BÀI 4

LẬP TRÌNH TƯƠNG TÁC VÀ MFC TRONG MÔ PHỎNG

Tóm tắt

Bài này giới thiệu các kỹ thuật giúp chương trình mô phỏng có tính tương tác cao hơn. Các vấn đề chính được trình bày gồm:

- Khái niệm về lập trình hướng sự kiện
- Tương tác với chương trình bằng bàn phím và con chuột
- Lập trình OpenGL sử dụng thư viện MFC

Nội dung

1. Lập trình hướng sự kiện
2. Lập trình tương tác trong Windows: bàn phím và chuột
3. Lập trình OpenGL sử dụng thư viện MFC

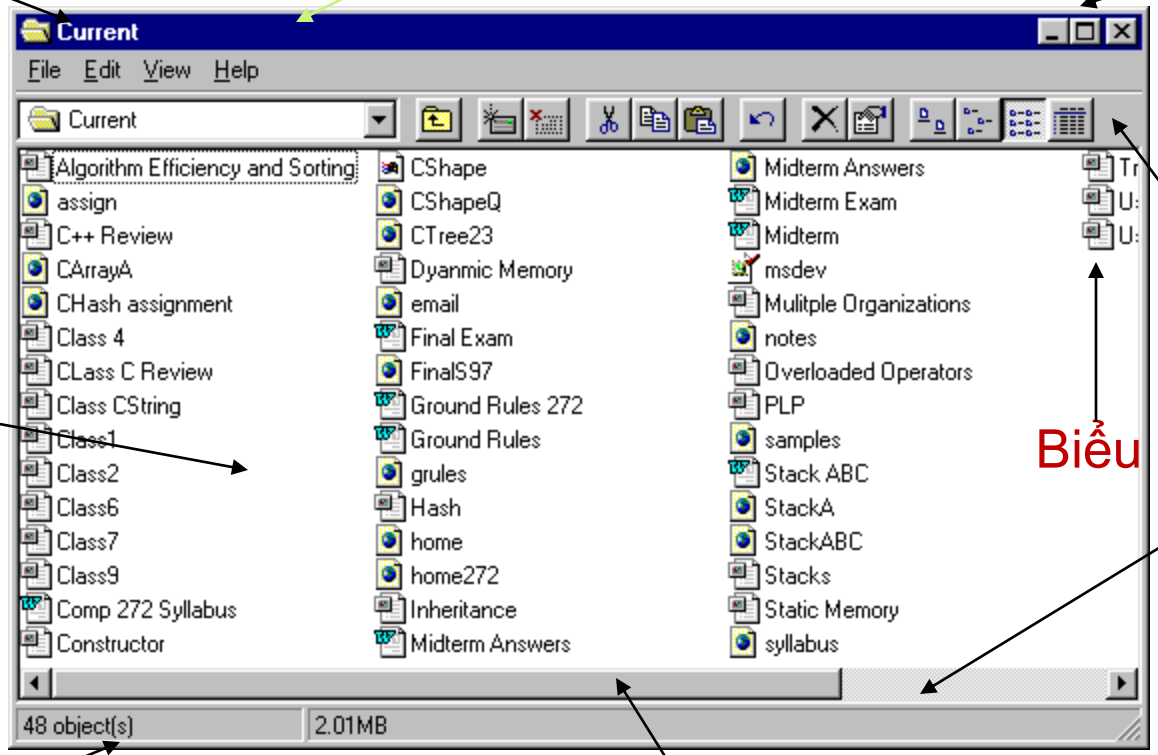
- Cơ bản về thư viện MFC
- Khởi tạo môi trường OpenGL trong MFC: Lớp OpenGlInit

Giao diện người dùng (UI)

- Giao diện người dùng là kết nối giữa người dùng và máy tính
  - Giao diện dòng lệnh (Console)
    - Dựa trên văn bản
  - Giao diện người dùng đồ họa (GUI)
    - Giao diện định hướng trực quan (WYSIWIG – What You See Is What You Get)
    - Người dùng tương tác với các đối tượng đồ họa
    - Trực quan hơn

Giao diện Tính năng chính

Cửa sổ!



Cửa sổ tiêu đề (title)

Thanh tiêu đề

Thu nhỏ
Phóng to
Đóng

Thực đơn

Thanh công

Vùng nội dung

Biểu tượng

Thanh cuộn

Kéo thanh cuộn

Thanh trạng thái

Không có tiêu chuẩn cho GUI

- ANSI / ISO C + + không không cung cấp khả năng tạo ra các giao diện người dùng đồ họa (GUI)
- MFC: Một bộ sưu tập lớn các lớp (và khuôn mẫu) trợ giúp lập trình trong Visual C++ tạo ra các ứng dụng mạnh mẽ một cách nhanh chóng trên Windows
- Thư viện tài liệu của Microsoft có tại: http://msdn.microsoft.com/library/

Tương tác người dùng

- Người dùng tương tác với giao diện đồ họa thông qua các thông điệp
- Khi một sự kiện xảy ra, hệ điều hành sẽ gửi một thông điệp đến chương trình
- Lập trình chức năng đáp ứng với những thông điệp này được gọi là lập trình hướng sự kiện
  - Thông điệp có thể được tạo ra bởi hành động của người dùng, các ứng dụng khác, và hệ điều hành

So sánh lập trình hướng sự kiện với lập trình văn bản

- Các chương trình đồ hoạ có một cấu trúc khác cơ bản với các chương trình dựa trên giao diện văn bản (console)
- Chương trình dựa trên giao diện văn bản :
  - yêu cầu người sử dụng đưa thông tin vào;
  - thực hiện một số thao tác;
  - in một số kết quả;
  - yêu cầu người sử dụng đưa thông tin vào;
  - tiếp tục
- Các chương trình quyết định khi nào xuất/nhập
- Mô hình giao diện đồ hoạ: người sử dụng kiểm soát!

Lập trình hướng sự kiện

- Cấu trúc chương trình giao diện cần đáp ứng các sự kiện người dùng. Các loại sự kiện: nhấn chuột, di chuyển chuột, bấm phím, v.v.
  - Trong Windows, được gọi là thông điệp (message)
- Cấu trúc điều khiển chính là một vòng lặp sự kiện:

```
while (1) { // Lặp vô tận
    • chờ đợi cho sự kiện tiếp theo
    • gửi sự kiện tới thành phần giao diện thích hợp
}
```

- Bạn chỉ cần viết mã để đáp ứng với các sự kiện.
- Mô hình giao diện đồ hoạ: Người sử dụng sẽ có thể đưa ra bất kỳ đầu vào bất cứ lúc nào →Không tuần tự!

Nội dung

1. Lập trình hướng sự kiện

2. Lập trình tương tác trong Windows: bàn phím và chuột

3. Lập trình OpenGL sử dụng thư viện MFC

- Cơ bản về thư viện MFC
- Khởi tạo môi trường OpenGL trong MFC: Lớp OpenGlInit

Vòng lặp chính của chương trình Windows

```
LRESULT WindowProc( HWND hWnd,  UINT   msg, WPARAM wParam,  LPARAM lParam )
{
   switch (uMsg)
   {
   case WM_SIZE:
      ResizeGraphics();
      break;

   case WM_CLOSE:
      DestroyWindow(hWnd);
      break;

   case WM_DESTROY:
      PostQuitMessage(0);
      break;

 return DefWindowProc (hWnd, uMsg, wParam, lParam);
      break;
   }

   return 1;
}
```

Các sự kiện chính của Windows

- Cửa sổ
  - WM_CREATE
  - WM_DESTROY
  - WM_MOVE
  - WM_SIZE
  - WM_ACTIVATE
  - WM_SETFOCUS
  - WM_CLOSE
  - WM_ERASEBKGND
  - WM_CONTEXTMENU
- Bàn phím
  - WM_KEYDOWN
  - WM_KEYUP
  - WM_CHAR
- Đồng hồ
  - WM_TIMER
- Chuột
  - WM_MOUSEMOVE
  - WM_LBUTTONDOWN
  - WM_LBUTTONUP
  - WM_LBUTTONDBLCLK
  - WM_RBUTTONDOWN
  - WM_RBUTTONUP
  - WM_RBUTTONDBLCLK
  - WM_MBUTTONDOWN
  - WM_MBUTTONUP
  - WM_MBUTTONDBLCLK
  - WM_MOUSEWHEEL

Ví dụ: Xử lý sự kiện chuột

```
case WM_MOUSEMOVE:
  {
    // Left mouse button
    if (wParam & MK_LBUTTON)
    {
      m_fRotX += (float)0.5f * diffY;
      m_fRotY += (float)0.5f * diffX;
    }
    // Right mouse button
    else if (wParam & MK_RBUTTON)
    {
      m_fZoom -= (float)0.1f * diffY;
    }
    // Middle mouse button
    else if (wParam & MK_MBUTTON)
    {
      m_fPosX += (float)0.05f * diffX;
      m_fPosY -= (float)0.05f * diffY;
    }
  }

  break;
```

Ví dụ: Xử lý sự kiện bàn phím

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

  case VK_SPACE:
    g_bOrbitOn = !g_bOrbitOn;
    break;
  case VK_LEFT:
    g_iLeftRightView-=1;
    break;
  case VK_RIGHT:
    g_iLeftRightView+=1;
    break;
  case VK_UP:
    g_iUpDownView+=1;
    break;
  case VK_DOWN:
    g_iUpDownView-=1;
    break;
}
```

MayaCamera

- Để thao tác với mô hình mô phỏng bằng chuột theo kiểu phần mềm Maya của Autodesk, có thể sử dụng lớp MayaCamera
- Khi đó, các thao tác điều khiển camera với chuột như sau:
  - phím trái chuột để xoay mô hình
  - phím phải chuột để thu phóng mô hình
  - phím giữa để tịnh tiến mô hình

Nội dung

1. Lập trình hướng sự kiện

2. Lập trình tương tác trong Windows: bàn phím và chuột

3. Lập trình OpenGL sử dụng thư viện MFC

- Cơ bản về thư viện MFC
- Khởi tạo môi trường OpenGL trong MFC: Lớp OpenGlInit

Cơ bản về thư viện MFC

Liên kết ngoài

KHỞI TẠO MÔI TRƯỜNG OPENGL TRONG MFC: LỚP OPENGLINIT

Khởi tạo môi trường OpenGL trong MFC: Lớp OpenGlInit

- Lớp OpenGlInit do tôi viết để giúp khởi tạo môi trường đồ hoạ trong ứng dụng MFC
- Tính năng tương tự như các hàm trong chương trình đầu tiên: opengl.cpp
  - void SetupPixelFormat()
  - void InitGraphics()
  - void ResizeGraphics()
  - void DrawGraphics()
- Được "đóng gói" vào một lớp để dễ dàng sử dụng

Khai báo của lớp

```
class OpenGlInit
{
public:
  // Hàm khởi tạo
  OpenGlInit();

  // Gọi trong hàm OnCreate()
  // Sửa nội dung tuỳ theo chương trình

  void OnCreate(HDC _hDC);
  // Gọi trong hàm OnSize(UINT nType, int cx, int cy)

  void OnSize(UINT nType, int cx, int cy);
  // Gọi trong hàm OnDestroy()

  void OnDestroy();
  // Gọi trong hàm OnDraw(CDC* /*pDC*/)
  // Sửa nội dung tuỳ theo chương trình

  void OnDraw();

  // Thêm trong hàm OnEraseBkgnd(CDC* /*pDC*/)
  // Thay thế nội dung bằng:
  // return TRUE;

  // Thêm trong hàm PreCreateWindow()
  // Vào đầu hàm
  // cs.style |= WS_CLIPSIBLINGS |
WS_CLIPCHILDREN;

  // Thêm trong hàm OnTimer(UINT_PTR nIDEvent)
  // Vào cuối hàm
  // RedrawWindow();
protected:
  void SetupPixelFormat();
public:
  HDC     hDC;
  HGLRC   hRC;
  int timerID;     // ID của bộ đếm thời gian (timer)
  int timerElapse;  // Khoảng thời gian giữa các lần
vẽ lại
};
```

Cách sử dụng

1. Tạo một ứng dụng MFC kiểu SDI có tên TestGl
2. Copy đoạn chương trình của lớp **OpenGlInit** vào đầu lớp View của ứng dụng MFC (ví dụ CTestGlView)
3. Thêm khai báo biến OpenGlInit **openGlInit**;
4. Thêm các hàm xử lý các sự kiện sau trong lớp View, nếu chưa có:

   PreCreateWindow(), OnCreate(), OnSize(), OnDraw(), OnEraseBkgnd(), OnTimer(), OnDestroy()

4. Thêm các dòng lệnh tương ứng vào các hàm xử lý sự kiện ở trên

PreCreateWindow()

```
BOOL CTestGlView::PreCreateWindow(CREATESTRUCT& cs)
{
   //*OGLI*
   cs.style |= WS_CLIPSIBLINGS | WS_CLIPCHILDREN;
   //*OGLI*

   return CView::PreCreateWindow(cs);
}
```

OnCreate()

```
int CTestGlView::OnCreate(LPCREATESTRUCT lpCreateStruct)
{
  if (CView::OnCreate(lpCreateStruct) == -1)
    return -1;

  //*OGLI*
  openGlInit.OnCreate(this->GetDC()->m_hDC);

  SetTimer(openGlInit.timerID, openGlInit.timerElapse, NULL);
  //*OGLI*

  return 0;
}
```

OnSize()

```
void CTestGlView::OnSize(UINT nType, int cx, int cy)
{
   CView::OnSize(nType, cx, cy);

   //*OGLI*
   openGlInit.OnSize(nType, cx, cy);
   //*OGLI*
}
```

OnDraw()

```
void CTestGlView::OnDraw(CDC* /*pDC*/)
{
   CStartMFCDoc* pDoc = GetDocument();
   ASSERT_VALID(pDoc);
   if (!pDoc)
      return;

   //*OGLI*
   openGlInit.OnDraw();
   //*OGLI*
}
```

OnEraseBkgnd()

```
BOOL CTestGlView::OnEraseBkgnd(CDC* pDC)
{
   //*OGLI*
   return TRUE;
   //*OGLI*

//   return CView::OnEraseBkgnd(pDC);
}
```

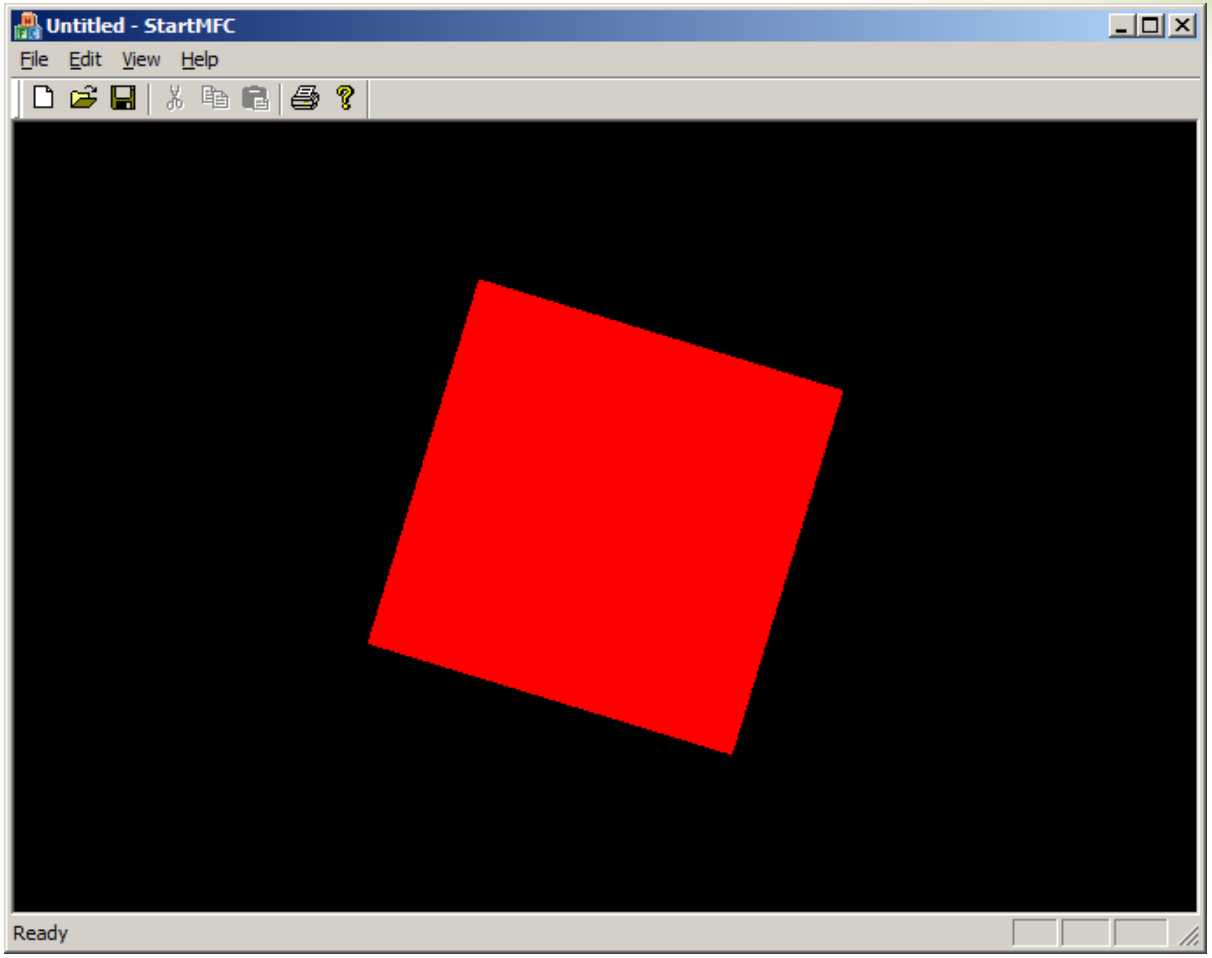
OnTimer()

```
void CTestGlView::OnTimer(UINT_PTR nIDEvent)
{
   CView::OnTimer(nIDEvent);

   //*OGLI*
   RedrawWindow();
   //*OGLI*
}
```

OnDestroy()

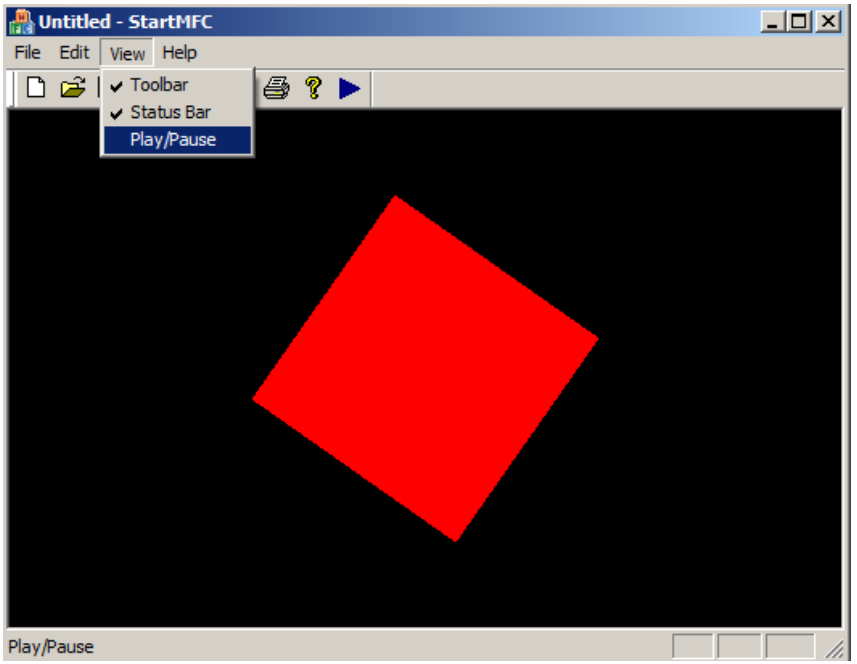
```
void CTestGlView::OnDestroy()
{
  //*OGLI*
  KillTimer(openGlInit.timerID);

  openGlInit.OnDestroy();
  //*OGLI*

  CView::OnDestroy();
  // TODO: Add your message handler code here
}
```

Minh họa



Bổ sung điều khiển: Nút lệnh và menu

- Phần sau đây sẽ minh hoạ việc tạo một nút lệnh trên thanh công cụ và một mục tương ứng trên menu, và xử lý sự kiện khi người dùng bấm vào các phần tử giao diện đó
- Chương trình mô phỏng sẽ chạy/dừng khi sự kiện nói trên diễn ra



Các bước thực hiện

1. Thêm một biến kiểu bool vào lớp OpenGlInit để xác định hình quay liên tục hay đứng yên

```
bool rotating;
```

2. Sửa hàm OnDraw() để chỉ khi rotating=true mới tăng góc định vị.

```
if (rotating)
        angle++;
glRotatef(float(angle), 0.0f, 0.0f, 1.0f);
```

3. Mở Resource View (Ctrl+Shift+E)
   - Tìm Toolbar để vẽ nút lệnh, đặt ID là ID_PLAY
   - Tìm Menu để thêm một mục vào menu view, tiêu đề là Play/Pause, ID cũng là ID_PLAY
4. Thêm hàm xử lý sự kiện ID_PLAY vào lớp View trong đó có lệnh
   - rotating =!rotating;

Câu hỏi?