```cpp
#include <windows.h>
#include <Windowsx.h>
#include <gl/gl.h>
#include <gl/glu.h>
#include "matrix4x4f.h"
#include <iostream>
#include <fstream>
#include <vector>

#include "STL_File.h"

using namespace std;

typedef vector<matrix4x4f> matrixarray; // Kiểu mảng của các ma trận matrix4x4f

bool g_shade = true;
bool g_moving = false;

const int K = 3; // Số khâu

int N = 0;  // Số vị trí mô phỏng (số hàng trong các tệp .MAT)

int count = 0; // Biến mô tả vị trí mô phỏng hiện thời

wstring matnames[K] = { // Tên các tệp dữ liệu
    L"tayquay.mat",
    L"thanhtruyen.mat",
    L"contruot.mat",};

string stlnames[K] = {  // Tên các tệp dữ liệu
    "tayquay.stl",
    "thanhtruyen.stl",
    "contruot.stl",};

CSTL_File stl[K]; // Tệp STL cho từng khâu
matrixarray mat[K]; // Tệp MAT (ma trận chuyển vị) cho từng khâu

GLdouble color[3][4] = { // Màu vẽ các khâu
    {1,0,0,0.5},
    {0,1,0,0.5},
    {0,0,1,0.5},};

Vec3D offset[K] = { // Dịch vị trí các hình trong file STL
    Vec3D(200.0000, 200.0000, 500.0000),
    Vec3D(200.0000, 245.0000, 500.0000),
    Vec3D(200.0000, 385.0000, 500.0000)
};

HWND    hWnd;
HDC     hDC;
HGLRC   hRC;

void DrawGrid(int gridSize = 10, float gridElev=0, float lineWidth = 0.5f)
{
    //  Set up some nice attributes for drawing the grid.
    glPushAttrib(GL_LINE_BIT | GL_ENABLE_BIT | GL_COLOR_BUFFER_BIT);
    glEnable(GL_LINE_SMOOTH);
    glEnable(GL_BLEND);
    glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
    glDisable(GL_LIGHTING);
    glHint(GL_LINE_SMOOTH_HINT, GL_NICEST);
    glLineWidth(lineWidth);

    //  Create the grid.

    glBegin(GL_LINES);
    {
        for (int i = -gridSize; i <= gridSize; i++)
        {
```

```
70              glColor4f(0.2f, 0.2f, 0.2f, 0.8f);
71              glVertex3f((float)i, 0, -(float)gridSize);
72              glVertex3f((float)i, 0, +(float)gridSize);
73              glVertex3f(-(float)gridSize, 0, (float)i);
74              glVertex3f(+(float)gridSize, 0, (float)i);
75          }
76      }
77      glEnd();
78
79      glPopAttrib();
80  }
81
82  void DrawAxes(float axisSize = 3.0f, float lineWidth = 2.0f)
83  {
84      const GLfloat xAxizColor[4] = {1, 0, 0, 1};
85      const GLfloat yAxizColor[4] = {0, 1, 0, 1};
86      const GLfloat zAxizColor[4] = {0, 0, 1, 1};
87
88          //  Set up some nice attributes for drawing the grid.
89      glPushAttrib(GL_LINE_BIT | GL_ENABLE_BIT | GL_COLOR_BUFFER_BIT);
90      glEnable(GL_LINE_SMOOTH);
91      glEnable(GL_BLEND);
92      glBlendFunc(GL_SRC_ALPHA, GL_ONE_MINUS_SRC_ALPHA);
93      glDisable(GL_LIGHTING);
94      glHint(GL_LINE_SMOOTH_HINT, GL_NICEST);
95      glLineWidth(lineWidth);
96      //  Create the axies.
97      glBegin(GL_LINES);
98      {
99          glColor4fv(xAxizColor);
100         glVertex3f(0, 0, 0);
101         glVertex3f(axisSize, 0, 0);
102         glColor4fv(yAxizColor);
103         glVertex3f(0, 0, 0);
104         glVertex3f(0, axisSize, 0);
105         glColor4fv(zAxizColor);
106         glVertex3f(0, 0, 0);
107         glVertex3f(0, 0, axisSize);
108     }
109     glEnd();
110
111     glPopAttrib();
112 }
113
114 class MayaCamera
115 {
116 public:
117     float    m_fLastX;
118     float    m_fLastY;
119     float    m_fPosX;
120     float    m_fPosY;
121     float    m_fZoom;
122     float    m_fRotX;
123     float    m_fRotY;
124
125     MayaCamera()
126     {
127         m_fPosX = 0.0f;     // X position of model in camera view
128         m_fPosY = 0.0f;     // Y position of model in camera view
129         m_fZoom = 10.0f;    // Zoom on model in camera view
130         m_fRotX = 0.0f;     // Rotation on model in camera view
131         m_fRotY = 0.0f;     // Rotation on model in camera view
132     }
133
134     void UpdateView()
135     {
136         glTranslatef(0.0f, 0.0f, -m_fZoom);
137         glTranslatef(m_fPosX, m_fPosY, 0.0f);
138         glRotatef(m_fRotX, 1.0f, 0.0f, 0.0f);
```

```
139         glRotatef(m_fRotY, 0.0f, 1.0f, 0.0f);
140     }
141     void UpdateOnMouseMove(WPARAM wParam, LPARAM lParam)
142     {
143         int xPos = GET_X_LPARAM(lParam);
144         int yPos = GET_Y_LPARAM(lParam);
145 
146         int diffX = (int)(xPos - m_fLastX);
147         int diffY = (int)(yPos - m_fLastY);
148         m_fLastX  = (float)xPos;
149         m_fLastY  = (float)yPos;
150 
151         int nFlags = wParam;
152 
153         // Left mouse button
154         if (nFlags & MK_LBUTTON)
155         {
156             m_fRotX += (float)0.5f * diffY;
157 
158             if ((m_fRotX > 360.0f) || (m_fRotX < -360.0f))
159             {
160                 m_fRotX = 0.0f;
161             }
162 
163             m_fRotY += (float)0.5f * diffX;
164 
165             if ((m_fRotY > 360.0f) || (m_fRotY < -360.0f))
166             {
167                 m_fRotY = 0.0f;
168             }
169         }
170 
171         // Right mouse button
172         else if (nFlags & MK_RBUTTON)
173         {
174             m_fZoom -= (float)0.1f * diffY;
175         }
176 
177         // Middle mouse button
178         else if (nFlags & MK_MBUTTON)
179         {
180             m_fPosX += (float)0.05f * diffX;
181             m_fPosY -= (float)0.05f * diffY;
182         }
183     }
184 };
185 
186 MayaCamera g_mc;
187 
188 // Set up pixel format for graphics initialization
189 void SetupPixelFormat()
190 {
191     PIXELFORMATDESCRIPTOR pfd, *ppfd;
192     int pixelformat;
193 
194     ppfd = &pfd;
195 
196     ppfd->nSize = sizeof(PIXELFORMATDESCRIPTOR);
197     ppfd->nVersion = 1;
198     ppfd->dwFlags = PFD_DRAW_TO_WINDOW | PFD_SUPPORT_OPENGL | PFD_DOUBLEBUFFER;
199     ppfd->dwLayerMask = PFD_MAIN_PLANE;
200     ppfd->iPixelType = PFD_TYPE_COLORINDEX;
201     ppfd->cColorBits = 16;
202     ppfd->cDepthBits = 16;
203     ppfd->cAccumBits = 0;
204     ppfd->cStencilBits = 0;
205 
206     pixelformat = ChoosePixelFormat(hDC, ppfd);
207     SetPixelFormat(hDC, pixelformat, ppfd);
```

```
208 }
209
210 // Initialize OpenGL graphics
211 void InitGraphics()
212 {
213     hDC = GetDC(hWnd);
214
215     SetupPixelFormat();
216
217     hRC = wglCreateContext(hDC);
218     wglMakeCurrent(hDC, hRC);
219
220     Vec3D pmin, pmax;
221     for (int j = 0; j < K; j++)
222     {
223         stl[j].Load(stlnames[j].c_str(), offset[j]);
224         stl[j].ComputeBoundingBox(pmin, pmax);
225     }
226     //cout << "Mo tep tin:" << endl;
227
228     ifstream files[K];
229     for (int j = 0; j < K; j++)
230     {
231         files[j].open(matnames[j].c_str());
232
233         if (!files[j].is_open())
234             cout << "Loi mo tep so " << j + 1 << endl;
235     }
236     matrix4x4f m;
237
238     while (files[0].good() && files[1].good() && files[2].good())
239     {
240         for (int j = 0; j < K; j++)
241         {
242             files[j] >> m;
243             mat[j].push_back(m);
244         }
245     }
246     N = mat[0].size();
247
248     glClearColor( 0.0f, 0.0f, 0.0f, 1.0f );
249     glEnable(GL_DEPTH_TEST);
250
251     glMatrixMode( GL_PROJECTION );
252     glLoadIdentity();
253     gluPerspective( 45.0f, 640.0f / 480.0f, 0.1f, 100.0f);
254 }
255
256 // Resize graphics to fit window
257 void ResizeGraphics()
258 {
259     // Get new window size
260     RECT rect;
261     GetClientRect(hWnd, &rect);
262
263     int nWidth  = rect.right;
264     int nHeight = rect.bottom;
265     glViewport(0, 0, nWidth, nHeight);
266
267     glMatrixMode( GL_PROJECTION );
268     glLoadIdentity();
269     gluPerspective( 45.0, (GLdouble)nWidth / (GLdouble)nHeight, 0.1, 100.0);
270
271 }
272
273 // Draw frame
274 void DrawGraphics()
275 {
276     glClear( GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT );
```

```
277
278     glMatrixMode( GL_MODELVIEW );
279     glLoadIdentity();
280
281     // Thay đổi tham số để thay đổi hướng nhìn
282     gluLookAt( -2.0, 1.0, 5.0,  // Camera position
283                0.0, 0.0, 0.0,   // Look-at point
284                0.0, 1.0, 0.0 ); // Up vector
285
286     g_mc.UpdateView();  // Để điều khiển camera bằng chuột:
287                         // + nút trái để xoay
288                         // + nút phải để thu phóng
289                         // + nút giữa để tịnh tiến
290
291     DrawAxes(2, 1); // Vẽ các trục tọa độ: x màu đỏ, y xanh lá cây, z xanh da trời
292
293 //  DrawGrid();
294
295     glColor3d(0, 1, 1); // Màu cyan
296
297     if (count >= N || count < 0)
298         count = 0;
299
300     for (int j = 0; j < K; j++)
301     {
302         glPushMatrix();
303         {
304             glScaled(0.01, 0.01, 0.01); // Thu nhỏ hình vì quá to
305
306             glMultMatrixf( mat[j][count].m);
307
308             if (g_moving)
309                 count++;
310
311             glColor4dv(color[j]);
312             stl[j].Draw(false, g_shade);
313         }
314         glPopMatrix();
315     }
316
317     Sleep(100);
318     // Show the new scene
319     SwapBuffers(hDC);
320 }
321
322 // Handle window events and messages
323 LONG WINAPI MainWndProc (HWND hWnd, UINT uMsg, WPARAM  wParam, LPARAM  lParam)
324 {
325     switch (uMsg)
326     {
327     case WM_SIZE:
328         ResizeGraphics();
329         break;
330
331     case WM_CLOSE:
332         DestroyWindow(hWnd);
333         break;
334
335     case WM_MOUSEMOVE: // Để điều khiển camera bằng chuột
336         g_mc.UpdateOnMouseMove(wParam, lParam);
337         break;
338
339     case WM_DESTROY:
340         PostQuitMessage(0);
341         break;
342
343        case WM_KEYDOWN:
344         {
345             switch( wParam )
```

```
346             {
347                 case 's':
348                 case 'S':
349                     g_shade = !g_shade;
350                     break;
351 
352                 case 'm':
353                 case 'M':
354                     g_moving = !g_moving;
355                     break;
356 
357                 case VK_LEFT:
358                     count--;
359                     break;
360 
361                 case VK_RIGHT:
362                     count++;
363                     break;
364             }
365         }
366     // Default event handler
367     default:
368         return DefWindowProc (hWnd, uMsg, wParam, lParam);
369         break;
370     }
371 
372     return 1;
373 }
374 
375 int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int
    nCmdShow)
376 {
377 
378     const LPCWSTR appname = TEXT("OpenGL Sample");
379 
380     WNDCLASS wndclass;
381     MSG      msg;
382 
383     // Define the window class
384     wndclass.style         = 0;
385     wndclass.lpfnWndProc   = (WNDPROC)MainWndProc;
386     wndclass.cbClsExtra    = 0;
387     wndclass.cbWndExtra    = 0;
388     wndclass.hInstance     = hInstance;
389     wndclass.hIcon         = LoadIcon(hInstance, appname);
390     wndclass.hCursor       = LoadCursor(NULL,IDC_ARROW);
391     wndclass.hbrBackground = (HBRUSH)(COLOR_WINDOW+1);
392     wndclass.lpszMenuName  = appname;
393     wndclass.lpszClassName = appname;
394 
395     // Register the window class
396     if (!RegisterClass(&wndclass)) return FALSE;
397 
398     // Create the window
399     hWnd = CreateWindow(
400             appname,
401             appname,
402             WS_OVERLAPPEDWINDOW | WS_CLIPSIBLINGS | WS_CLIPCHILDREN,
403             CW_USEDEFAULT,
404             CW_USEDEFAULT,
405             800,
406             600,
407             NULL,
408             NULL,
409             hInstance,
410             NULL);
411 
412     if (!hWnd) return FALSE;
413 
```

```
414     // Initialize OpenGL
415     InitGraphics();
416 
417     // Display the window
418     ShowWindow(hWnd, nCmdShow);
419     UpdateWindow(hWnd);
420 
421     // Event loop
422     while (1)
423     {
424         if (PeekMessage(&msg, NULL, 0, 0, PM_NOREMOVE) == TRUE)
425         {
426             if (!GetMessage(&msg, NULL, 0, 0)) return TRUE;
427 
428             TranslateMessage(&msg);
429             DispatchMessage(&msg);
430         }
431         DrawGraphics();
432     }
433 
434     wglDeleteContext(hRC);
435     ReleaseDC(hWnd, hDC);
436 }
```