Ứng dụng MFC (Visual C++) trong mô phỏng Robot và hệ Cơ điện tử

Bài 11:
Tương tác người dùng với OpenGL

PHẠM MINH QUÂN

mquan.ph@gmail.com

Nội dung

1. Các chế độ của OpenGL
2. Thao tác với chế độ GL_SELECT
   - *2.1. Xác định vùng chọn*
   - *2.2. Đặt tên cho các đối tượng được vẽ*
   - *2.3. Xác định đối tượng nằm trong vùng chọn*
3. Tương tác với tọa độ chuột
   - *3.1. Tìm tọa độ con trỏ chuột trong không gian OpenGL*
   - *3.2. Di chuyển đối tượng theo con trỏ chuột*

1. Các chế độ của OpenGL

❑Các chế độ biểu diễn cơ bản của OpenGL

GLint **glRenderMode**(GLenum mode); // Câu lệnh chuyển đổi giữa các chế độ

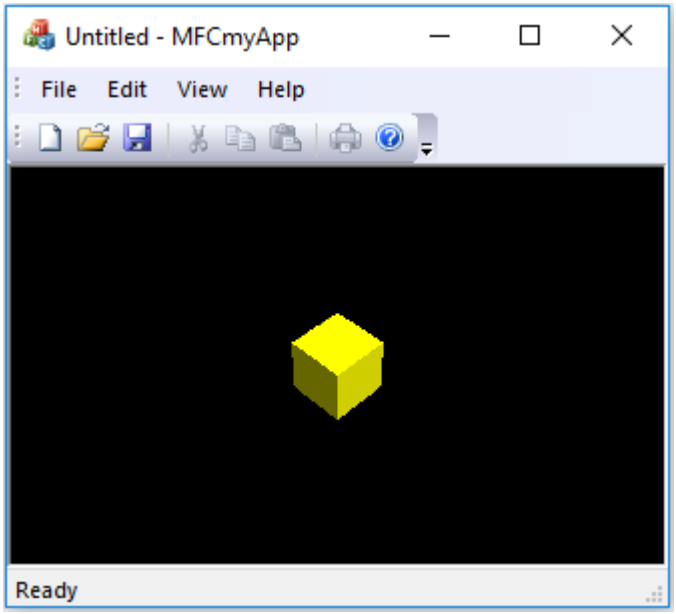
- GL_RENDER : Chế độ vẽ cơ bản (chế độ mặc định)
- GL_SELECT : Trả về các đối tượng đã được vẽ trong chế độ GL_RENDER
- GL_FEEDBACK : Trả về tọa độ và thuộc tính các đỉnh (vertex) đã được vẽ trong chế độ GL_RENDER

2. Thao tác với chế độ GL_SELECT

❑Tạo ứng dụng và cài đặt OpenGL

- Tạo project MFC có kiểu ứng dụng SDI và cài đặt OpenGL vào lớp "C…View" như hướng dẫn trong Bài 10

Kết quả chạy:



2. Thao tác với chế độ GL_SELECT

❑Tạo ứng dụng và cài đặt OpenGL

- Thêm một số thuộc tính vào lớp COpenGLControl và sửa lại hàm OnSize() để thống nhất các tham số phối cảnh khi vẽ trong chế độ GL_RENDER và GL_SELECT sau này.

// File OpenGLControl.h

```
double FoVY;
double Znear;
double Zfar;
```

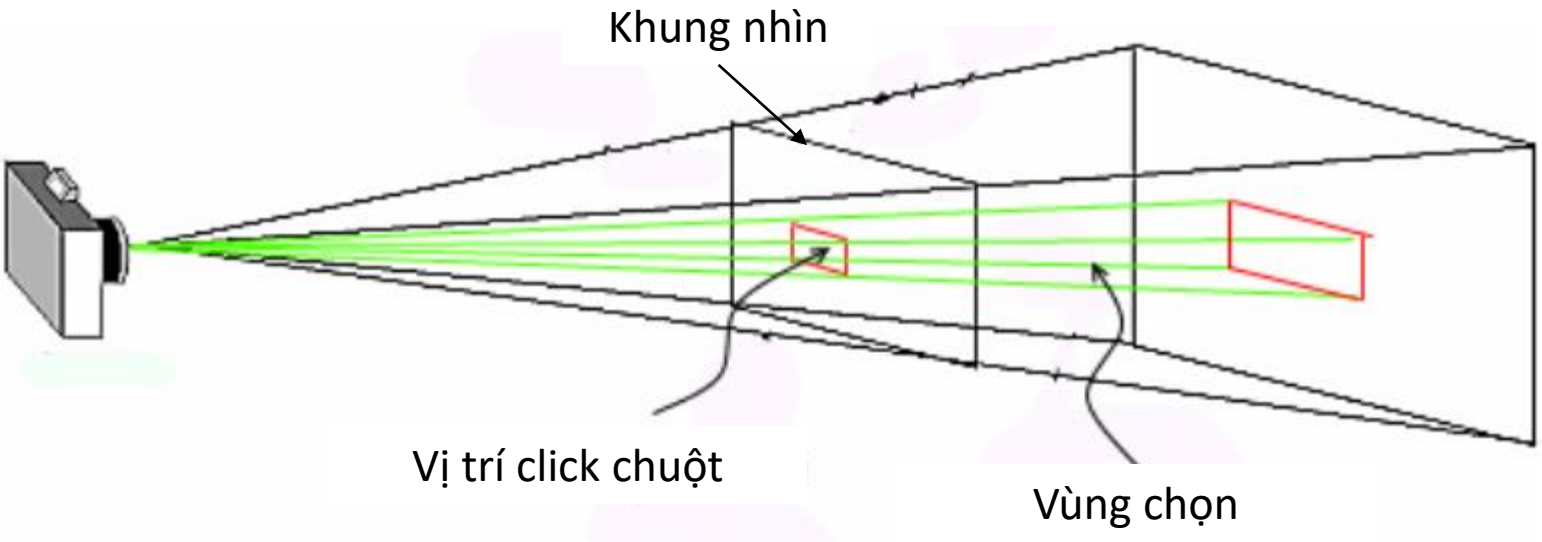
// File OpenGLControl.cpp

```
COpenGLControl::COpenGLControl(void)
{
    ...
    FoVY = 35.0;      // Field of View in Y direction
    Znear = 0.1;
    Zfar = 2000.0;
}


void COpenGLControl::OnSize(UINT nType, int cx, int cy)
{
    ...
    // Set our current view perspective
    gluPerspective(FoVY, (float)cx / (float)cy, Znear, Zfar);
    ...
}
```

2. Thao tác với chế độ GL_SELECT

2.1. Xác định vùng chọn

- Khái niệm vùng chọn



2. Thao tác với chế độ GL_SELECT

2.1. Xác định vùng chọn

- Thêm 1 biến để lưu thông tin về đối tượng (hình khối OpenGL) được chọn

```
int SelectedItem;
```

```
COpenGLControl::COpenGLControl(void)
{
    ...
    SelectedItem = 0;
}
```

- Sửa code hàm oglDrawScene()

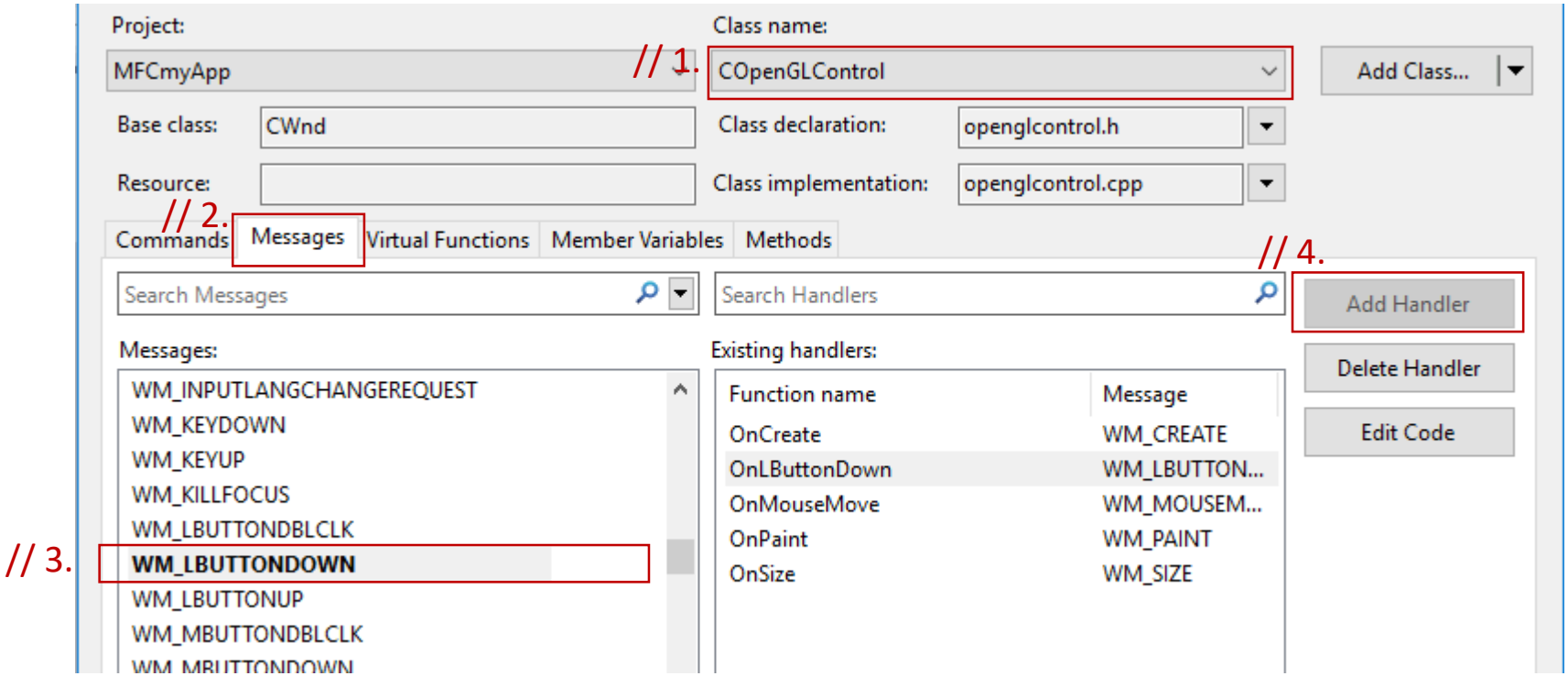
```
void COpenGLControl::oglDrawScene(void)
{
    // Clear color and depth buffer bits
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    GLfloat mat_color_normal[] = { 1.0, 1.0, 0.0 };
    GLfloat mat_color_selected[] = { 1.0, 0.0, 0.0 };
    GLfloat mat_specular[] = { 1.0, 1.0, 1.0 };

    if (SelectedItem) glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, mat_color_selected);
    else glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, mat_color_normal);
    glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
    glMaterialf(GL_FRONT, GL_SHININESS, 30);

    glutSolidCube (1.0);

    // Swap buffers
    SwapBuffers(hdc);
}
```

// Vẽ khối lập phương.
Khi khối không được chọn thì tô màu vàng.
Khi khối được chọn thì tô màu đỏ.

2. Thao tác với chế độ GL_SELECT

2.1. Xác định vùng chọn

- Thêm hàm sự kiện OnLButtonDown() vào lớp COpenGLControl. Hàm này được gọi tới khi có sự kiện Click chuột trái.



2. Thao tác với chế độ GL_SELECT

2.1. Xác định vùng chọn

- Viết code cho hàm OnLButtonDown() như sau:

```
#define BUFSIZE 512
...
void COpenGLControl::OnLButtonDown(UINT nFlags, CPoint point)
{
    // TODO: Add your message handler code here and/or call default
    GLuint selectBuf[BUFSIZE];
    GLint hits;
    GLint viewport[4];
    glGetIntegerv (GL_VIEWPORT, viewport);
    glSelectBuffer (BUFSIZE, selectBuf);
    glRenderMode (GL_SELECT); //Bắt đầu chế độ SELECT
    glMatrixMode(GL_PROJECTION);
    glPushMatrix ();
    glLoadIdentity();
    gluPickMatrix ((GLdouble) point.x, (GLdouble) (viewport[3]- point.y), 1.0, 1.0, viewport);
    gluPerspective(FoVY, (float)viewport[2]/(float)viewport[3], Znear, Zfar);
    glMatrixMode(GL_MODELVIEW);
    oglDrawScene();
    glMatrixMode(GL_PROJECTION);
    glPopMatrix ();
    hits = glRenderMode (GL_RENDER); //Trở về chế độ RENDER
    if (hits) SelectedItem = 1;
    else SelectedItem = 0;
    glMatrixMode(GL_MODELVIEW);
    oglDrawScene();

    CWnd::OnLButtonDown(nFlags, point);
}
```

//Do trục Y của windows hướng từ trên xuống còn trục Y của OpenGL hướng từ dưới lên, cho nên cần phép tính trừ này.

//Kích thước vùng được chọn trên màn hình theo đơn vị Pixel

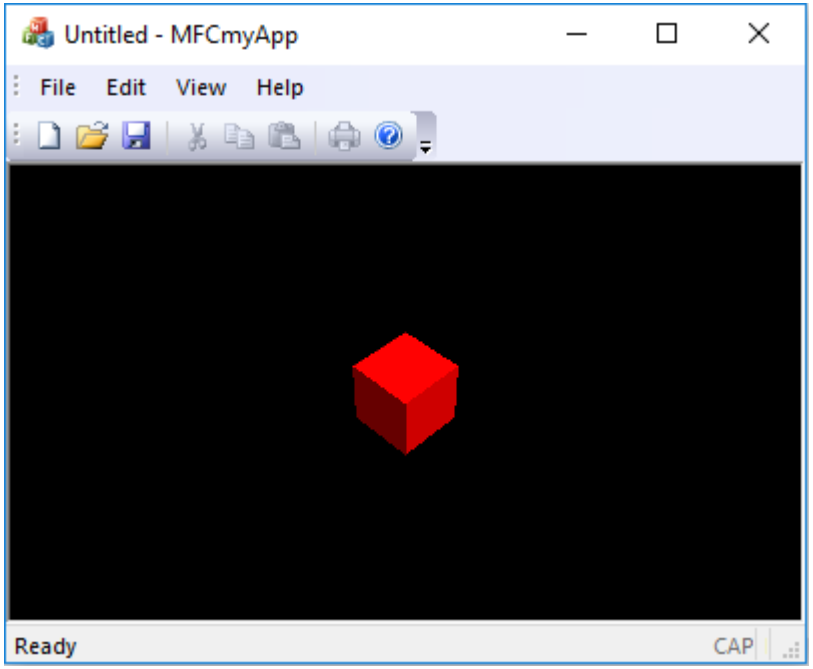
2. Thao tác với chế độ GL_SELECT

2.1. Xác định vùng chọn

Kết quả chạy:

- Khi click chuột trái lên khối hộp -> khối hộp chuyển sang màu đỏ

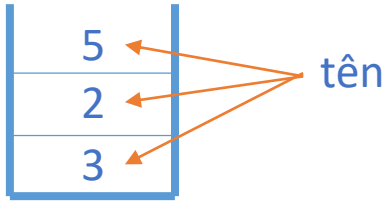
- Khi click chuột trái bên ngoài khối hộp -> khối hộp chuyển sang màu vàng



2. Thao tác với chế độ GL_SELECT

2.2. Đặt tên cho các đối tượng được vẽ

- Khi có nhiều đối tượng được vẽ, cần phải đặt tên để phân biệt:
  - Tên là 1 số nguyên
  - Tên được đặt trong stack và phân cấp theo thứ tự trong stack



*void **glInitNames**(void);*	: Khởi tạo stack lưu tên
*void **glPushName**(GLuint name);*	: Đẩy 1 tên vào stack
*void **glPopName**(GLuint name);*	: Lấy 1 tên ra khỏi stack
*void **glLoadName**(GLuint name);*	: Thay thế tên trên đỉnh stack

2. Thao tác với chế độ GL_SELECT

2.2. Đặt tên cho các đối tượng được vẽ

- Tạo lớp quản lý các hình khối, lớp 'OglObject', như trình bày trong Bài 9.
- Khai báo 2 đối tượng của lớp 'OglObject' làm thuộc tính của lớp COpenGLControl. Thiết lập đối tượng thứ nhất có dạng khối hộp, đối tượng thứ hai là khối cầu.

// File OpenGLControl.h

```
OglObject Obj1;
OglObject Obj2;
```

// File OpenGLControl.cpp

```
COpenGLControl::COpenGLControl(void)
{
    ...
    Obj1.SetType(BOX);
    Obj2.SetType(SPHERE);
}
```

2. Thao tác với chế độ GL_SELECT

2.2. Đặt tên cho các đối tượng được vẽ

- Thêm hàm vẽ để dùng chung cho cả 2 chế độ GL_RENDER và GL_SELECT.

// File OpenGLControl.h

```
    void DrawInMode(GLint mode);
```

// File OpenGLControl.cpp

```
void COpenGLControl::DrawInMode(GLint mode)
{
    glPushMatrix();
    GLfloat mat_color_normal[] = { 1.0, 1.0, 0.0 };
    GLfloat mat_color_selected[] = { 1.0, 0.0, 0.0 };
    GLfloat mat_specular[] = { 1.0, 1.0, 1.0 };
    glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
    glMaterialf(GL_FRONT, GL_SHININESS, 30);

    if (SelectedItem==BOX) glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, mat_color_selected);
    else glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, mat_color_normal);
    if (mode == GL_SELECT) glLoadName(BOX);      // Nếu trong chế độ Select thì đặt tên cho khối hộp là BOX
    Obj1.Draw();

    glTranslatef (3.0, 0.0, 0.0);
    if (SelectedItem==SPHERE) glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, mat_color_selected);
    else glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, mat_color_normal);
    if (mode == GL_SELECT) glLoadName(SPHERE);  // Nếu trong chế độ Select thì đặt tên cho khối cầu là SPHERE
    Obj2.Draw();

    glPopMatrix();
}
```

2. Thao tác với chế độ GL_SELECT

2.2. Đặt tên cho các đối tượng được vẽ

- Sửa code của hàm oglDrawScene().

```
void COpenGLControl::oglDrawScene(void)
{
    // Clear color and depth buffer bits
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    // Draw in GL_RENDER mode
    DrawInMode(GL_RENDER);

    // Swap buffers
    SwapBuffers(hdc);
}
```

2. Thao tác với chế độ GL_SELECT

2.3. Xác định đối tượng nằm trong vùng chọn

- Thêm hàm xử lý dữ liệu trả về từ chế độ GL_SELECT.

// File OpenGLControl.h

```
void ProcessHits (GLint hits, GLuint *buffer);
```

// File OpenGLControl.cpp

```
void COpenGLControl::ProcessHits (GLint hits, GLuint *buffer)
{
    float min_z_min;
    SelectedItem = 0;
    GLuint *ptr = (GLuint *) buffer;
    for (int i = 0; i < hits; i++)
    {
        GLuint names = *ptr; ptr++;
        float z_min = (float) *ptr/0x7fffffff; ptr++;
        float z_max = (float) *ptr/0x7fffffff; ptr++;
        GLuint name = *ptr;
        ptr++;
        if ( i == 0 || min_z_min > z_min )
        {
            min_z_min = z_min;
            SelectedItem = name;
        }
    }
}
```

2. Thao tác với chế độ GL_SELECT

2.3. Xác định đối tượng nằm trong vùng chọn

- Sửa code hàm sự kiện OnLButtonDown():

```
void COpenGLControl::OnLButtonDown(UINT nFlags, CPoint point)
{
    // TODO: Add your message handler code here and/or call default
    GLuint selectBuf[BUFSIZE];
    GLint hits;
    GLint viewport[4];
    glGetIntegerv (GL_VIEWPORT, viewport);
    glSelectBuffer (BUFSIZE, selectBuf);
    glRenderMode (GL_SELECT);
    glInitNames();
    glPushName(0);
    glMatrixMode(GL_PROJECTION);
    glPushMatrix ();
    glLoadIdentity();
    gluPickMatrix ((GLdouble) point.x, (GLdouble) (viewport[3]- point.y), 1.0, 1.0, viewport);
    gluPerspective(FoVY, (float)viewport[2]/(float)viewport[3], Znear, Zfar);
    glMatrixMode(GL_MODELVIEW);
    DrawInMode(GL_SELECT);
    glMatrixMode(GL_PROJECTION);
    glPopMatrix ();
    hits = glRenderMode (GL_RENDER);
    ProcessHits (hits, selectBuf);
    glMatrixMode(GL_MODELVIEW);
    oglDrawScene();

    CWnd::OnLButtonDown(nFlags, point);
}
```

2. Thao tác với chế độ GL_SELECT

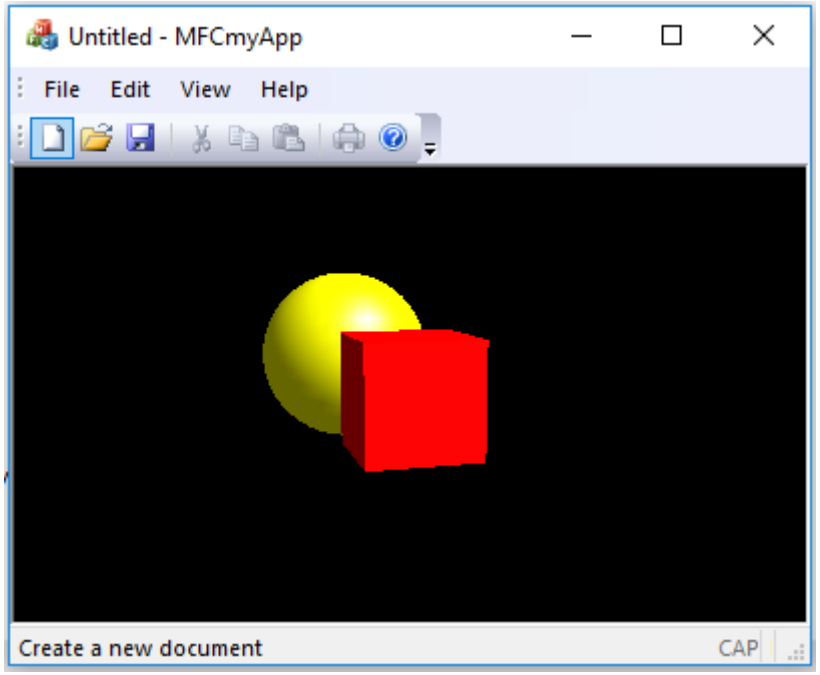
2.3. Xác định đối tượng nằm trong vùng chọn

Kết quả chạy:

- Khi click chuột trái lên khối nào thì khối đó có màu đỏ, khối còn lại màu vàng.

- Khi click bên ngoài cả hai khối thì cả hai có màu vàng.

- Khi 2 khối che khuất nhau thì khối nằm phía trước sẽ chuyển màu đỏ khi được click, khối còn lại giữ màu vàng.



3. Tương tác với tọa độ chuột

❑Chuẩn bị dữ liệu

- ➢ Thêm các thuộc tính chỉ vị trí và các phương thức liên quan vào lớp OglObject

// File OglObject.h

```
class OglObject
{
public:
    OglObject(void);
    ~OglObject(void);
protected:
    int type;
    float x, y, z;
public:
    void SetType(int);
    void SetX(float);
    void SetY(float);
    void SetZ(float);
    void SetPosition(float, float, float);
    float GetX();
    float GetY();
    float GetZ();
    void GetPosition(float &, float &, float &);
    void Draw();
};
```

// File OglObject.cpp

```
OglObject::OglObject(void) {type = BOX; x = 0; y = 0; z = 0;}
OglObject::~OglObject(void) {}

void OglObject::SetType(int tp) {type = tp;}
void OglObject::SetX(float posx) {x = posx;}
void OglObject::SetY(float posy) {y = posy;}
void OglObject::SetZ(float posz) {z = posz;}
void OglObject::SetPosition(float posx, float posy, float posz)
{
    x = posx; y = posy; z = posz;
}
float OglObject::GetX() {return x;}
float OglObject::GetY() {return y;}
float OglObject::GetZ() {return z;}
void OglObject::GetPosition(float &posx, float &posy, float &posz)
{
    posx = x; posy = y; posz = z;
}
void OglObject::Draw()
{
    glPushMatrix();
    glTranslatef(x, y, z);
    switch(type)
    {
    case BOX:
...
```

3. Tương tác với tọa độ chuột

❑Chuẩn bị dữ liệu

- ➢ Thay đổi code của lớp COpenGLControl để phù hợp với cập nhật mới của OglObject

```
COpenGLControl::COpenGLControl(void)        // File OpenGLControl.cpp
{
    ...
    Obj2.SetPosition(3.0, 0.0, 0.0);
}

void COpenGLControl::DrawInMode(GLint mode)
{
    glPushMatrix();
    GLfloat mat_color_normal[] = { 1.0, 1.0, 0.0 };
    GLfloat mat_color_selected[] = { 1.0, 0.0, 0.0 };
    GLfloat mat_specular[] = { 1.0, 1.0, 1.0 };
    glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
    glMaterialf(GL_FRONT, GL_SHININESS, 30);

    if (SelectedItem==BOX) glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, mat_color_selected);
    else glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, mat_color_normal);
    if (mode == GL_SELECT) glLoadName(BOX);
    Obj1.Draw();
    // Xóa dòng lệnh dịch chuyển hệ tọa độ ở đây
    if (SelectedItem==SPHERE) glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, mat_color_selected);
    else glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, mat_color_normal);
    if (mode == GL_SELECT) glLoadName(SPHERE);
    Obj2.Draw();

    glPopMatrix();
}
```

3. Tương tác với tọa độ chuột

❑Chuẩn bị dữ liệu

- ➢ Vẽ thêm các ô lưới trên mặt phẳng XY để dễ hình dung tọa độ 3D

// File OpenGLControl.h

```
    void DrawGridXY();
```

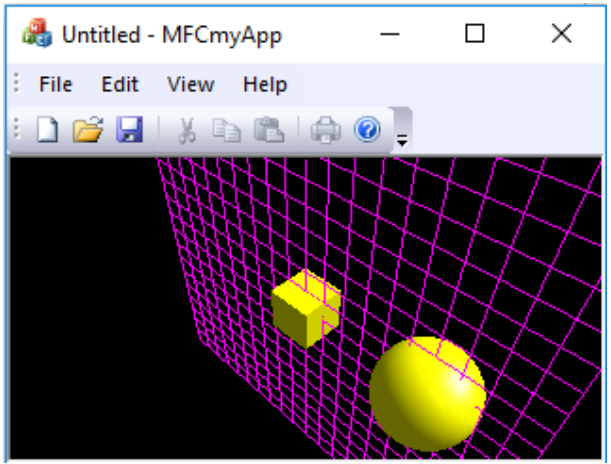
// File OpenGLControl.cpp

```
void COpenGLControl::DrawGridXY()
{
    GLfloat grid_color[] = { 1.0, 0.0, 1.0 };
    glMaterialfv(GL_FRONT, GL_AMBIENT_AND_DIFFUSE, grid_color);
    float step = 0.5;
    int num = 10;
    for (int i = -num; i <= num; i++)
    {
        glBegin(GL_LINES);
        glVertex3f(i*step, -num*step, 0);
        glVertex3f(i*step, num*step, 0);
        glVertex3f(-num*step, i*step, 0);
        glVertex3f(num*step, i*step, 0);
        glEnd();
    }
}
```

// File OpenGLControl.cpp

```
void COpenGLControl::oglDrawScene(void)
{
    // Clear color and depth buffer bits
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    DrawInMode(GL_RENDER);
    DrawGridXY();

    // Swap buffers
    SwapBuffers(hdc);
}
```

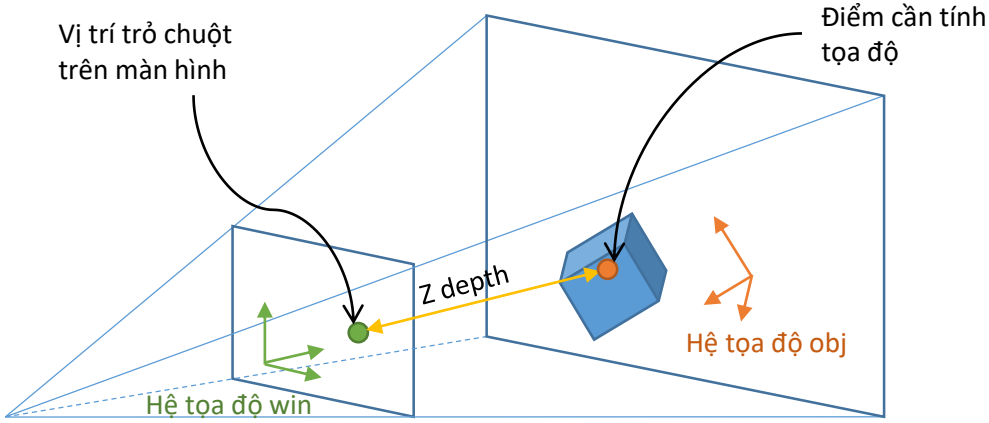
Kết quả chạy:



3. Tương tác với tọa độ chuột

3.1. Tìm tọa độ con trỏ chuột trong không gian OpenGL

- Công thức quy đổi tọa độ của một điểm: từ hệ tọa độ màn hình sang hệ tọa độ toàn cục



GLint **gluUnProject**(GLdouble *winX*,
GLdouble *winY*,
GLdouble *winZ*,
const GLdouble **model*,
const GLdouble **proj*,
const GLint **view*,
GLdouble* *objX*,
GLdouble* *objY*,
GLdouble* *objZ*)

$$\begin{pmatrix} objX \\ objY \\ objZ \\ W \end{pmatrix} = INV(PM) \begin{pmatrix} \frac{2(winX - view[0])}{view[2]} - 1 \\ \frac{2(winY - view[1])}{view[3]} - 1 \\ 2(winZ) - 1 \\ 1 \end{pmatrix}$$

3. Tương tác với tọa độ chuột

3.1. Tìm tọa độ con trỏ chuột trong không gian OpenGL

- Thêm hàm tính chuyển đổi tọa độ

// File OpenGLControl.h

```
void GetMousePositionInOgl(int x, int y, bool cal_zdepth, float &z_depth, double &posX, double &posY, double &posZ);
```

// File OpenGLControl.cpp

```
void COpenGLControl::GetMousePositionInOgl(int x, int y, bool cal_zdepth, float &z_depth,
                                           double &posX, double &posY, double &posZ)
{
    GLint viewport[4];
    GLdouble modelview[16];
    GLdouble projection[16];

    glGetDoublev( GL_MODELVIEW_MATRIX, modelview );
    glGetDoublev( GL_PROJECTION_MATRIX, projection );
    glGetIntegerv( GL_VIEWPORT, viewport );

    if(cal_zdepth)
    {
        float zdepth_tmp;
        glReadPixels(x, viewport[3] - y, 1, 1, GL_DEPTH_COMPONENT, GL_FLOAT, &zdepth_tmp);
        if (zdepth_tmp!=1) z_depth = zdepth_tmp;
    }

    gluUnProject((float)x, (float)viewport[3] - (float)y, z_depth, modelview, projection, viewport, &posX, &posY, &posZ);
}
```

3. Tương tác với tọa độ chuột

3.1. Tìm tọa độ con trỏ chuột trong không gian OpenGL

- Thêm các biến lưu thông tin tọa độ con trỏ chuột trong không gian 3D

// File OpenGLControl.h

```
float m_LastMouseZDepth;
double m_LastMousePosX, m_LastMousePosY, m_LastMousePosZ;
```

- Thêm code vào hàm OnLButtonDown()

// File OpenGLControl.cpp

```
void COpenGLControl::OnLButtonDown(UINT nFlags, CPoint point)
{
    ...
    GetMousePositionInOgl(point.x, point.y, 1, m_LastMouseZDepth, m_LastMousePosX, m_LastMousePosY, m_LastMousePosZ);
    ...
}
```

3. Tương tác với tọa độ chuột

3.2. Di chuyển đối tượng theo con trỏ chuột

- Sửa code hàm OnMouseMove()

```
void COpenGLControl::OnMouseMove(UINT nFlags, CPoint point)
{
    int diffX = (int)(point.x - m_fLastX);
    int diffY = (int)(point.y - m_fLastY);
    m_fLastX  = (float)point.x;
    m_fLastY  = (float)point.y;

     // Left mouse button
     if (nFlags & MK_LBUTTON)
     {
         if (!SelectedItem) // No item is selected
         {
             m_fRotX += (float)0.5f * diffY;
             if ((m_fRotX > 360.0f) || (m_fRotX < -360.0f))
                 m_fRotX = 0.0f;
             m_fRotY += (float)0.5f * diffX;
             if ((m_fRotY > 360.0f) || (m_fRotY < -360.0f))
                 m_fRotY = 0.0f;
         }
         else
         {
             double MousePosX, MousePosY, MousePosZ;
             GetMousePositionInOgl(point.x, point.y, 0,
                 m_LastMouseZDepth, MousePosX, MousePosY,
                 MousePosZ);
             ...
             switch (SelectedItem)
             {
             case BOX: // The box is selected
                 {
                     Obj1.SetX(Obj1.GetX() + MousePosX-m_LastMousePosX);
                     Obj1.SetY(Obj1.GetY() + MousePosY-m_LastMousePosY);
                     Obj1.SetZ(Obj1.GetZ() + MousePosZ-m_LastMousePosZ);
                     break;
                 }
             case SPHERE: // The sphere is selected
                 {
                     Obj2.SetX(Obj2.GetX() + MousePosX-m_LastMousePosX);
                     Obj2.SetY(Obj2.GetY() + MousePosY-m_LastMousePosY);
                     Obj2.SetZ(Obj2.GetZ() + MousePosZ-m_LastMousePosZ);
                     break;
                 }
             }
             m_LastMousePosX = MousePosX;
             m_LastMousePosY = MousePosY;
             m_LastMousePosZ = MousePosZ;
         }
     }
    // Right mouse button
    else if (nFlags & MK_RBUTTON)
    ...
```

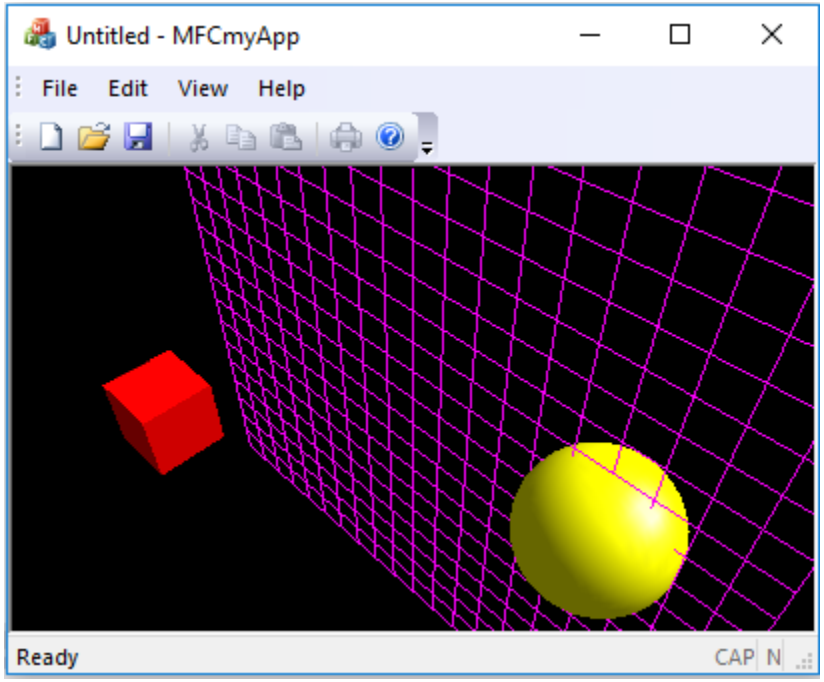
3. Tương tác với tọa độ chuột

3.2. Di chuyển đối tượng theo con trỏ chuột

Kết quả chạy:

- Khi nhấn, giữ chuột trái lên một khối hình và di chuyển chuột thì khối hình cũng bám theo vị trí chuột.

- Chuyển động của khối theo phương song song với màn hình.



3. Tương tác với tọa độ chuột

3.2. Di chuyển đối tượng theo con trỏ chuột

- Trong trường hợp muốn khối hình chỉ di chuyển trên một mặt (ví dụ XY) thì sửa code hàm OnMouseMove() như sau:

```
...
double MousePosX, MousePosY, MousePosZ;
GetMousePositionInOgl(point.x, point.y, 1, m_LastMouseZDepth, MousePosX, MousePosY, MousePosZ);
switch (SelectedItem)
{
case BOX: // The box is selected
    {
        Obj1.SetX(Obj1.GetX() + MousePosX-m_LastMousePosX);
        Obj1.SetY(Obj1.GetY() + MousePosY-m_LastMousePosY);
        //Obj1.SetZ(Obj1.GetZ() + MousePosZ-m_LastMousePosZ);
        break;
    }
case SPHERE: // The sphere is selected
    {
        Obj2.SetX(Obj2.GetX() + MousePosX-m_LastMousePosX);
        Obj2.SetY(Obj2.GetY() + MousePosY-m_LastMousePosY);
        //Obj2.SetZ(Obj2.GetZ() + MousePosZ-m_LastMousePosZ);
        break;
    }
}
m_LastMousePosX = MousePosX;
m_LastMousePosY = MousePosY;
m_LastMousePosZ = MousePosZ;
...
```

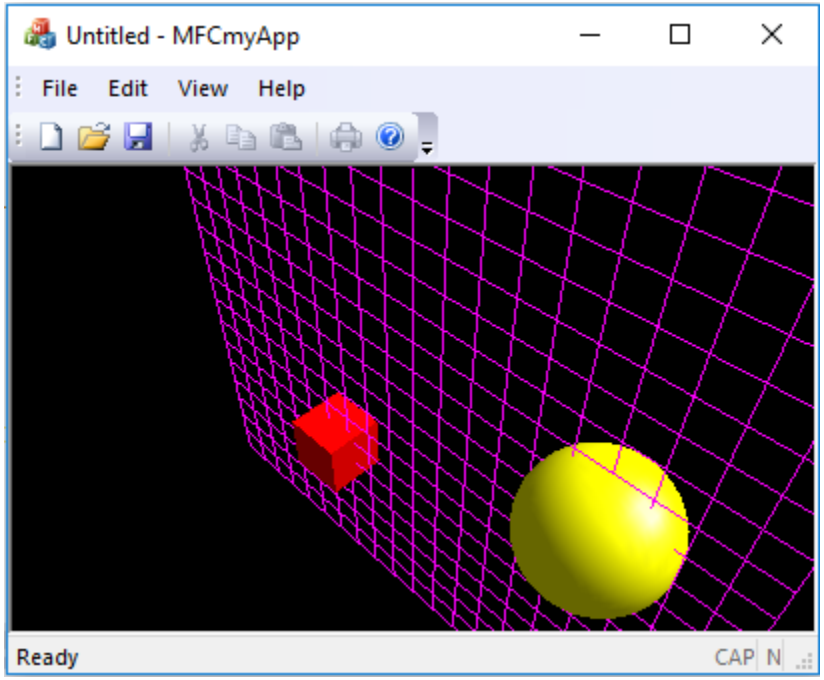
3. Tương tác với tọa độ chuột

3.2. Di chuyển đối tượng theo con trỏ chuột

Kết quả chạy:

- Khi nhấn, giữ chuột trái lên một khối hình và di chuyển chuột thì khối hình cũng bám theo vị trí chuột.

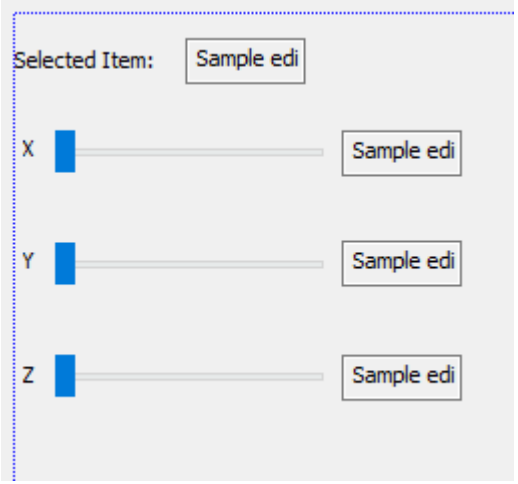
- Chuyển động của khối luôn nằm trên mặt phẳng XY.



3. Tương tác với tọa độ chuột

3.2. Di chuyển đối tượng theo con trỏ chuột

- Thêm 1 ô cửa (dockable pane) gắn vào mainframe như hướng dẫn trong Bài 10. Thiết kế dialog nằm bên trong ô cửa này như dưới đây:



3. Tương tác với tọa độ chuột

3.2. Di chuyển đối tượng theo con trỏ chuột

- Sử dụng các nội dung đã trình bày trong bài 10, tạo kết nối giữa vị trí slider và giá trị hiển thị trên editbox với tọa độ x, y, z của khối hình đang được chọn

Gợi ý :
- Thêm 1 hàm vào lớp …Dlg, có chức năng lấy dữ liệu từ View và hiển thị.

// File ActionDlg.h

```
void DisplayDataFromView();
```

// File ActionDlg.cpp

```
void ActionDlg::DisplayDataFromView()
{
    CMFCmyAppView *pView = CMFCmyAppView::GetView();
    int SliderXpos, SliderYpos, SliderZpos;
    CString strName, strX, strY, strZ;
    switch (pView->m_oglWindow.SelectedItem)
    {
    case 0:
        {
            strName = _T("NONE");
            SliderXpos = 0; SliderYpos = 0; SliderZpos = 0;
            strX = _T("0"); strY = _T("0"); strZ = _T("0");
            break;
        }
    case BOX:
        {
            strName = _T("BOX");
            SliderXpos = pView->m_oglWindow.Obj1.GetX()/SliderStep;
            SliderYpos = pView->m_oglWindow.Obj1.GetY()/SliderStep;
            SliderZpos = pView->m_oglWindow.Obj1.GetZ()/SliderStep;
            …
            strX.Format(NumberFormat, pView->m_oglWindow.Obj1.GetX());
            strY.Format(NumberFormat, pView->m_oglWindow.Obj1.GetY());
            strZ.Format(NumberFormat, pView->m_oglWindow.Obj1.GetZ());
            break;
        }
    case SPHERE:
        {
            strName = _T("SPHERE");
            SliderXpos = pView->m_oglWindow.Obj2.GetX()/SliderStep;
            SliderYpos = pView->m_oglWindow.Obj2.GetY()/SliderStep;
            SliderZpos = pView->m_oglWindow.Obj2.GetZ()/SliderStep;
            strX.Format(NumberFormat, pView->m_oglWindow.Obj2.GetX());
            strY.Format(NumberFormat, pView->m_oglWindow.Obj2.GetY());
            strZ.Format(NumberFormat, pView->m_oglWindow.Obj2.GetZ());
            break;
        }
    }
    SliderX_ctrl.SetPos(SliderXpos);
    SliderY_ctrl.SetPos(SliderYpos);
    SliderZ_ctrl.SetPos(SliderZpos);
    EditX_ctrl.SetWindowTextW(strX);
    EditY_ctrl.SetWindowTextW(strY);
    EditZ_ctrl.SetWindowTextW(strZ);
    EditName_ctrl.SetWindowTextW(strName);
}
```

3. Tương tác với tọa độ chuột

3.2. Di chuyển đối tượng theo con trỏ chuột

- Sử dụng các nội dung đã trình bày trong bài 10, tạo kết nối giữa vị trí slider và giá trị hiển thị trên editbox với tọa độ x, y, z của khối hình đang được chọn

Gợi ý :

- Gọi tới hàm vừa tạo từ các hàm thao tác chuột của lớp OpenGLControl

```
...
ActionPane *pActionPane = ActionPane::GetView(ID_VIEW_ACTIONWINDOW);
pActionPane->m_ActDlg.DisplayDataFromView();
...
```

- Trong trường hợp gặp thông báo lỗi: *'IDD_DIALOG_...' : undeclared identifier* thì cần chú ý khai báo thêm dòng lệnh sau trong file …Dlg.h

```
#include "resource.h"
```

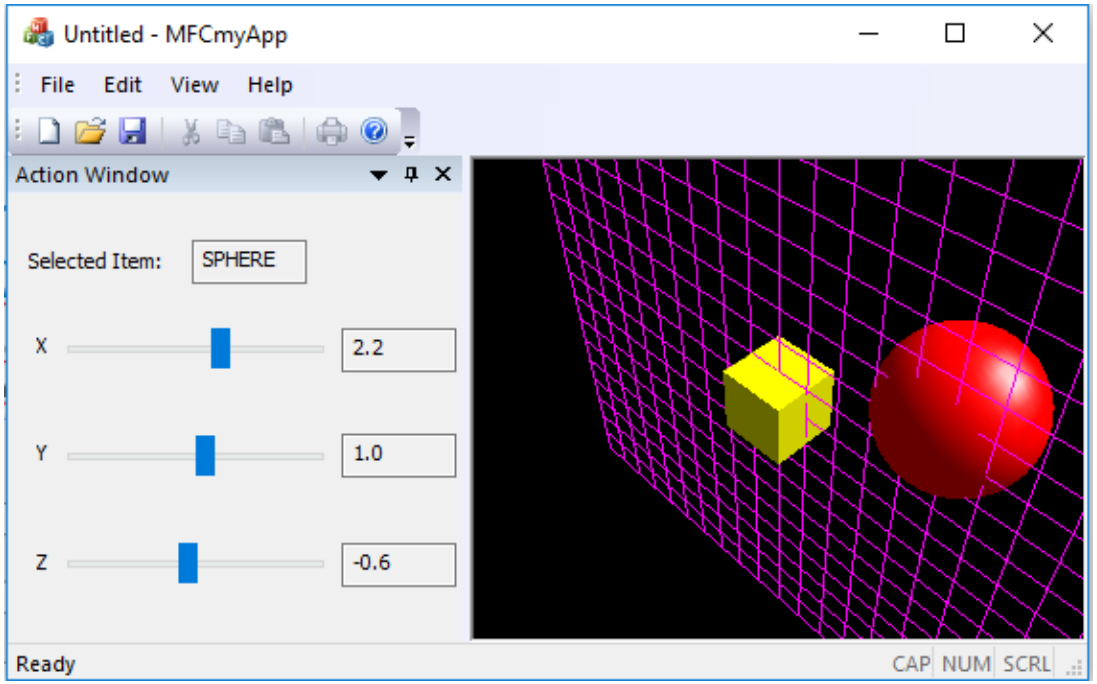
3. Tương tác với tọa độ chuột

3.2. Di chuyển đối tượng theo con trỏ chuột

Kết quả chạy:

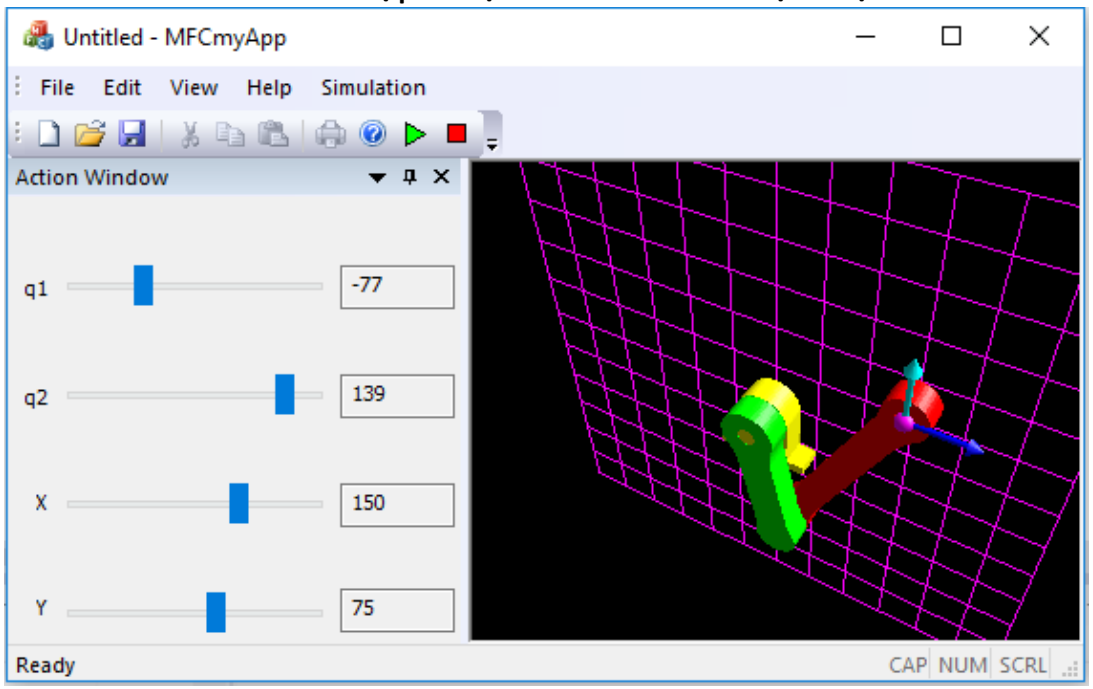
- Khi click chuột trái vào khối hình nào thì tên của khối hình đó được hiển thị trên dialog, đồng thời các slider và editbox thể hiện tọa độ của khối hình đó.

- Khi nhấn giữ chuột trái để di chuyển vị trí khối hình thì các slider và editbox cập nhật tức thời tọa độ của khối hình đó.



Tổng hợp kiến thức

- Áp dụng các nội dung đã trình bày, xây dựng ứng dụng MFC mô phỏng robot 2 bậc tự do. Ở điểm tác động cuối của robot có 1 khối cầu và 2 mũi tên, biểu diễn hệ trục tọa độ Oxy.
  - Khi dùng chuột kéo khối cầu thì điểm tác động cuối di chuyển trên mặt phẳng XY. Khi kéo các mũi tên thì điểm tác động cuối trượt theo phương tương ứng (X, Y).
  - Một ô cửa với các slider sẽ cập nhật tức thời các tọa độ của điểm tác động cuối.



Hết Bài 11