Chương 5

MÔ HÌNH WIREFRAME

Thế giới quanh chúng ta hầu hết đều là những đối tượng 3 chiều (3D, gồm chiều rộng, chiều dài và chiều cao). Bằng những kỹ thuật như vẽ, chụp hình, quay phim,… chúng ta thường cố gắng để biểu diễn hình ảnh thực 3 chiều của đối tượng dưới những góc nhìn khác nhau bằng các hình vẽ trên mặt phẳng hai chiều của giấy, khung vẽ hay màn ảnh,… để thể hiện lại các đối tượng. Các hình ảnh đó phải tuân theo một số quy luật về phối cảnh, sáng tối nhằm giúp người xem có thể tưởng tượng lại hình ảnh mong muốn.

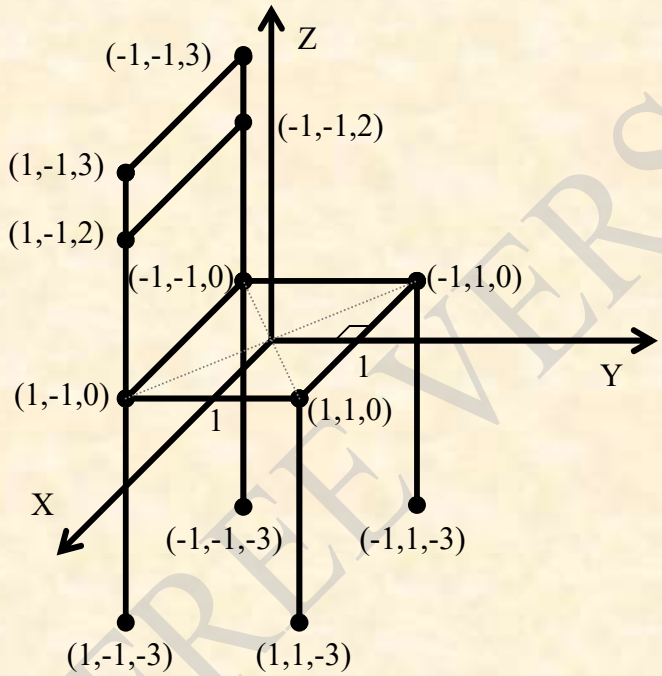
Ngày nay, dưới sự tiến bộ của máy tính điện tử thì nhu cầu thể hiện các đối tượng bằng máy tính trở thành một nhu cầu thiết yếu. Thể hiện các đối tượng thực bằng máy tính, đặc biệt là biểu diễn các đối tượng 3 chiều là một công việc đem lại lợi ích rất lớn, bởi nó dễ dàng cho chúng ta những góc nhìn khác nhau, những cách thể hiện, những biến đổi và nhiều tác động khác một cách hiệu quả mà không mất nhiều thời gian và tiền của. Trước đây, các kiến trúc sư thường làm việc trên bàn giấy với công cụ là bút vẽ và giấy, do đó công việc rất khó nhọc. Đôi khi, một bảng vẽ phải được tiến hành vẽ lại toàn bộ chỉ vì một vài sai sót nhỏ, hay vì nhu cầu sửa đổi một số chi tiết trên bản vẽ. Ngày nay, với sự trợ giúp đắc lực của máy tính, các kiến trúc sư có thể nhanh chóng thể hiện các ý tưởng của mình trên máy tính, công việc thể hiện bảng vẽ trở nên dễ dàng hơn, các sai sót hay sửa đổi được thực hiện một cách dễ dàng mà không nhất thiết phải thực hiện vẽ lại toàn bộ bảng vẽ. Ngoài ra, máy tính còn có thể thể hiện ra nhiều bảng vẽ khác nhau cho cùng một đối tượng dưới nhiều góc nhìn, và nhiều chức năng tiện ích khác đi kèm như: Tính kết cấu lực, tạo phối cảnh, tạo đoạn phim mô tả công trình,…

Trong phần này, chúng ta sẽ giới thiệu một số khái niệm và kỹ thuật trong việc biểu diễn các đối tượng 3 chiều trên máy tính, bắt đầu từ các đối tượng đơn giản như các các mặt, các hình khối, các đa diện,…

1. MÔ HÌNH WIREFRAME

Mô hình Wireframe, hay còn gọi là mô hình khung dây, lưu trữ thông tin về hình dáng (hay bộ khung) của đối tượng. Thông tin lưu trữ được tổ chức thành 2 danh sách, một *danh sách đỉnh*, và một *danh sách cạnh* được tạo nên từ các đỉnh *chứa trong danh sách đỉnh*. Danh sách đỉnh lưu giữ tọa độ các đỉnh của đối tượng mà mỗi đỉnh bao gồm các thành phần tọa độ (x,y,z), danh sách cạnh lưu giữ danh sách các cặp thứ tự điểm tạo nên cạnh mà tọa độ của các điểm này chứa trong danh sách đỉnh.

Ví dụ: Để lưu trữ hình dáng một chiếc ghế gỗ thô sơ như hình vẽ sau:



Hình 5.1. Minh họa công đoạn số hóa đối tượng 3 chiều

Thông tin mô tả đối tượng chiếc ghế là 2 danh sách sau:

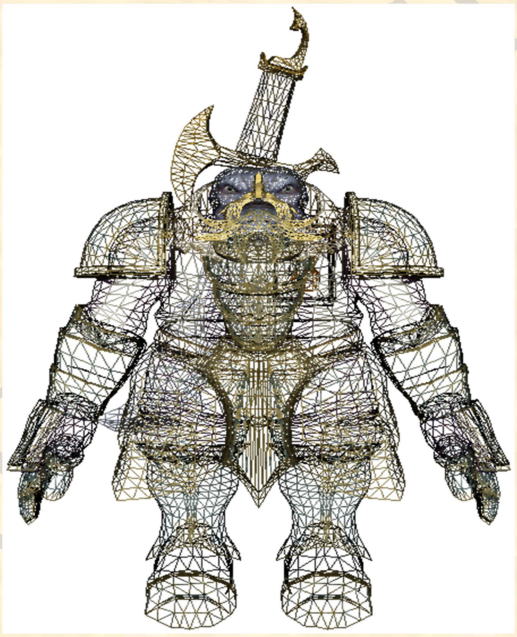
Bảng 5.1. Danh sách thông tin lưu trữ theo mô hình WireFrame của chiếc ghế

Danh sách đỉnh			
Stt	X	Y	Z
1	1	1	-3

Danh sách cạnh		
Stt	đỉnh 1	đỉnh 2
1	1	5

2	-1	1	-3
3	-1	-1	-3
4	1	-1	-3
5	1	1	0
6	-1	1	0
7	-1	-1	0
8	1	-1	0
9	1	-1	2
10	-1	-1	2
11	1	-1	3
12	-1	-1	3

2	2	6
3	3	12
4	4	11
5	5	6
6	6	7
7	7	8
8	8	5
9	9	10
10	11	12



Hình 5.2. Mô hình wireframe cho một nhân vật trong game

Có nhiều cách thức để tổ chức lưu trữ và xử lý thông tin của đối tượng theo mô hình WireFrame trong một ứng dụng và mỗi hình thức đều có những ưu nhược điểm riêng. Ở đây, chúng ta sẽ từng bước xây dựng một lớp (class) có tên CWireFrame, với các thuộc tính và các phương thức có thể đáp ứng được bài toán mô phỏng đối tượng 3 chiều trong không gian trên máy tính với các chức năng cơ bản như: Hiển thị

hình ảnh đối tượng 3D lên mặt phẳng 2 chiều của thiết bị đồ họa (màn hình), thay đổi vị trí quan sát, thay đổi kích cỡ của đối tượng,… Chi tiết được trình bày ở mục 2.3 dưới đây.

2. VẼ HÌNH DỰA TRÊN DỮ LIỆU KIỂU WIREFRAME VỚI CÁC PHÉP CHIẾU

Vẽ một đối tượng ở dạng mô hình khung thì một cách đơn giản là chúng ta vẽ các cạnh trong danh sách. Song, để vẽ một cạnh trong không gian 3 chiều lên mặt phẳng 2 chiều thì chúng ta cần thực hiện các phép chiếu để chiếu chúng lên mặt phẳng chiếu, hay nói một cách cụ thể là chúng ta cần chiếu các điểm đầu mút của các cạnh lên mặt phẳng chiếu rồi sau đó nối chúng lại.

Kỹ thuật chung để vẽ một đoạn thẳng 3D:

- Chiếu mỗi điểm đầu mút thành từng điểm 2 chiều.
- Vẽ đoạn thẳng nối hai điểm chiếu chúng ta được đoạn thẳng chiếu.

Điều này làm được, bởi vì các phép chiếu nói chung bảo toàn đường thẳng. Vấn đề còn lại là đi tìm hình chiếu của một điểm trong không gian 3 chiều lên mặt phẳng chiếu. Chúng ta xem xét hai phép chiếu dưới đây:

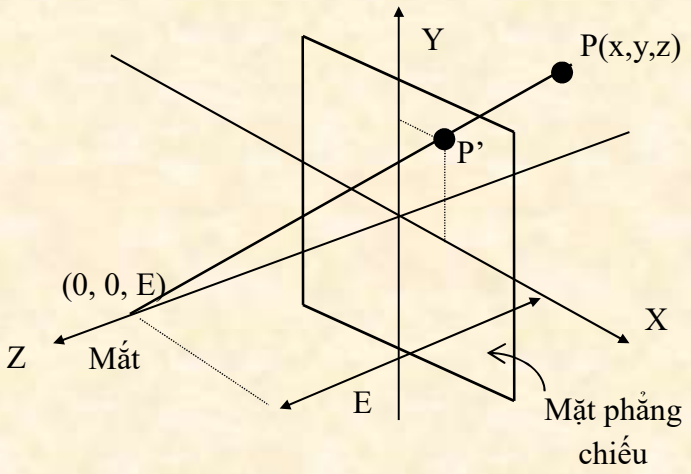
2.1. Phép chiếu trực giao đơn giản

Chiếu một điểm P(x,y,z) lên một mặt phẳng thì đơn giản nhất là chọn mặt phẳng chiếu là mặt phẳng tọa độ. Ví dụ như chiếu lên mặt phẳng OXY, khi đó chúng ta chỉ việc bỏ đi thành phần z trong tọa độ của điểm và giữ nguyên các thành phần x,y chúng ta được tọa độ của điểm chiếu trên mặt phẳng OXY.

2.2. Phép chiếu phối cảnh đơn giản

Phép chiếu phối cảnh được thực hiện khá đơn giản, song nó cho chúng ta một cái nhìn khá tự nhiên hơn về đối tượng 3D so với phép chiếu trực giao. Chúng ta đã làm quen với cách nhìn phối cảnh trong đời sống hàng ngày.

Phép chiếu phối cảnh phụ thuộc vào vị trí tương đối của mắt và mặt phẳng chiếu. Ví dụ như trong Hình 5.3:



Hình 5.3. Cách bố trí một phép chiếu phối cảnh đơn giản

Như đã trình bày trong chương IV (Các phép biến đổi hình học) chúng ta có hình chiếu phối cảnh P' của P lên mặt phẳng OXY có các thành phần tọa độ là:

$$x' = \frac{x}{1 - \frac{z}{E}} \text{ và } y' = \frac{y}{1 - \frac{z}{E}}$$

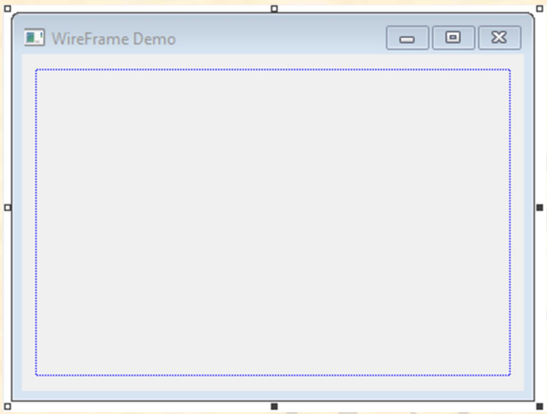
Dễ dàng thấy rằng phép chiếu phối cảnh gần giống với phép chiếu trực giao trước đây, ngoại trừ x và y được nhân một hệ số t=1/(1–z/E). Hệ số này thụ thuộc vào khoảng cách tương đối E giữa mắt nhìn và mặt phẳng chiếu, và nó sẽ làm cho vật thể (hay thành phần) ở gần mắt thì có hình dáng to ra, còn những vật thể (hay thành phần) ở xa thì có hình dáng nhỏ lại.

2.3. Cài đặt thực nghiệm cho mô hình wireframe

Chúng ta cần kết hợp phần kiến thức ở chương 4, mục 4. "Quan sát vật thể 3 chiều và quay hệ quan sát", với nội dung của chương này. Từ đó, xây dựng một chương trình cho phép mô phỏng việc quan sát một vật thể 3 chiều. Chương trình cho phép người sử dụng quan sát vật thể dưới nhiều góc độ khác nhau thông qua thao tác rê chuột. Sau đây là các bước thiết kế chương trình:

Bài thực nghiệm số 3:

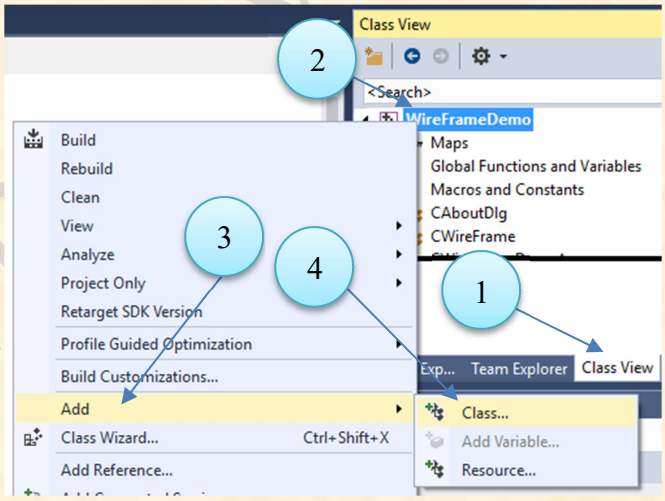
- Bước 1: Tạo một chương trình dạng Dialog based như hướng dẫn ở mục 2.3 với tên gọi WireFrameDemo.
- Bước 2: Thiết kế giao diện như hình sau:



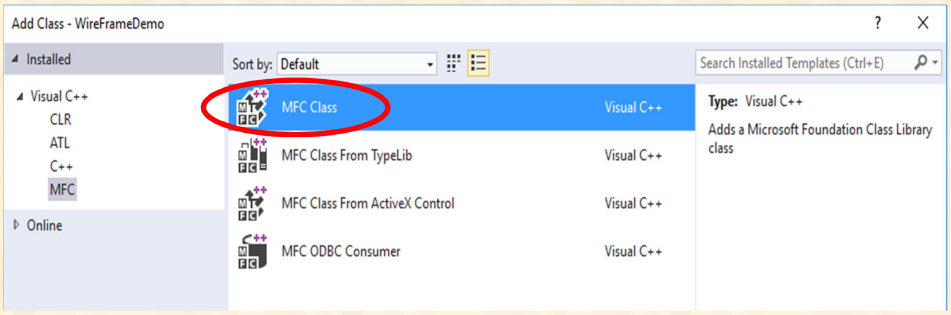
Hình 5.4. Giao diện chương trình WireFrameDemo

- Bước 3: Tạo một MFC Class với tên gọi CWireFrame:

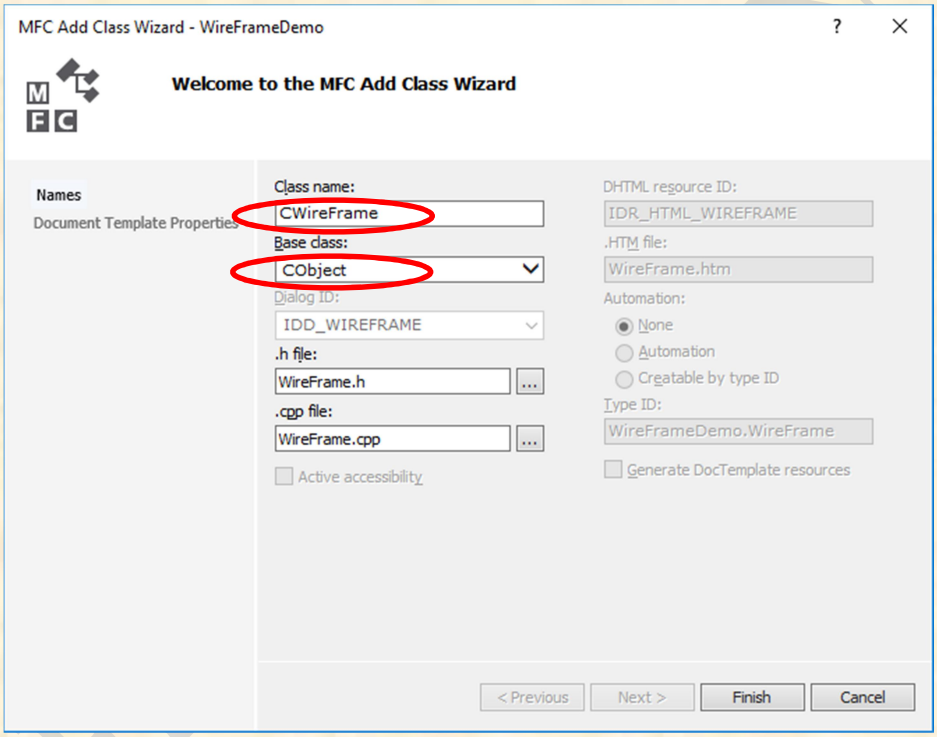
Các bước thực hiện được mô tả qua các hình ảnh dưới đây:



Hình 5.5. Menu ngữ cảnh cho phép tạo một class cho project



Hình 5.6. Tạo một MFC Class



Hình 5.7. Thiết lập tham số cho lớp CWireFrame

+ File header của lớp CWireFrame được khai báo với các nội dung chính như dưới đây:

```
struct Vertex    // Lưu trữ giá trị tọa độ một đỉnh của đối tượng
{
   double x, y, z;
};

struct  Edge     /* Lưu trữ thông tin một cạnh, bao gồm 2 giá trị
```

```
chỉ mục đỉnh */
{
   int VertexIndex1;
   int VertexIndex2;
};

class CWireFrame : public CObject
{
public:
   CWireFrame();             // Hàm khởi dựng đối tượng
   virtual ~CWireFrame();// Hàm hủy đối tượng
   int NumVertex;               // Số đỉnh của đối tượng
   Vertex *ListVertex;       // Danh sách đỉnh (được cấp phát động)
   int NumEdge;              // Số cạnh của đối tượng
   Edge *ListEdge;           // Danh sách cạnh (được cấp phát động)
   double Zoom;// Hệ số co giãn (phóng lớn hay thu nhỏ) đối tượng

   Vertex *ListVertexInViewCoordinates = NULL; /* Danh sách chứa
tọa độ của đối tượng đo trong hệ tọa độ người quan sát. Được cấp
phát động, khi không cần thiết có thể giải phóng bộ nhớ nhưng cần
nhớ gán lại giá trị NULL sau khi giải phóng */

   void CreateChair();   /* Khởi tạo thông tin cho đối tượng, mô tả
một chiếc ghế */
   void TransVertexToViewCoordinates(unsigned int R, unsigned int
DeltaAngle, unsigned int PhiAngle); /* Hàm chuyển đổi số đo các đỉnh
của đối tượng từ hệ tọa độ cục bộ sang hệ tọa độ người quan sát, dữ
liệu được  chứa vào danh sách ListVertexInViewCoordinates */
   void DrawObject(CDC *p_DC, CRect R);   /* Hàm vẽ đối tượng trong
hệ tọa độ  người quan sát lên màn hình */

private:
   double LUT_Cos[360];  // Bảng tra giá trị hàm Cos
   double LUT_Sin[360];  // Bảng tra giá trị hàm Sin
};
```

+ File source (mã nguồn) của lớp CWireFrame được mã hóa như sau:

```
#include "stdafx.h"
#include "WireFrameDemo.h"
#include "WireFrame.h"
```

```
const double PI = 3.14159265358979323846;

// CWireFrame

CWireFrame::CWireFrame()
{
   for (unsigned int i = 0; i<360; i++)
   {
      double Angle = i*PI / 180;
      this->LUT_Cos[i] = cos(Angle);
      this->LUT_Sin[i] = sin(Angle);
   }
}

CWireFrame::~CWireFrame()
{
   delete[] ListVertex;
   delete[] ListEdge;

   if (ListVertexInViewCoordinates != NULL)
   {
      delete[]ListVertexInViewCoordinates;
   }
}

void CWireFrame::CreateChair()
{
   NumVertex = 12;
   ListVertex = new Vertex[NumVertex];
   ListVertex[0].x= 1;ListVertex[0].y = 1;  ListVertex[0].z =-3;
   ListVertex[1].x=-1;ListVertex[1].y = 1;  ListVertex[1].z =-3;
   ListVertex[2].x=-1;ListVertex[2].y =-1;  ListVertex[2].z =-3;
   ListVertex[3].x= 1;ListVertex[3].y =-1;  ListVertex[3].z =-3;
   ListVertex[4].x= 1;ListVertex[4].y = 1;  ListVertex[4].z = 0;
   ListVertex[5].x=-1;ListVertex[5].y = 1;  ListVertex[5].z = 0;
   ListVertex[6].x=-1;ListVertex[6].y =-1;  ListVertex[6].z = 0;
   ListVertex[7].x= 1;ListVertex[7].y =-1;  ListVertex[7].z = 0;
   ListVertex[8].x= 1;ListVertex[8].y =-1;  ListVertex[8].z = 2;
   ListVertex[9].x=-1;ListVertex[9].y =-1;  ListVertex[9].z = 2;
   ListVertex[10].x= 1;  ListVertex[10].y=-1;  ListVertex[10].z =
3;
```

```
   ListVertex[11].x=-1;  ListVertex[11].y=-1;  ListVertex[11].z =
3;

   NumEdge = 10;
   ListEdge = new Edge[NumEdge];
   ListEdge[0].VertexIndex1 = 0; ListEdge[0].VertexIndex2 = 4;
   ListEdge[1].VertexIndex1 = 1; ListEdge[1].VertexIndex2 = 5;
   ListEdge[2].VertexIndex1 = 2; ListEdge[2].VertexIndex2 = 11;
   ListEdge[3].VertexIndex1 = 3; ListEdge[3].VertexIndex2 = 10;
   ListEdge[4].VertexIndex1 = 4; ListEdge[4].VertexIndex2 = 5;
   ListEdge[5].VertexIndex1 = 5; ListEdge[5].VertexIndex2 = 6;
   ListEdge[6].VertexIndex1 = 6; ListEdge[6].VertexIndex2 = 7;
   ListEdge[7].VertexIndex1 = 7; ListEdge[7].VertexIndex2 = 4;
   ListEdge[8].VertexIndex1 = 8; ListEdge[8].VertexIndex2 = 9;
   ListEdge[9].VertexIndex1 = 10; ListEdge[9].VertexIndex2 = 11;
   Zoom = 30;
}
void CWireFrame::TransVertexToViewCoordinates(unsigned int R,
unsigned int DeltaAngle, unsigned int PhiAngle)
{
   if (ListVertexInViewCoordinates == NULL)
   {
      this->ListVertexInViewCoordinates = new Vertex[this-
>NumVertex];
   }

   double x, y, z;

   for (unsigned int i = 0; i<this->NumVertex; i++)
   {
      x = ListVertex[i].x*this->Zoom;
      y = ListVertex[i].y*this->Zoom;
      z = ListVertex[i].z*this->Zoom;

      ListVertexInViewCoordinates[i].x =
      -x*LUT_Sin[DeltaAngle] + y*LUT_Cos[DeltaAngle];

      ListVertexInViewCoordinates[i].y =
      -x*LUT_Cos[DeltaAngle] * LUT_Sin[PhiAngle] -
y*LUT_Sin[DeltaAngle] * LUT_Sin[PhiAngle] + z*LUT_Cos[PhiAngle];

      ListVertexInViewCoordinates[i].z = -x*LUT_Cos[DeltaAngle] *
```

```
            LUT_Cos[PhiAngle] - y*LUT_Sin[DeltaAngle] *
LUT_Cos[PhiAngle] - z*LUT_Sin[PhiAngle] + R;
    };
}

void CWireFrame::DrawObject(CDC *p_DC, CRect R)
{
    long X_C = R.CenterPoint().x;
    long Y_C = R.CenterPoint().y;
    int x1, y1, x2, y2;

    CPen RedPen(PS_SOLID, 1, RGB(255, 0, 0));
    CPen* OldPen = p_DC->SelectObject(&RedPen);

    for (unsigned int i = 0; i < this->NumEdge; i++)
    {
        x1 =
int(ListVertexInViewCoordinates[ListEdge[i].VertexIndex1].x + 0.5)
+ X_C;
        y1 =-
int(ListVertexInViewCoordinates[ListEdge[i].VertexIndex1].y + 0.5)
+ Y_C; // Đảo chiều trục Y rồi tịnh tiến đến tâm của vùng R

        x2 =
int(ListVertexInViewCoordinates[ListEdge[i].VertexIndex2].x + 0.5)
+ X_C;
        y2 =-
int(ListVertexInViewCoordinates[ListEdge[i].VertexIndex2].y + 0.5)
+ Y_C;

        p_DC->MoveTo(x1, y1);
        p_DC->LineTo(x2, y2);
    }

    p_DC->SelectObject(OldPen);
}
```

❖ Bước 4: Lập trình xử lý các sự kiện chuột nhằm thể hiện hình ảnh của đối tượng (*là chiếc ghế*) dưới góc nhìn của người quan sát:

+ Các khai báo ***cần bổ sung*** trong file WireFrameDemoDlg.h là:

```
1. #include "WireFrame.h" /* Thêm khai báo để có thể sử dụng lớp CWireFrame */

2. public:

   CWireFrame WF_Object; /* Khai báo biến đối tượng thuộc lớn WireFrame để lưu trữ và xử lý với đối tượng 3 chiều theo mô hình wireframe */
   int R=50, DeltaAngle=30, PhiAngle=35; /* Khai báo các thông số: khoảng cách (R), góc θ (DeltaAngle), và góc ϕ (PhiAngle) */

   bool MouseDown=false; // Báo hiệu phím trái chuột có được bấm xuống?
   CPoint Position_MouseDown; /* Lưu trữ vị trí tọa độ (Position) trên cửa sổ mà tại đó người sử dụng đã bấm giữ nút chuột trái */
   int MouseDown_DeltaAngle, MouseDown_PhiAngle; /* Lưu giữ giá trị các góc θ và ϕ tại thời điểm bắt đầu bấm nút trái chuột, phục vụ cho việc tính toán giá trị các góc θ và ϕ mới khi di chuyển chuột (mouse move) */
```

+ Add một hàm có dạng `void ViewObject();` vào trong lớp CWireFrameDemoDlg nhằm cho phép thể hiện hình ảnh của đối tượng lên cửa sổ chương trình. Mã lệnh như sau:

```
void CWireFrameDemoDlg::ViewObject()
{
   CDC *p_DC = this->GetDC(); /*Lấy device context của cửa sổ chương trình */
   CRect Rect;
   this->GetClientRect(&Rect); /* Lấy tọa độ của vùng không gian bên trong cửa sổ phục vụ cho việc vẽ đối tượng */

   WF_Object.TransVertexToViewCoordinates(R, DeltaAngle, PhiAngle); /* Chuyển số đo của đối tượng từ hệ tọa độ cục bộ sang hệ quan sát */
   p_DC->FillSolidRect(Rect, RGB(255, 255, 255)); /* Xóa trắng cửa sổ bằng màu trăng */

   WF_Object.DrawObject(p_DC, Rect); // Vẽ đối tượng lên cửa sổ
}
```

+ Thêm đoạn mã lệnh sau vào cuối hàm đáp ứng sự kiện `CWireFrameDemoDlg::OnInitDialog()` như sau:

```
BOOL CWireFrameDemoDlg::OnInitDialog()
{
   ... /* Các câu lệnh mặc định có sẵn trong hàm OnInitDialog() do trình Wizard tạo ra */

   // TODO: Add extra initialization here
   WF_Object.CreateChair(); /* Tạo thông tin mô tả về chiếc ghế tựa cho biến đối tượng WF_Object */

   ViewObject(); // Thể hiện đối tượng lên trên cửa sổ chương trình

   return TRUE;
}
```

Thêm đoạn mã lệnh sau vào cuối hàm đáp ứng sự kiện `CWireFrameDemoDlg:: OnPaint` như sau:

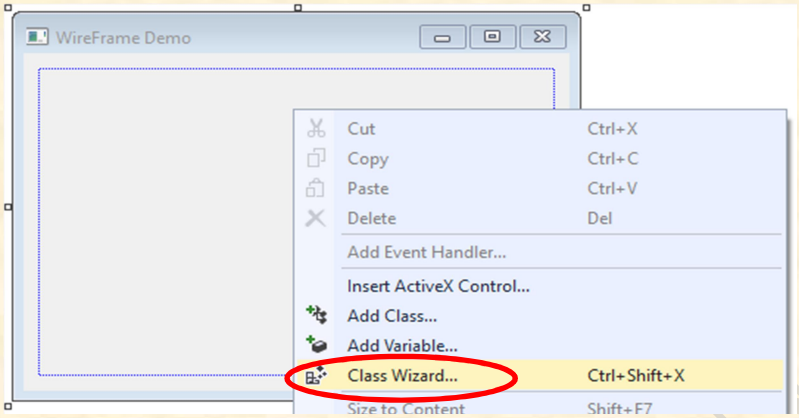
```
void CWireFrameDemoDlg::OnPaint()
{

   ... /* Các lệnh mặc định có sẵn trong hàm OnPaint() do trình Wizard tạo ra */

   ViewObject(); // Câu lệnh thêm vào cuối nhằm vẽ đối tượng lên cửa sổ
}
```

+ Thêm 3 hàm xử lý sự kiện chuột trên cửa sổ giao diện gồm:

- WM_LBUTTONDOWN
- WM_LBUTTONUP
- WM_MOUSEMOVE.

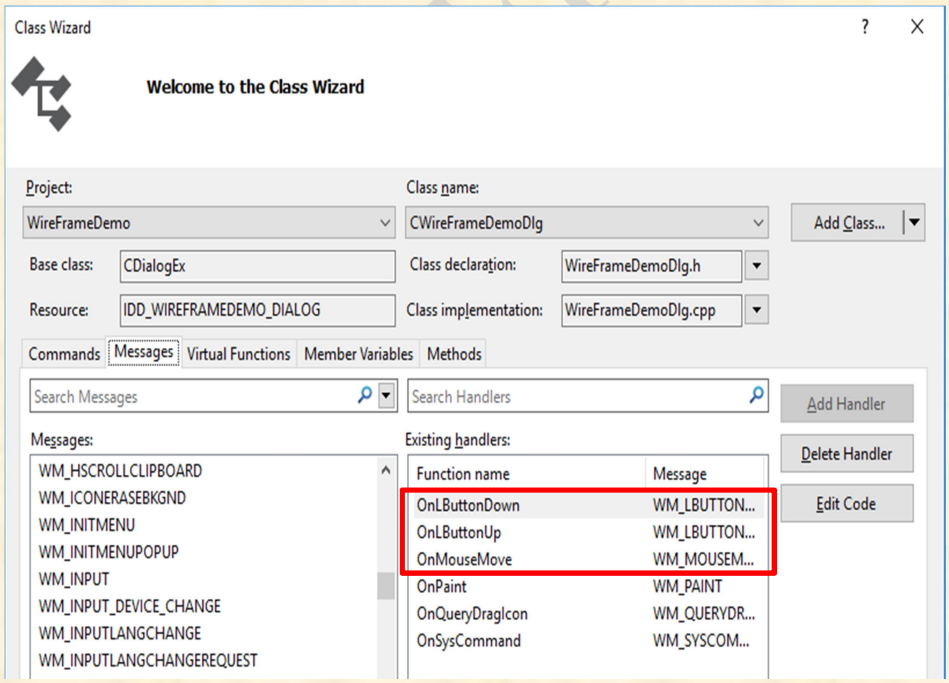
Để thực hiện cần tiến hành theo các bước sau:

- Click chuột phải trên cửa sổ chính của chương trình để làm xuất hiện menu ngữ cảnh, kế đến chọn mục “Class Wizard…”:



Hình 5.8. Thực hiện Class Wizard với lớp CWireFrameDemoDlg để thêm các sự kiện

- Chọn Messages Tab, tiếp đến cần tìm kiếm và add hàm xử lý cho các sự kiện chuột gồm: WM_LBUTTONDOWN, WM_LBUTTONUP, WM_MOUSEMOVE như trong hình vẽ dưới đây:

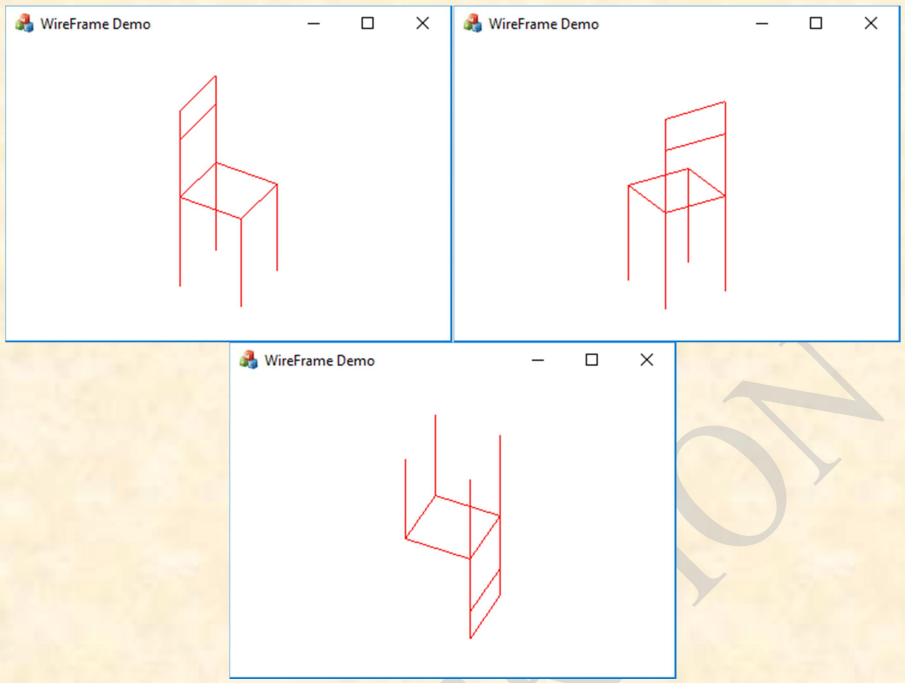


Hình 5.9. Thêm các hàm xử lý sự kiện chuột trên cửa sổ chính của chương trình

- Viết mã lệnh cho các hàm đáp ứng sự kiện như sau:

```
void CWireFrameDemoDlg::OnLButtonDown(UINT nFlags, CPoint point)
{
   MouseDown = true;
   Position_MouseDown = point;
   MouseDown_DeltaAngle = DeltaAngle;
   MouseDown_PhiAngle = PhiAngle;
}

void CWireFrameDemoDlg::OnLButtonUp(UINT nFlags, CPoint point)
{
   MouseDown = false;
}

void CWireFrameDemoDlg::OnMouseMove(UINT nFlags, CPoint point)
{
   if (MouseDown)
   {
      int Del_X = point.x - Position_MouseDown.x;
      int Del_Y = point.y - Position_MouseDown.y;

      DeltaAngle = MouseDown_DeltaAngle - Del_X;
      PhiAngle = MouseDown_PhiAngle + Del_Y;
      DeltaAngle = DeltaAngle % 360;
      PhiAngle = PhiAngle % 360;

      if (DeltaAngle < 0) DeltaAngle += 360;
      if (PhiAngle < 0)   PhiAngle += 360;

      ViewObject();
   }
}
```


❖ Bước 5: Biên dịch và chạy thực nghiệm. Cửa sổ chính của chương trình xuất hiện hình ảnh một chiếc ghế tựa màu đỏ, các thao tác rê chuột trái (gồm bấm giữ nút trái chuột và di chuyển) sẽ kéo theo sự thay đổi góc thể hiện hình ảnh trên cửa sổ, dưới đây là hình minh họa tương ứng với 3 góc nhìn khác nhau mà người sử dụng có được bằng thao tác rê chuột trái:



Hình 5.10. Một số góc quan sát đối tượng được thiết lập thông qua các thao tác rê chuột

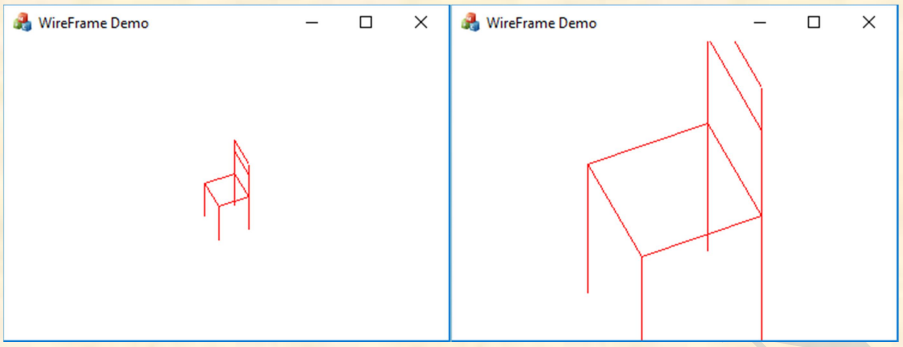
❖ Bước cuối: Nâng cấp chương trình với chức năng phóng to hay thu nhỏ đối tượng thông qua thao tác lăn chuột giữa:

+ Add thêm hàm xử lý sự kiện chuột WM_MOUSEWHEEL

+ Viết mã đáp ứng sự kiện như sau:

```
BOOL CWireFrameDemoDlg::OnMouseWheel(UINT nFlags, short zDelta,
CPoint pt)
{
   WF_Object.Zoom += 0.01*WF_Object.Zoom*(zDelta / WHEEL_DELTA);
   if (WF_Object.Zoom > 100)   WF_Object.Zoom = 100;

   if (WF_Object.Zoom < 0.01)  WF_Object.Zoom = 0.01;

   ViewObject();
   return CDialogEx::OnMouseWheel(nFlags, zDelta, pt);
}
```

Chức năng phóng to thu nhỏ được thể hiện qua hình dưới đây:



Hình 5.11. Đối tượng được thể hiện với các kích cỡ khác nhau

3. BÀI TẬP CUỐI CHƯƠNG

♦ ***Hướng dẫn nâng cao:*** Sinh viên cần nâng cấp chương trình trên bằng việc xây dựng các mô-đun sau:

- Xây dựng hàm đọc thông tin mô tả đối tượng 3 chiều theo mô hình wireframe trong một file text với cấu trúc dạng:
  + Dòng đầu tiên: Chứa tên của đối tượng.
  + Dòng thứ 2: Chứa 2 số nguyên, lần lượt là số đỉnh (gọi mà M) và số cạnh (gọi là N) của đối tượng.
  + Dòng thứ 3 đến dòng thứ (M+2): Mỗi dòng mô tả thông tin tọa độ của một đỉnh, bao gồm 3 số thực tương ứng với 3 thành phần tọa độ x, y, z của đỉnh.
  + Dòng thứ (M+3) đến dòng thứ (M+N+2): Mỗi dòng mô tả thông tin của một cạnh. Thông tin gồm 2 số nguyên tương ứng với thứ tự của hai đỉnh tạo nên cạnh.

Ví dụ, thông tin mô tả chiếc ghế trên sẽ được mô tả trong file text như sau:

```
Hinh cai ghe tua
12 10
1  1   -3
-1 1   -3
```

```
-1 -1  -3
 1 -1  -3
 1  1   0
-1  1   0
-1 -1   0
 1 -1   0
 1 -1   2
-1 -1   2
 1 -1   3
-1 -1   3
 1  5
 2  6
 3  12
 4  11
 5  6
 6  7
 7  8
 8  5
 9  10
 1112
```


- Xây dựng thêm hàm cho phép thực hiện chiếu phối cảnh rồi ánh xạ lên màn hình thay thế cho phép chiếu trực giao đang được sử dụng trong chương trình gốc. Ở đây, chúng ta dùng công thức chiếu phối cảnh đã tìm ra trong chương ***các phép biến đổi hình học*** mục ***Quan sát vật thể 3 chiều và quay hệ quan sát***, đó là công thức:

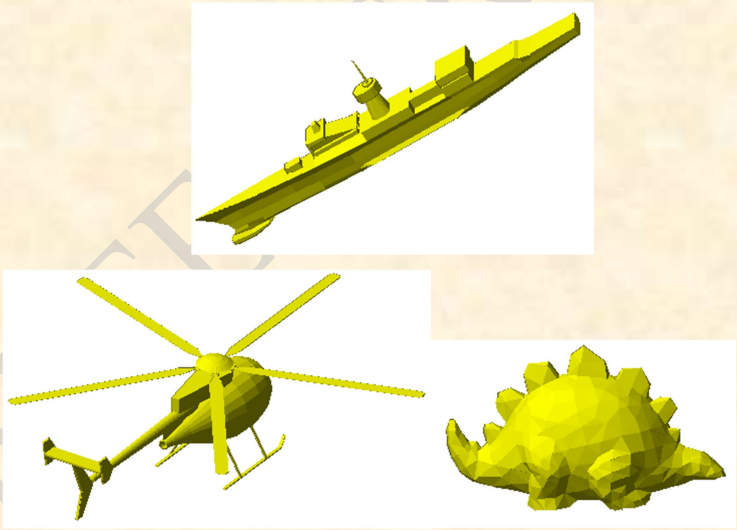
$$x' = x.D/z \quad \text{và} \quad y' = y.D/z$$

- Người sử dụng có thể lựa chọn việc thể hiện hình ảnh lên mặt phẳng chiếu qua phép chiếu trực giao hay chiếu phối cảnh, dựa trên một nút chọn (Radio button) hay hộp kiểm (Check box).

Chương 6

MÔ HÌNH CÁC MẶT ĐA GIÁC VÀ VẤN ĐỀ KHỬ MẶT KHUẤT

Tiếp tục vấn đề mô phỏng đối tượng 3 chiều trên máy tính ở chương trước, trong chương này chúng ta sẽ tìm hiểu một mô hình khá hiệu quả để biểu diễn các đối tượng 3 chiều đó là mô hình các mặt đa giác (Polygon mesh model). Với mô hình các mặt đa giác, các vật thể 3 chiều sẽ được mô tả đầy đủ thông tin hơn về hình dáng và tính chất bề mặt, giúp tạo ra hình ảnh chân thực cho đối tượng. Có thể nói, mô hình các mặt đa giác là mô hình phổ biến nhất được áp dụng trong việc mô phỏng đối tượng 3 chiều trong lĩnh vực đồ họa 3 chiều. Tuy nhiên, mô hình này cũng đặt ra nhiều khó khăn thách thức trong việc giải quyết các vấn đề liên quan như xử lý mặt khuất, chiếu sáng…



Hình 6.1. Hình ảnh của một số đối tượng 3 chiều thể hiện theo mô hình các mặt đa giác

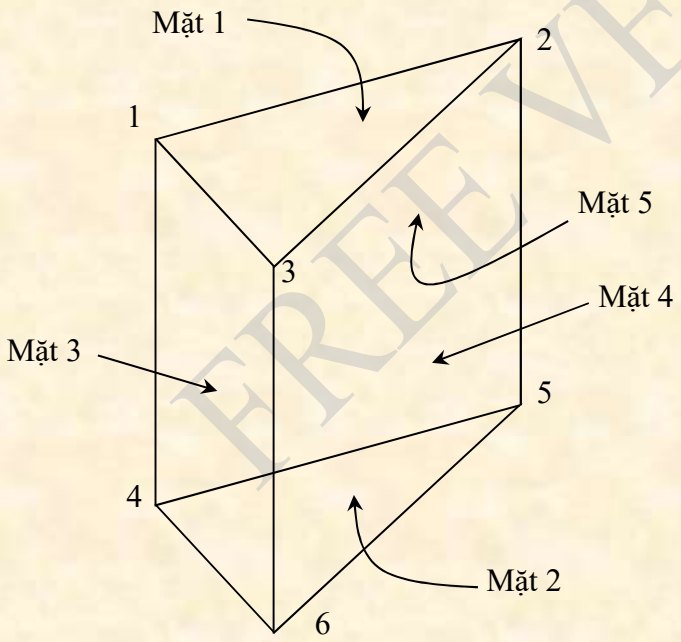
1. MÔ TẢ ĐỐI TƯỢNG 3 CHIỀU BẰNG MÔ HÌNH CÁC MẶT ĐA GIÁC

Một vật thể 3 chiều (3D) có thể biểu diễn trong máy tính bằng nhiều mô hình khác nhau, song hai mô hình phổ biến nhất đó là mô hình khung

dây (WireFrame) và mô hình ***các mặt đa giác*** (Polygon mesh model).

- Mô hình WireFrame: Đã trình bày ở chương trước, nó cho chúng ta hình dáng của vật thể dưới dạng một bộ khung.
- Mô hình các mặt đa giác: Ở đây một vật thể 3D được xác định thông qua các mặt (thay vì các cạnh như trong mô hình WireFrame), và mỗi một mặt lại được xác định thông qua các điểm mà các điểm này được xem như là các đỉnh của mặt đa giác, với mô hình các mặt đa giác thì chúng ta không chỉ tạo ra được hình dáng của vật thể như mô hình Wireframe mà còn thể hiện được các đặc tính về màu sắc và nhiều tính chất khác của vật thể. Song, để có thể mô tả vật thể 3D một cách trung thực (như trong thế giới thực) thì đòi hỏi người lập trình phải tính toán và giả lập nhiều thông tin, mà mấu chốt là vấn đề khử mặt khuất và chiếu sáng. Trong chương này, chúng ta sẽ tập trung nghiên cứu vấn đề khử mặt khuất.

Ví dụ: Mô tả vật thể như trong hình vẽ sau:



Hình 6.2. Minh họa việc số hóa thông tin vật thể 3 chiều theo mô hình các mặt đa giác