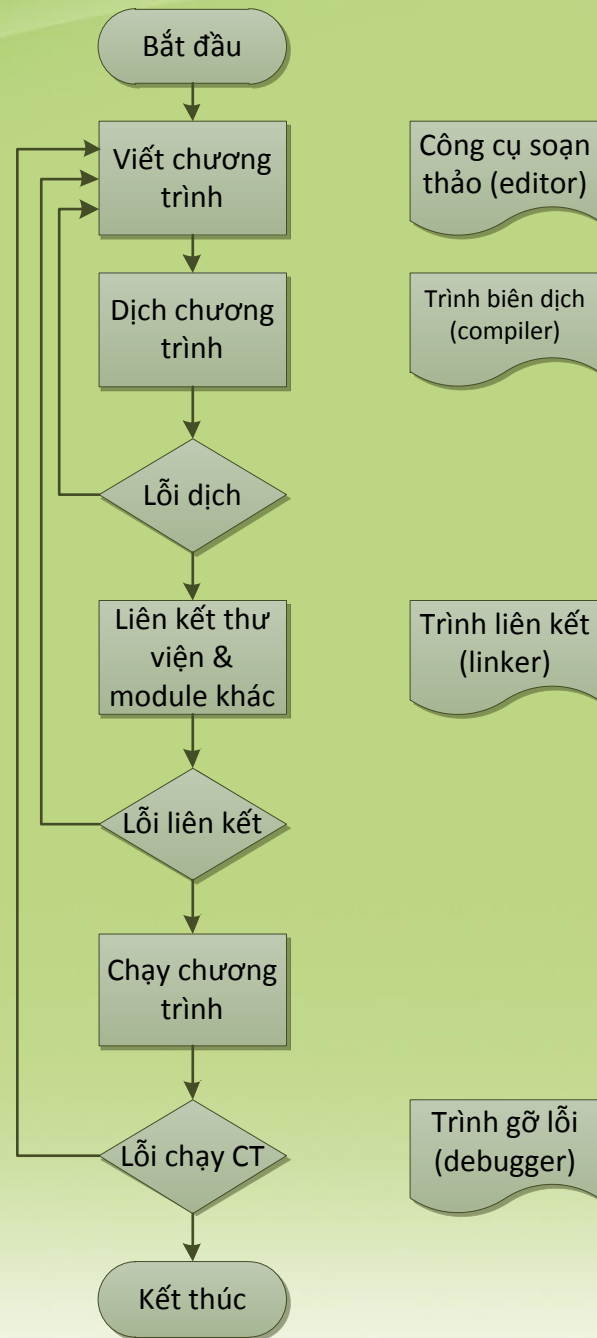
# Tại sao chuẩn ngôn ngữ lập trình quan trọng

- Ngôn ngữ C được chuẩn hóa bằng ISO C99 hay ANSI C (ISO/IEC 9899:1999)
- Ngôn ngữ C++ được chuẩn hóa bằng ISO C++98 (ISO/IEC 14882:1998), C++11 (ISO/IEC 14882:2011)
- Chương trình trong một môi trường chuyển sang môi trường khác không cần sửa đổi
- Đảm bảo sự đầu tư và ổn định cho sản phần mềm
- Trình dịch Visual C++ 6.0 không tương thích với rất nhiều đặc điểm của C++ chuẩn. Chương trình viết trên Visual C++ 6.0 mang sang Visual C++ mới hơn phải sửa đổi nhiều
- Visual C++ .NET 2003 là phiên bản C++ đầu tiên của Microsoft hỗ trợ tốt C++ chuẩn
- Giống như Internet Explorer cản trở sự phát triển của Internet, Visual C++ 6.0 cản trở sự phát triển lập trình C++

**XIN LỖI, Ở ĐÂY KHÔNG CÓ CHỖ  
CHO VISUAL C++ 6.0.**

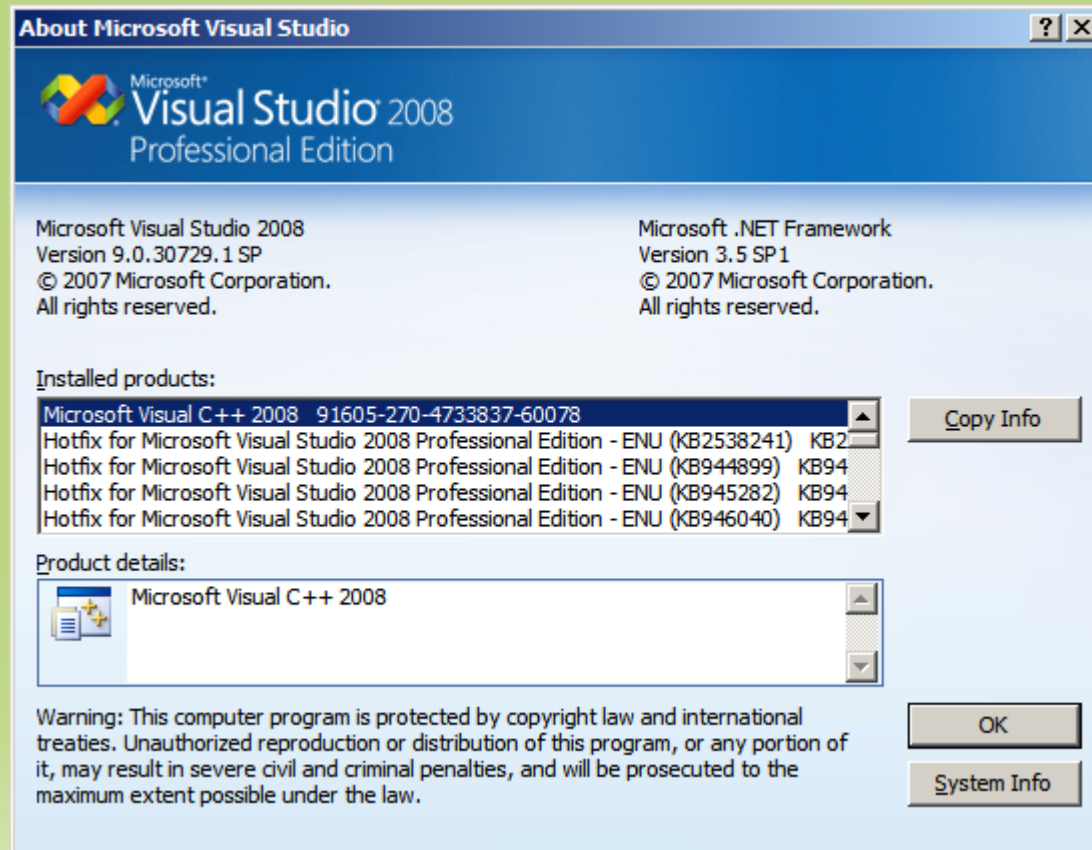
**KHÔNG GIẢI ĐÁP  
KHÔNG KHUYẾN KHÍCH**

# Quy trình phát triển một chương trình C/C++





# Mình họa quy trình phát triển



# Môi trường phát triển tích hợp (IDE)

Chạy chương trình

Công cụ dịch & liên kết

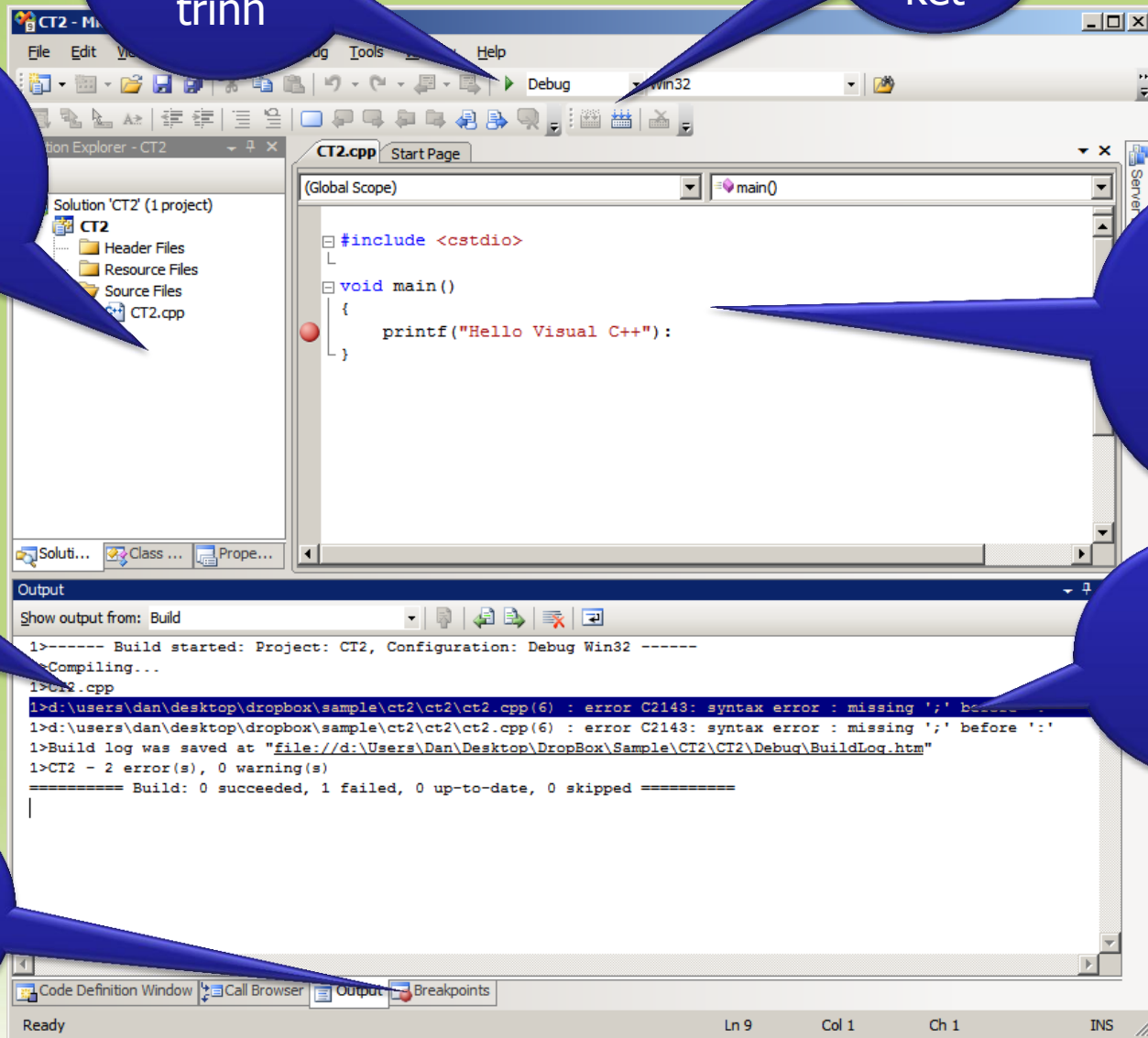
Quản lý đề án (tệp tin), thiết kế giao diện

Soạn thảo chương trình

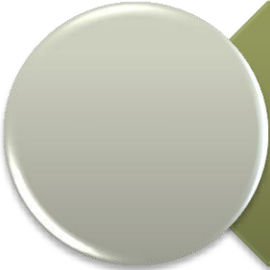
Dịch và liên kết

Thông báo lỗi thân thiện

Công cụ gỡ rối



# Nội dung



1 Giới thiệu Visual  
C++



2 Chương trình  
OpenGL đầu tiên



3 OpenGL: Các khái  
niệm cơ bản

## 2 Chương trình OpenGL đầu tiên

Các bước thực  
hiện

Chạy chương  
trình

Xem xét các phần  
của chương trình

# Các bước tạo một đề án OpenGL đầu tiên

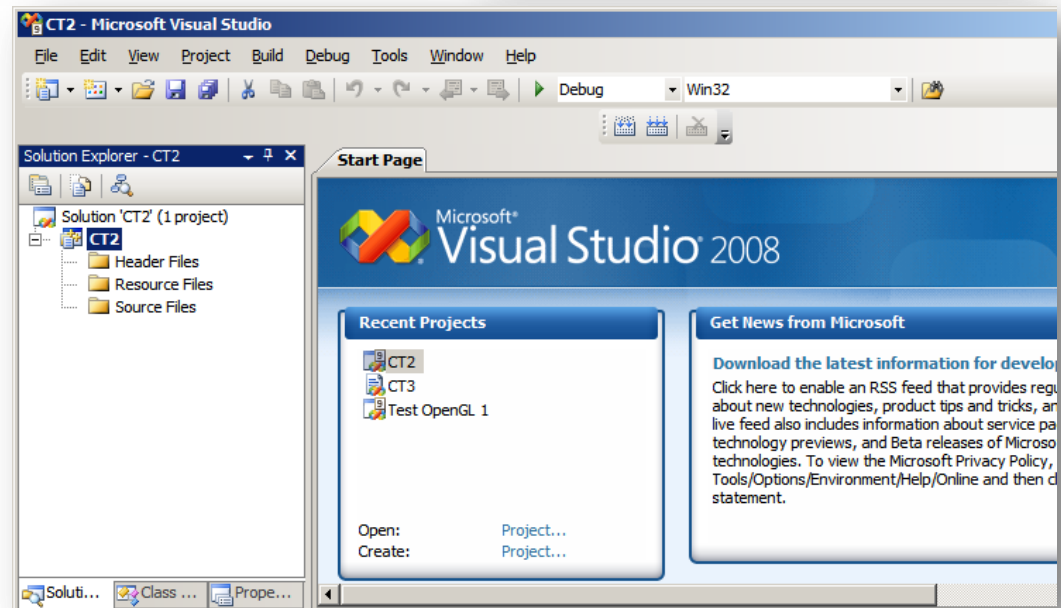
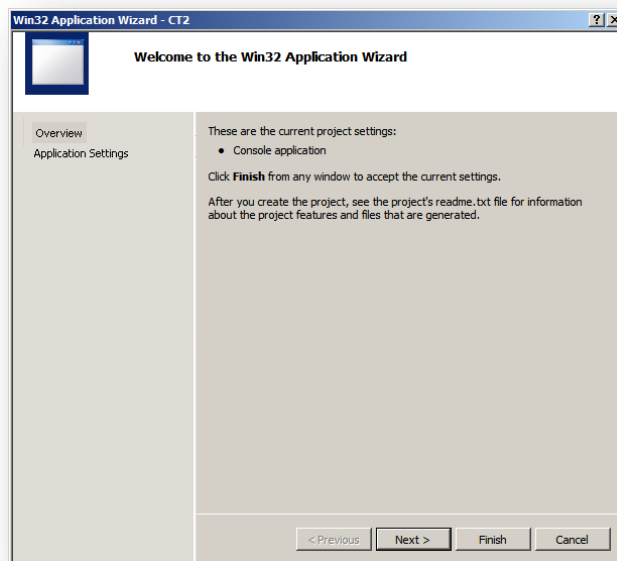
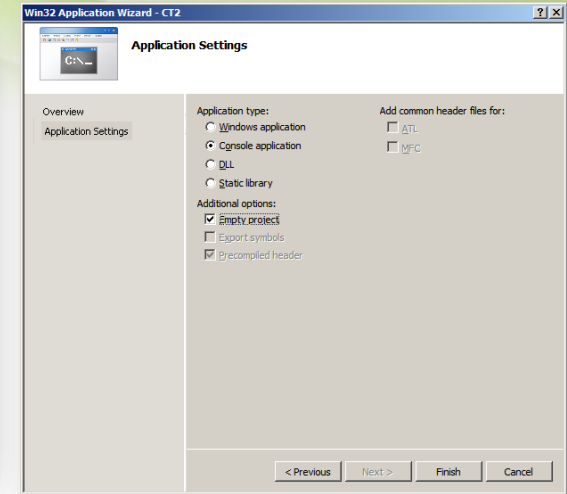
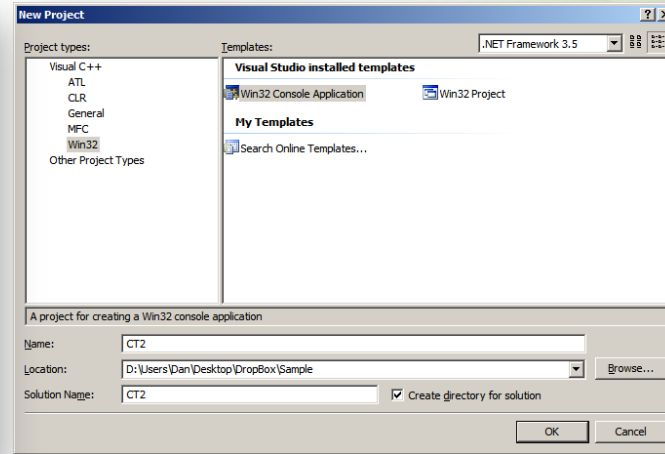
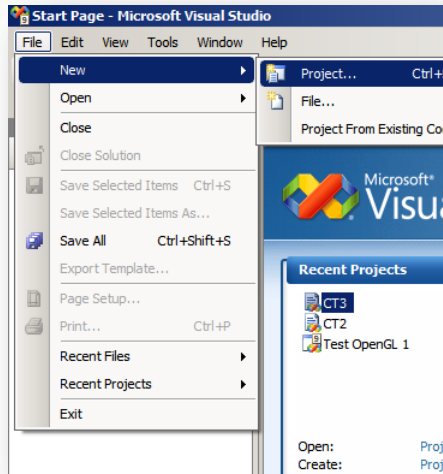


1. Mở Visual C++
2. Chọn menu **File/New/Project...**
3. Trong danh sách hiện ra chọn kiểu project là **Win32 Project** và nhập tên đề án, ví dụ CT1 rồi bấm **OK**, rồi **Next**
4. Trong hộp thoại **Application Settings**, chọn Application Type là **Windows Application** (đã được chọn mặc định), trong **Additional options** chọn **Empty project**, rồi bấm **Finish**

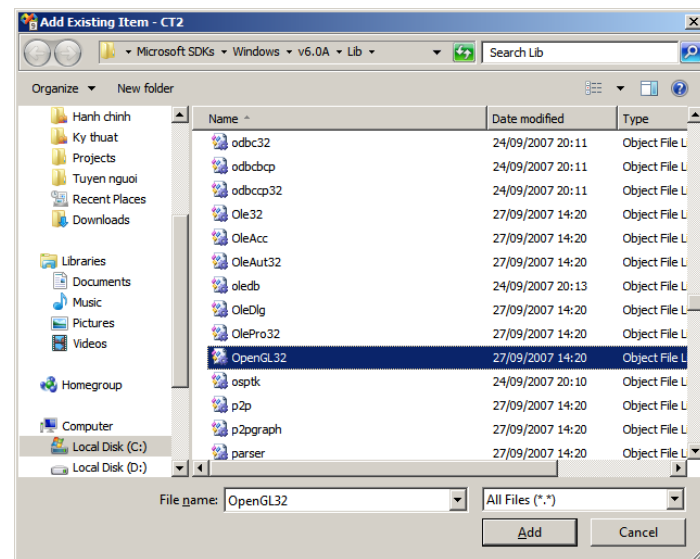
# Các bước tạo một đề án OpenGL đầu tiên (tiếp)

5. Tìm tệp tin `opengl.cpp` (download từ <http://db.tt/wr1sr3e>) và gắn thả vào danh sách tệp tin của project
6. Tìm các tệp tin **`opengl32.lib`** và **`glu32.lib`** (thư mục mặc định là `C:\Program Files\Microsoft SDKs\Windows\v6.0A\Lib`) rồi gắn thả vào danh sách tệp tin của project
7. Xây dựng tệp tin chạy `.exe` (bấm nút lệnh Build hoặc phím F7)
8. Chạy chương trình (Bấm Ctrl+F5)

# Minh họa 1

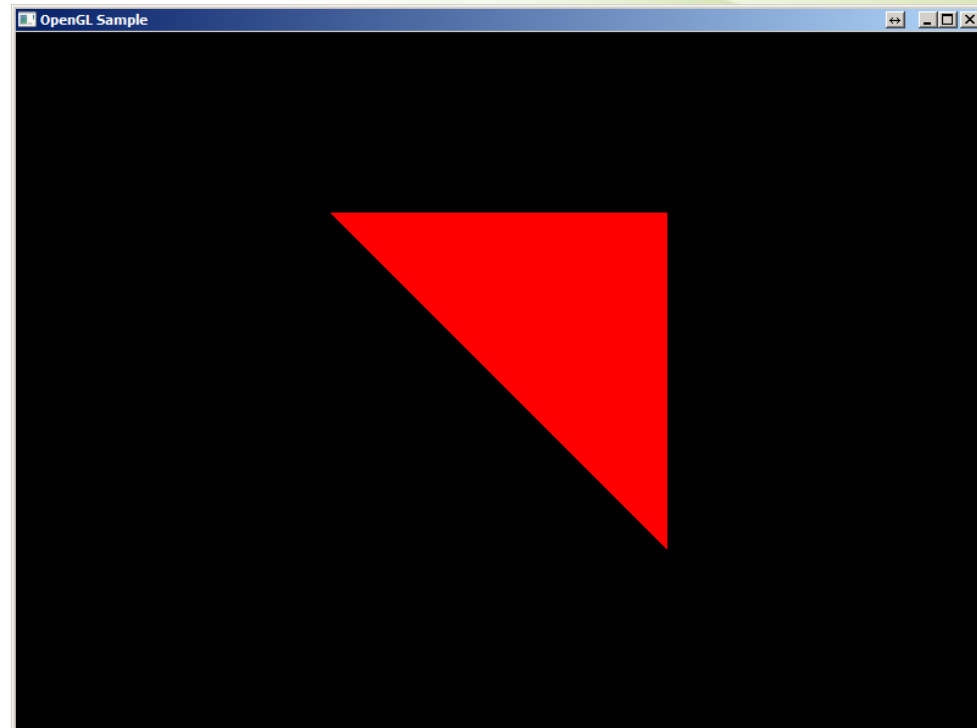
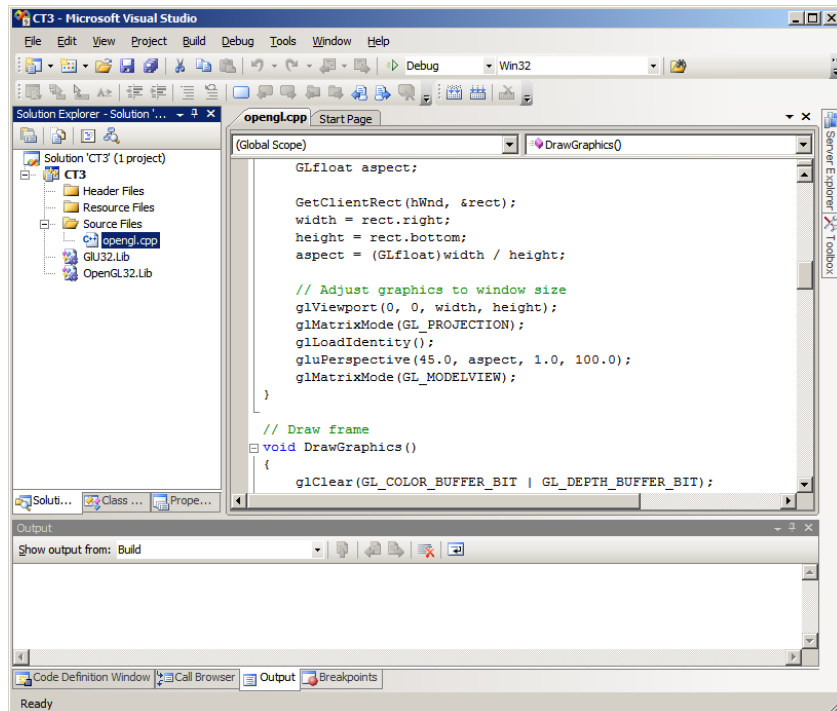








# Minh họa 3



# Sửa đổi chương trình

1. Thay đổi màu vẽ tam giác
2. Thay đổi màu nền
3. Thay đổi tọa độ điểm

# Cấu trúc của một chương trình đồ họa OpenGL

`void SetupPixelFormat()`

- Khởi tạo định dạng ảnh

`void InitGraphics()`

- Khởi tạo môi trường đồ họa OpenGL

`void ResizeGraphics()`

- Hàm thay đổi kích thước vùng vẽ phù hợp với kích thước cửa sổ

`void DrawGraphics()`

- Vẽ hình

`LONG WINAPI MainWndProc ()`

- Hàm xử lý các sự kiện xảy ra với cửa sổ

`int WINAPI WinMain ()`

- Hàm đầu vào chương trình Windows

# Chỉ thị biên dịch và biến tổng thể

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

Cho các hàm thư viện của  
Windows và thư viện đồ  
họa OpenGL

```
HWND  hWnd;  
HDC   hDC;  
HGLRC hRC;
```

Biến cho cửa sổ  
Biến cho vùng vẽ (DC)  
Biến vùng vẽ OpenGL

# SetupPixelFormat()

```
void SetupPixelFormat()  
{  
    PIXELFORMATDESCRIPTOR pfd, *ppfd;  
    int pixelformat;  
  
    ppfd = &pfd;  
  
    ppfd->nSize = sizeof(PIXELFORMATDESCRIPTOR);  
    ppfd->nVersion = 1;  
    ppfd->dwFlags = PFD_DRAW_TO_WINDOW |  
PFD_SUPPORT_OPENGL | PFD_DOUBLEBUFFER;  
    ppfd->dwLayerMask = PFD_MAIN_PLANE;  
    ppfd->iPixelFormat = PFD_TYPE_COLORINDEX;  
    ppfd->cColorBits = 16;  
    ppfd->cDepthBits = 16;  
    ppfd->cAccumBits = 0;  
    ppfd->cStencilBits = 0;  
  
    pixelformat = ChoosePixelFormat(hDC, ppfd);  
    SetPixelFormat(hDC, pixelformat, ppfd);  
}
```

- Khởi tạo định dạng điểm ảnh của vùng vẽ

# InitGraphics()

```
void InitGraphics()
{
    hDC = GetDC(hWnd);

    SetupPixelFormat();

    hRC = wglCreateContext(hDC);
    wglMakeCurrent(hDC, hRC);

    glClearColor(0, 0, 0, 0.5);
    glClearDepth(1.0);
    glEnable(GL_DEPTH_TEST);
}
```

- Khởi tạo môi trường đồ họa OpenGL
- Màu nền (0,0,0 – đen)

# ResizeGraphics()

```
void ResizeGraphics()
{
    // Get new window size
    RECT rect;
        int width, height;
        GLfloat aspect;

    GetClientRect(hWnd, &rect);
    width = rect.right;
    height = rect.bottom;
    aspect = (GLfloat)width / height;

    // Adjust graphics to window size
    glViewport(0, 0, width, height);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    gluPerspective(45.0, aspect, 1.0, 100.0);
    glMatrixMode(GL_MODELVIEW);
}
```

- Hàm thay đổi kích thước vùng vẽ phù hợp với kích thước cửa sổ
- Lấy kích thước cửa sổ
- Thay đổi kích thước vùng vẽ

# DrawGraphics()

```
void DrawGraphics()
{
    glClear(GL_COLOR_BUFFER_BIT |
GL_DEPTH_BUFFER_BIT);

    // Set location in front of camera
    glLoadIdentity();
    glTranslated(0, 0, -10);

    // Draw a triangle
    glBegin(GL_TRIANGLES);
    glColor3d(1, 0, 0);
    glVertex3d(-2, 2, 0);
    glVertex3d(2, 2, 0);
    glVertex3d(2, -2, 0);
    glEnd();

    // Show the new scene
    SwapBuffers(hDC);
}
```

- Vẽ hình tam giác
- Màu đỏ (1,0,0)
- Tọa độ



# MainWndProc()

```
// Handle window events and messages
LONG WINAPI MainWndProc (HWND hWnd, UINT uMsg,
WPARAM wParam, LPARAM lParam)
{
    switch (uMsg)
    {
        case WM_SIZE:
            ResizeGraphics();
            break;

        case WM_CLOSE:
            DestroyWindow(hWnd);
            break;

        case WM_DESTROY:
            PostQuitMessage(0);
            break;

        // Default event handler
        default:
            return DefWindowProc (hWnd, uMsg, wParam, lParam);
            break;
    }

    return 1;
}
```

- Hàm xử lý các sự kiện xảy ra với cửa sổ
  - Thay đổi kích thước
  - Đóng
  - Hủy
  - Mặc định

# WinMain()

```
int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE
hPrevInstance, LPSTR lpCmdLine, int nCmdShow)
{

    const LPCWSTR appname = TEXT("OpenGL Sample");

    WNDCLASS wndclass;
    MSG      msg;

    // Define the window class
    wndclass.style      = 0;
    wndclass.lpfnWndProc = (WNDPROC)MainWndProc;
    wndclass.cbClsExtra  = 0;
    wndclass.cbWndExtra  = 0;
    wndclass.hInstance   = hInstance;
    wndclass.hIcon        = LoadIcon(hInstance, appname);
    wndclass.hCursor      = LoadCursor(NULL, IDC_ARROW);
    wndclass.hbrBackground = (HBRUSH)(COLOR_WINDOW+1);
    wndclass.lpszMenuName = appname;
    wndclass.lpszClassName = appname;

    // Register the window class
    if (!RegisterClass(&wndclass)) return FALSE;

    // Create the window
```

```
    hWnd = CreateWindow(appname, appname,
WS_OVERLAPPEDWINDOW | WS_CLIPSIBLINGS |
WS_CLIPCHILDREN, CW_USEDEFAULT, CW_USEDEFAULT, 800,
600, NULL, NULL, hInstance, NULL); if (!hWnd) return FALSE;

    // Initialize OpenGL
    InitGraphics();

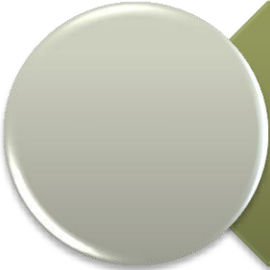
    // Display the window
    ShowWindow(hWnd, nCmdShow);
    UpdateWindow(hWnd);

    // Event loop
    while (1)
    {
        if (PeekMessage(&msg, NULL, 0, 0, PM_NOREMOVE) ==
TRUE)
        {
            if (!GetMessage(&msg, NULL, 0, 0)) return TRUE;

            TranslateMessage(&msg);
            DispatchMessage(&msg);
        }
        DrawGraphics();
    }

    wglDeleteContext(hRC);
    ReleaseDC(hWnd, hDC);
}
```

# Nội dung



1 Giới thiệu Visual  
C++



2 Chương trình  
OpenGL đầu tiên



3 OpenGL: Các khái  
niệm cơ bản

# 3 OpenGL: Các khái niệm cơ bản

## Các đối tượng hình học cơ bản

- Điểm
- Đoạn thẳng
- Đa giác

## Quan sát (Viewing)

- Màu sắc
- Bóng màu (shading)
- Phép biến đổi quan sát (Viewing Transformation)
- Phép chiếu
- Ánh sáng

# Các đối tượng hình học cơ bản

- OpenGL định nghĩa những đối tượng hình học cơ bản là: điểm, đoạn thẳng và đa giác
- Mỗi đối tượng hình học được mô tả bằng loại đối tượng và một tập hợp các đỉnh
- Lệnh glVertex\*() được dùng để chỉ định một đỉnh. Dấu \* được sử dụng vì có một số biến thể của lệnh glVertex()
- Ví dụ glVertex3fv(): **3** là số tham số cần truyền vào của lệnh, **f** chỉ ra tham số kiểu **float**, **v** cho biết tham số thuộc loại véc-tơ (mảng)

# Vẽ điểm

- Để vẽ điểm, truyền tham số `GL_POINTS` cho lệnh `glBegin()`. Kết thúc vẽ bằng `glEnd()`. Muốn thay đổi kích thước điểm ảnh, dùng `glPointSize()`
- Ví dụ

```
glBegin(GL_POINTS);  
    glVertex2f(0.0, 0.0);  
    glVertex2f(0.0, 3.0);  
    glVertex2f(4.0, 3.0);  
    glVertex2f(6.0, 1.5);  
    glVertex2f(4.0, 0.0);  
glEnd();
```

# Vẽ đoạn thẳng

- Để vẽ đoạn thẳng có thể dùng các tham số sau cho lệnh glBegin()
  - GL\_LINES: Vẽ một loạt các đoạn thẳng tách biệt
  - GL\_LINE\_STRIP: Vẽ một đoạn thẳng từ đỉnh đầu tiên cuối cùng. Đoạn thẳng có thể giao nhau tùy tiện
  - GL\_LINE\_LOOP: Tương tự như GL\_STRIP, ngoại trừ cạnh cuối được tự động thêm vào để đóng đa giác

# Vẽ đa giác

- Có một số lệnh để vẽ đa giác: 3 đỉnh (hình tam giác, GL\_TRIANGLES) và bốn đỉnh (tứ giác, GL\_QUADS) và đa giác tổng quát (GL\_POLYGON)
- Ví dụ

```
glBegin(GL_POLYGON);  
    glColor3f(1.0, 1.0, 0.0); // Màu vàng  
    glVertex2f(0.0, 0.0);  
    glVertex2f(0.0, 3.0);  
    glVertex2f(4.0, 3.0);  
    glVertex2f(6.0, 1.5);  
    glVertex2f(4.0, 0.0)  
glEnd();
```



# Quan sát (Viewing) - Màu sắc

- Khác với vẽ trên giấy, trên máy tính ta cần xóa và đặt màu nền trước khi vẽ ra lệnh vẽ các đối tượng. Trong OpenGL, lệnh xóa và đặt màu là `glClearColor ()`
- Tham số của lệnh là bộ 4 màu thành phần: đỏ, xanh lá cây, xanh dương và độ mờ (Red, Green, Blue, Alpha blending – RGBA). Giá trị màu thay đổi từ 0.0 đến 1.0
- Để thiết lập một màu sắc, sử dụng lệnh `glColor3f ()`. Nó có ba thông số nằm giữa 0.0 và 1.0

- Giá trị một số màu cơ bản

`glColor3f(0.0, 0.0, 0.0); //black`

`glColor3f(1.0, 0.0, 0.0); //red`

`glColor3f(0.0, 1.0, 0.0); //green`

`glColor3f(0.0, 1.0, 1.0); //cyan`

`glColor3f(0.0, 0.0, 1.0); //blue`

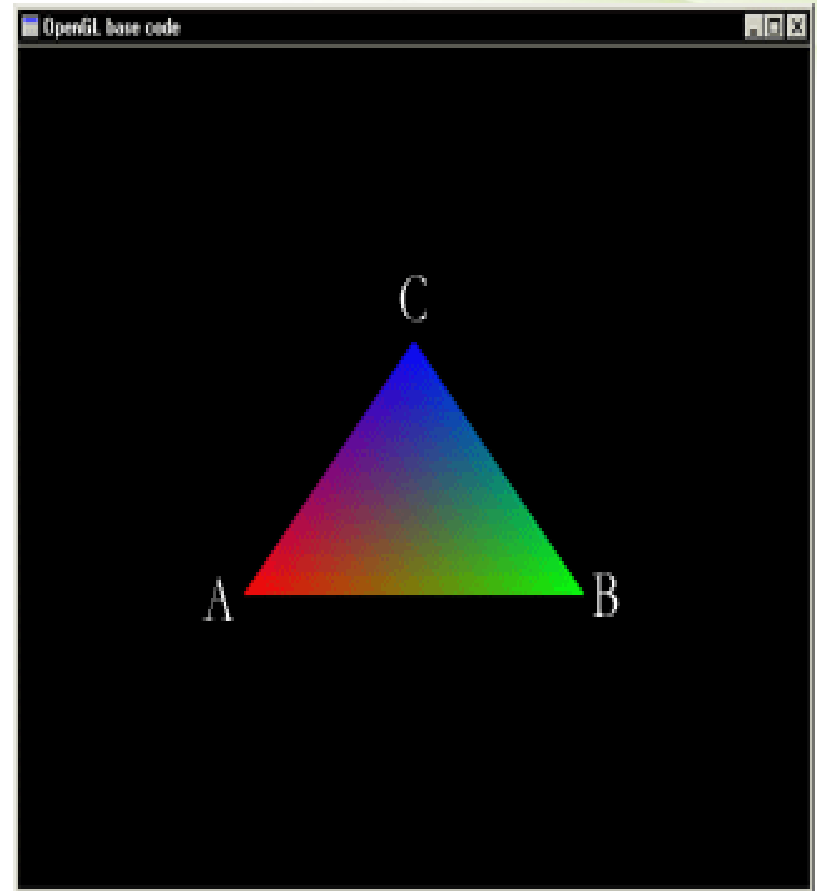
`glColor3f(1.0, 1.0, 0.0); //yellow`

`glColor3f(1.0, 0.0, 1.0); //magenta`

`glColor3f(1.0, 1.0, 1.0); //white`

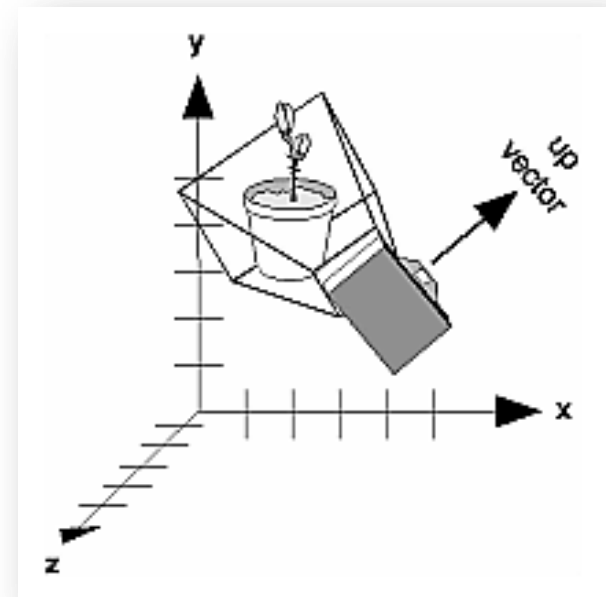
# Bóng màu (shading)

- Trong OpenGL có thể đặt màu cho mỗi đỉnh riêng rẽ của đối tượng. OpenGL sẽ tự động nội suy để ra được màu ở vùng trung gian
- Phép đổ bóng này được gọi là Đổ bóng Gouraud



# Phép biến đổi quan sát (Viewing Transformation)

- Để thay đổi hình ảnh của một đối tượng, cần thay đổi vị trí của đối tượng hoặc của người quan sát. Trong đồ họa máy tính, một camera đặc trưng cho người quan sát
- Để reset lại góc quan sát của camera, dùng lệnh `glLoadIdentity()` (Đặt ma trận chuyển đổi về ma trận đơn vị)
- Dùng lệnh `gluLookAt()` để đặt vị trí của camera (người quan sát) và hướng nhìn
- Ví dụ: lệnh `gluLookAt(0.0, 0.0, 5.0, 0.0, 0.0, 0.0, 0.0, 1.0, 0.0)`
- đặt camera tại điểm  $(0.0, 0.0, 5.0)$ , nhìn về điểm  $(0.0, 0.0, 0.0)$  và đặt véc-tơ hướng lên của camera là  $(0,1,0)$  (tức là nằm ngang)

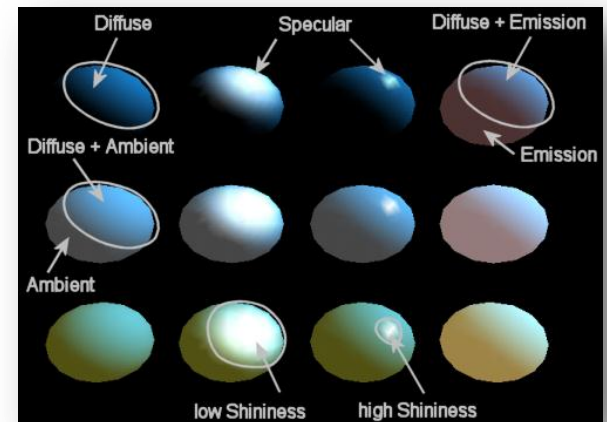
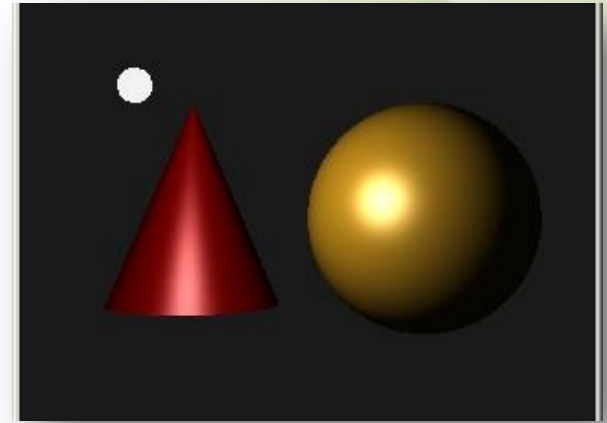


# Phép chiếu (projection)

- Khi vẽ trong OpenGL luôn sử dụng kích thước thật của đối tượng. Thể hiện của đối tượng được tính toán tùy thuộc vào vị trí camera và phương pháp chiếu hình
- Có hai phương pháp chiếu hình chính: trực đo (orthographic) và phối cảnh (perspective)
- Để đặt phương pháp chiếu trực đo, sử dụng lệnh `glOrtho()`, với phối cảnh dùng lệnh `glFrustum()`

# Ánh sáng

- Hiệu ứng ánh sáng là rất quan trọng trong đồ họa 3D vì nếu không có ánh sáng một đối tượng 3D sẽ trông như đối tượng 2D
- OpenGL cung cấp hai loại nguồn sáng: hướng và vị trí
- Một nguồn sáng hướng được coi là nằm cách đối tượng một khoảng cách vô hạn (như AS mặt trời). Vì vậy, các tia ánh sáng được coi là song song khi tiếp cận các đối tượng
- Ngược lại, một nguồn sáng vị trí, gần hoặc ở ngay trong các cảnh và có các tia sáng không song song (như AS đèn)
- Lệnh `glLightfv()` được sử dụng để đặt vị trí của ánh sáng, cho cả AS hướng và vị trí. Cũng được dùng khi chỉ định nguồn sáng có màu thuộc loại nền, khuếch tán, phản chiếu, hay phát xạ



# Ánh sáng (tiếp)

- Trong OpenGL có thể tạo đồng thời tối đa tám nguồn ánh sáng có tên lần lượt GL\_LIGHT0, GL\_LIGHT1,...
- Để tạo ra một nguồn ánh sáng, phải chọn tên, vị trí nguồn sáng và các thông số màu sắc và chất lượng
- Ví dụ

```
GLfloat light_diffuse[] = {1.0, 1.0, 1.0, 1.0};
GLfloat light_position[] = {1.0, 1.0, 1.0, 0.0};
glLightfv(GL_LIGHT0, GL_DIFFUSE, light_diffuse);
glLightfv(GL_LIGHT0, GL_POSITION, light_position);
glEnable(GL_LIGHTING);
glEnable(GL_LIGHT0);
```

# Câu hỏi?

