ФАКУЛТЕТ ИНЖЕЊЕРСКИХ НАУКА

Универзитета у Крагујевцу

Назив студијског програма: **Машинско инжењерство**

Ниво студија: **Основне академске студије**

Модул: **Информатика у инжењерству**

Предмет: **Софтверски инжењеринг**

Број индекса: **47/2010**

Никола З. Бараћ

ИЗРАДА АПЛИКАЦИЈЕ ЗА ИСЦРТАВАЊЕ 3D МОДЕЛА НА ОСНОВУ STL ФАЈЛА

-ЗАВРШНИ РАД-

Комисија за преглед и одбрану:

1. др Ненад Филиповић - ментор
2. _____ Датум одбране:_____
3. _____ Оцена:_____

Крагујевац, септембар 2015

У оквиру овог завршног рада кандидат треба да опише основне појмове из области *3D* графике, *CAD* модела, њиховог исцртавања у програмском језику *C++.* Практичан део рада односи се на креирање апликације која ће учитати одабрани *STL* фајл и на основу истог исцртати 3Д модел. Такође, потребно је укључити и основне функционалности *CAD* софтвера, као сто су ротација око оса, зумирање, промена погледа, управљање помоћу миша и слично.

Препоручена литература:

[1] Ненад Филиповић (2006), Објектно орјентисано програмирање, ФТН Чачак.

[2] *The Red Book*. OpenGL Programming Guide, 8th Edition. A tutorial and reference book.

Крагујевац, 26. 10. 2014.

Ментор:

др Ненад Филиповић, ред. проф.

_____

Резиме:

Рад је настао као резултат интересовања за област *3D* рачунарске графике и објектно орјентисаног програмирања. Приказани су основни појмови у области *3D* графике. Креирана је апликација која врши учитавање *STL* фајла, његову конверзију у нумеричке вредности и исцртавање *3D* модела на основу добијених вредности.

САДРЖАЈ

1. УВОД

Завршни рад “Израда апликације за исцртавање *3D* модела на основу *STL* фајла” настао је као резултат интересовања за област *3D* рачунарске графике и објектно орјентисаног програмирања. Упознајући се са овим областима дошао сам до многобројних сазнања о особинама програмског језика *C++,* као и о могућностима практичне примене апликација које се у њему израђују.

Овај завршни рад се састоји из пет поглавља:

- У првом поглављу изложене су уводне напомене, као и напомене о програмском језику у којем је израђена апликација и библиотекама које су коришћене.

- У другом поглављу објашњене су основе компјутерске графике.

- Треће поглавље приказује израду апликације корак по корак, уз објашњења о начину функционисања програма и алгоритма.

- Четврто поглавље је закључак овог завршног рада, где су сумирана достигнућа овог рада.

2. ОСНОВНИ ПОЈМОВИ

2.1 *STL* фајл формат

STL (***ST****ereo****L****ithography*) је формат пореклом из *stereolithography CAD* софтвера koji је створио *3D Systems Company*. Овај формат датотеке је подржан од стране многих *3D* софтверских пакета, широко се користи за брзу израду прототипова , *3D* штампање и *computer-aided manufacturing*. *STL* фајлови описују само геометрију површина тродимензионалног објекта без икаквог представљања боја, текстура или другог заједничког *CAD model* атрибута. Формат може имати *ASCII* и *binary* репрезентацију. *Binary* фајлови су чешћи, јер су компактнији. Ова апликација вршиће учитавање *ASCII* репрезентације *STL* фајла.

ASCII формат почиње линијом:

```
solid name
```

где је *name* опционални *string* (текст). Фајл може садржати неодређен број торуглова, а сваки троугао је представљен на следећи начин:

```
facet normal ni nj nk
    outer loop
        vertex v1x v1y v1z
        vertex v2x v2y v2z
        vertex v3x v3y v3z
    endloop
endfacet
```

n и *v* представљају реалне бројеве. Фајл се завршава линијом:

```
endsolid name
```

2.2 C++ програмски језик

C++ је виши програмски језик који је првобитно развијен у *Bell Labs* (лабораторији телекомуникационе компаније *Bell*) за објектно оријентисано програмирање у пројекту под руководством Бјарнеа Строуструпа током 1980их као проширење програмском језику *C*, па му је оригинално име било „C са класама“ (*eng*. *C with classes*). Због велике потражње за објектно оријентисаним језицима и способностима, стандард за програмски језик *C++* ратификован је 1998. у стандарду ISO/IEC 14882.

Програмски пример односно пример најпростијег “Hello World” програма:

```
#include <iostream>
using namespace std;
int main(){
    cout << "Hello World!" << endl;
    return 0;
}
```

Програмски језик C++ је наследио целокупну синтаксу од програмског језика C, а додата су следећа проширења:

- класе
  - дефинисање функција чланица и ограничавања њиховог нивоа приступа (јавне, заштићене и приватне)
  - наслеђивање класа
  - виртуелне и апстрактне методе класа
  - конструктори и деструктори
  - дефинисање функција за операторе над класних типовима (тзв. „операторске функције“)
- референце (тип података уведен ради поједностављивања рада са показивачима)
- именски простори (*eng*. *namespace*)
- шаблони (*eng*. *Templates*)
- нови оператори за манипулацију динамичком меморијом, **new** и **delete**
- низ нових библиотека објеката и функција

Главне библиотеке које које су коришћене при изради апликације су *Win32* і *OpenGL*. *Win32* је саставни део *Windows API* (*application programming interfaces*) и представља стандардни саставни део C++ језика. *OpenGL* је нестандардна библиотека која је широко прихваћена за писање програма који раде са дводимензионалном и тродимензионалном компјутерском графиком.

2.3 OpenGL

OpenGL (*eng. Open Graphics Library*) је стандардна спецификација која описује вишеплатформски програмски интерфејс за писање програма који раде са дводимензионалном и тродимензионалном рачунарском графиком. Интерфејс чини преко 250 различитих функција које се могу користити за израду комплексних тродимензионалних сцена од једноставних елемената. OpenGL је развијен од стране Силикон Графикс (*eng. Silicon Graphics Inc*.) 1992. године и популаран је у индустрији видео игара где је пандан Мајкрософтовом *Direct3D*. Поред овога, много чешће се користи у научне сврхе, код *CAD*-програма, у пројектима виртуелне стварности као и у разним симулаторима.

Следи једноставан пример једне *OpenGL* сцене са коментарима. На слици је приказан излаз нацртан директно у *OpenGL*.

```
glClear(GL_COLOR_BUFFER_BIT);
```

Чишћење фрејма пре почетка цртања. Цео фрејм ће бити обојен у подразумевану боју која је у овом случају црна.

```
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
```

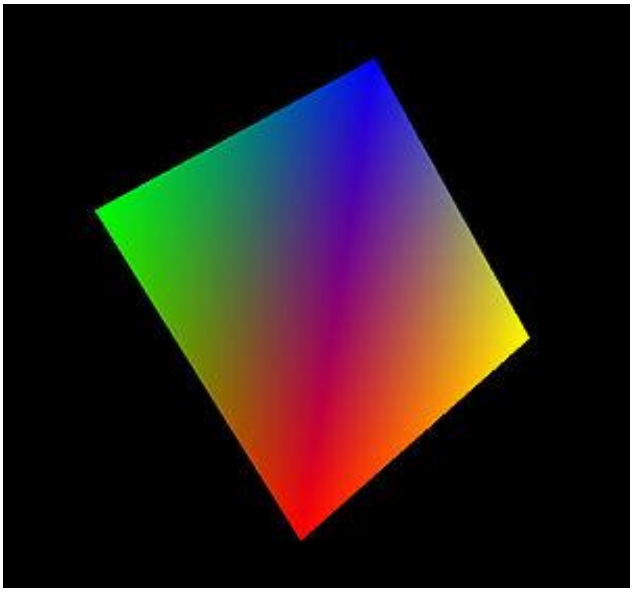
Наглашавање да ће матрица за приказ тродимензионог модела бити трансформисана и њено подешавање на идентичну матрицу.

```
glTranslatef(0,0,-5);
glRotatef(45,0,1,1);
```

Транслација за *-5* јединица по *Z* оси (помоћу *glTranslatef*) и ротација за *45°* око вектора *(0,1,1)* (помоћу *glRotatef*). Ове трансформације се врше на претходно изабраној матрици.

```
glBegin(GL_POLYGON);
       glColor3f(1,0,0); glVertex3f(-1,-1,0); // црвено теме
       glColor3f(0,1,0); glVertex3f(-1, 1,0); // зелено теме
       glColor3f(0,0,1); glVertex3f( 1, 1,0); // плаво теме
       glColor3f(1,1,0); glVertex3f( 1,-1,0); // жуто теме
glEnd();
```

Следи цртање модела. Биће исцртан квадрат у *XY* равни са тачкама у *(±1,±1)* (задатим са *glVertex3f*), чија темена редом имају боје (задате са *glColor3f*): црвена, зелена, плава, жута. Због претходних трансформација над матрицом којом се координате квадрата трансформишу, исти ће бити приказан у пројекцији.



Слика 1. Графички приказ описане сцене

3. ОСНОВЕ КОМПЈУТЕРСКЕ ГРАФИКЕ

Компјутерска графика (eng. Computer Graphics) јесте поље визуелног рачунарства где се помоћу компјутера ствара слика. Компјутерска графика се може подијелити у неколико поља: *real time 3D rendering* (користи се у рачунарским играма), компјутерска анимација, стварање специјалних ефеката, обрада слике и моделирање (користи се у медицинске сврхе, као и у архитектури и CAD софтверима).

3.1 *2D* компјутерска графика

Најзначајнији помак у рачунарској графици је било представљање катодне цеви. Имамо два приступа у *2D* графици: векторска и растерска графика. Векторска графика садржи тачне геометријске податке, топологију, координатне позиције тачака, везе између тачака (за формирање линија и путања), боју и тако даље. Векторска графика користи једноставне облике као што су кругови и правоугаоник и остали. Због некомпатибилности са већином програма векторска графика конвертује се у растерску (*.jpg, .bmp*...). Растерска графика је стална дводимензијонална мрежа пиксела. Сваки пиксел има своју вредност, као што је осветљење, боја, провидност. Растерска има коначну резолуцију и ако се она повећа најчешће губи квалитет, што није случај са векторском. Коначно, постоје формати који укључују и растерску и векторску графику (*.pdf, .swf...).*

3.2 3D компјутерска графика

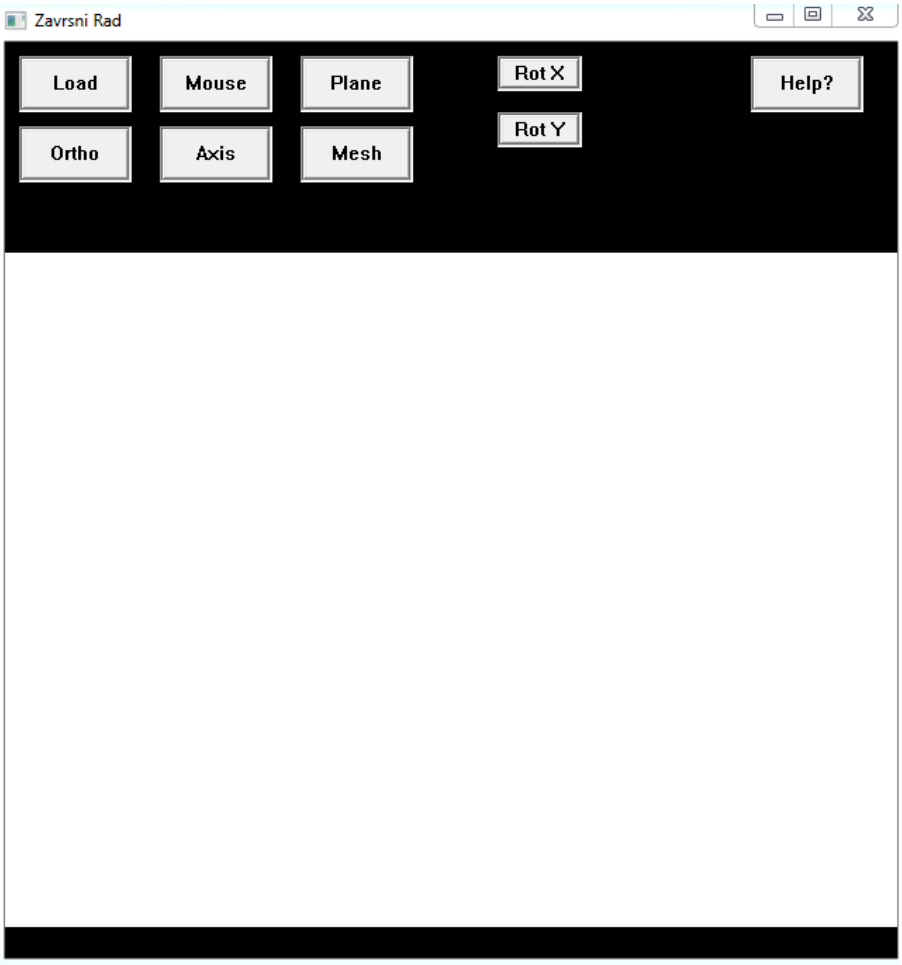
Представљањем веома јаких компјутера дошла је и *3D* компјутерска графика, темељена на векторској графици. Принцип је скоро исти, уместо чувања података о тачкама, линијама и кривим у дводимензионалном простору, чувају се подаци у тродимензионалном простору. Тродимензионални полигони су суштина *3D* компјутерске графике. Данас осим чувања *3D* полигона у меморији, графички софтвер може извршавати и много компликованије ствари као што су сенчење, додавање текстура и растеризација који дају осјећај стварности неком рачунарски направљеном објекту.

Поступак сенчења је рачунање или симулирање како ће полигон изгледати када на њега пада нестварна (рачунарски створена, виртуелна) светлост. Светлост није једини фактор, битан је и начин сенчења неког полигона. Сенчење омогућава пуно реалнији приказ *3D* слике, посебно се користи у компјутерским играма и *CGI* (*eng. Computer-generated imagery*) филмовима.

Текстура је слика која се “лепи” на *3D* полигоне, Осим боја, могуће је стављати слике за реалнији приказ модела.

4. ИЗРАДА АПЛИКАЦИЈЕ

Пре почетка израде саме апликације потребно је замислити њен изглед и начин на који ће функционисати. Како би дошло до решења задатка, односно до креирања апликације која ће учитати *STL* фајл, конвертовати текст у бројне вредности, односно координате елементарних троуглова које се чувају у објекту типа *model* , затим исцртати 3D модел коришћењем конвертованих координата, биће нам потребна два прозора (*eng. Windows Form)* . Први прозор који ће бити родитељ, односно главни прозор (*eng. Parent Windows*), садржаће контроле типа *button* помоћу којих ће се управљати програмом и моделом. Други прозор, дете прозор (*eng. Child Windows*) , налазиће се у оквиру главног прозора и користиће се за приказивање односно исцртавање 3D модела. Коначни изглед апликације може се видети на слици 2.



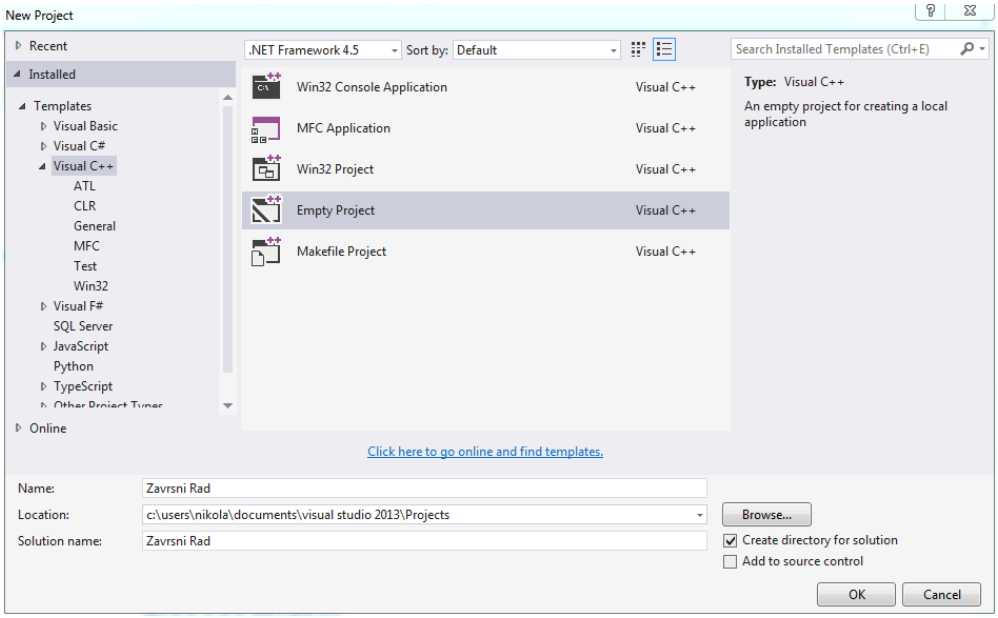
Слика 2. Коначни изглед апликације

4.1 Креирање апликације у програму *Visual Studio*

Visal Studio је комплетан скуп развојних алата за изградњу *ASP.NET* (*eng. Active Server Pages .NET*) Веб апикација, ХМЛ Веб сервиса, десктоп апликација И мобилних апликација. Програмски језици укључени у *Visal Studio* су: *Visual Basic*, *Visual C#, Visual C++*. Они користе исто развојно окрузење (*IDE*), алат који омогућава дељење и олакшава стварање мешовитих језичких решења. *Microsoft* обезбеђује бесплатно преузимање "*Express*" издања свог *Visal Studio 2010* са својим компонентама.

Да бисмо направили нову апликацију потребно је урадити следеће:

1. У менију *File* потребно је изабрати *New Project*, отвара се оквир за дијалог *New Project* (слика 2). Овај оквир излистава шаблоне које можемо користити као полазну основу за прављење апликације.
2. У *Project Types* окну потребно је изабрати *Visual C++* опцију. У окну *Templates* изабрацемо *Empty Project*.
3. У поље *Name* потребно је уписати назив пројекта, а затим потврдити притиском на *OK*.
4. Да би променили препоручену локацију пројекта потребно је одабрати *Tools* на менију и селектовати *Options*. Затим је потребно изабрати *Project and Solutions* и селектовати опцију *General*. У *Project Location text box* можемо унети жељену локацију за фајлове и пројекте.



Слика 3. Креирање новог пројекта

4.2 Креирање прозора

За реализацију пројекта потребна су два прозора. У првом прозору биће смештене контроле, а у другом ће се вршити приказ 3D модела.

Да би се креирао главни прозор односно *Parent Windows* најпре региструјемо *WNDCLASSEX* класу:

```
WNDCLASSEX wcMain;
wcMain.cbSize = sizeof(WNDCLASSEX);
wcMain.style = CS_OWNDC;
wcMain.lpfnWndProc = ParentWindowProc;
wcMain.cbClsExtra = 0;
wcMain.cbWndExtra = 0;
wcMain.hInstance = hInstance;
wcMain.hIcon = LoadIcon(NULL, IDI_APPLICATION);
wcMain.hCursor = LoadCursor(NULL, IDC_ARROW);
wcMain.hbrBackground = (HBRUSH)GetStockObject(BLACK_BRUSH);
wcMain.lpszMenuName = NULL;
wcMain.lpszClassName = "ParentWindowClass";
wcMain.hIconSm = LoadIcon(NULL, IDI_APPLICATION);
if (!RegisterClassEx(&wcMain))
    return 0;
```

Слика 4. Регистрација *Parent* прозора

Ако је регистрација успешна, врши се креирање прозора позивањем функције:

```
pHwnd = CreateWindowEx(0,
                       "ParentWindowClass",
                       WP_NAME,
                       WS_OVERLAPPEDWINDOW ^ WS_THICKFRAME,
                       WP_POS_X,
                       WP_POS_Y,
                       WP_WIDTH,
                       WP_HEIGHT,
                       NULL,
                       NULL,
                       hInstance,
                       NULL);
```

Слика 5. Креирање *Parent* прозора

Сличан поступак је и за креирање дете прозора (*eng. Child Window*). Најпре се региструје *WNDCLASSEX* класа:

```
WNDCLASSEX wcChild;
wcChild.cbSize = sizeof(WNDCLASSEX);
wcChild.style = CS_OWNDC;
wcChild.lpfnWndProc = ChildWindowProc;
wcChild.cbClsExtra = 0;
wcChild.cbWndExtra = 0;
wcChild.hInstance = hInstance;
wcChild.hIcon = LoadIcon(NULL, IDI_APPLICATION);
wcChild.hCursor = LoadCursor(NULL, IDC_ARROW);
wcChild.hbrBackground = (HBRUSH)GetStockObject(WHITE_BRUSH);
wcChild.lpszMenuName = NULL;
wcChild.lpszClassName = "ChildWindowClass";
wcChild.hIconSm = LoadIcon(NULL, IDI_APPLICATION);;
if (!RegisterClassEx(&wcChild))
    return 0;
```

Слика 6. Регистрација *Child* прозора

Ако је регистрација успешна, врши се креирање прозора, при чему је овај пут веома важно водити рачуна о аргументима које треба проследити функцији за креирање прозора:

```
HWND cHwnd = CreateWindowEx(0,
                        "ChildWindowClass",
                        "",
                        WS_CHILD, //Child Type
                        0,
                        150,
                        WC_WIDTH,
                        WC_HEIGHT,
                        pHwnd,    //Parent Window
                        NULL,
                        hInstance,
                        NULL);
```

Слика 7. Креирање *Child* прозора

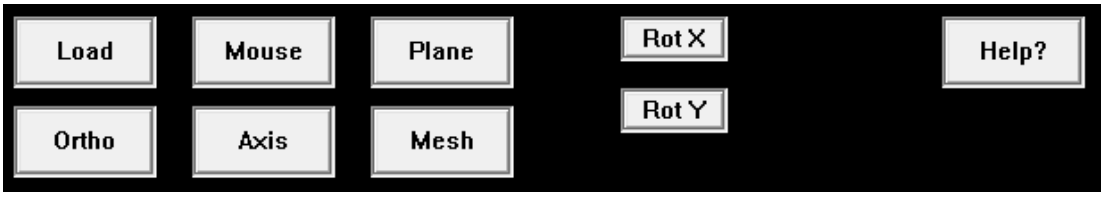
4.3 Додавање контрола

Након креирања прозора, у главни прозор додају се контроле за управљање програмом (слика 8). Помоћу ових контрола вршиће се све од учитавања *STL* фајла до управљања моделом и окружењем.

```
case WM_CREATE:
{
    CreateButtonEx("Load", BS_TEXT, 10, 10, 80, 40, hwnd, ID_BUTTON_01, NULL);
    CreateButtonEx("Ortho", BS_TEXT, 10, 60, 80, 40, hwnd, ID_BUTTON_02, NULL);
    CreateButtonEx("Mouse", BS_TEXT, 110, 10, 80, 40, hwnd, ID_BUTTON_03, NULL);
    CreateButtonEx("Axis", BS_TEXT, 110, 60, 80, 40, hwnd, ID_BUTTON_04, NULL);
    CreateButtonEx("Plane", BS_TEXT, 210, 10, 80, 40, hwnd, ID_BUTTON_05, NULL);
    CreateButtonEx("Mesh", BS_TEXT, 210, 60, 80, 40, hwnd, ID_BUTTON_06, NULL);
    CreateButtonEx("Rot X", BS_TEXT, 350, 10, 60, 25, hwnd, ID_BUTTON_07, NULL);
    CreateButtonEx("Rot Y", BS_TEXT, 350, 50, 60, 25, hwnd, ID_BUTTON_08, NULL);
    CreateButtonEx("Help?", BS_TEXT, 530, 10, 80, 40, hwnd, ID_BUTTON_09, NULL);
}
```

Слика 8. Додавање контрола типа *Button*

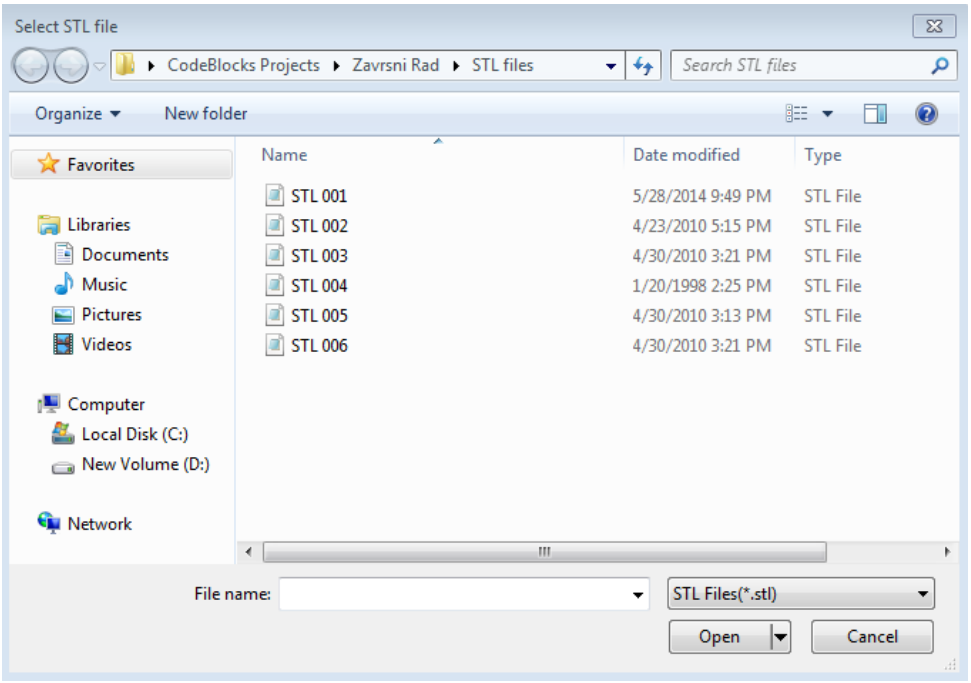
Након компајлирања програма, могу се видети додате контроле (слика 9).



Слика 9. Контроле типа *Button*

4.4 Учитавање *STL* фајла

На самом почетку дизајнирања апликације потребно је омогућити кориснику да одабере *STL* фајл који жели да буде приказан. Да би се то остварило потребно је отворити фајл дијалог (слика 10).



Слика 10. Фајл дијалог (*eng. File Dialog*)

Притиском на дугме *Load* покреће се код који врши отварање дијалога и као аргумент враћа локацију и име изабраног фајла, и на тај начин се може учитати жељени фајл:

```
string  GetFileName(){
    const int BUFSIZE = 1024;
    char buffer[BUFSIZE] = {0};
    OPENFILENAME ofns = {0};
    ofns.lpstrFilter = "STL Files(*.stl)\0*.stl\0";
    ofns.lStructSize = sizeof( ofns );
    ofns.lpstrFile = buffer;
    ofns.nMaxFile = BUFSIZE;
    ofns.lpstrTitle = "Select STL file";
    GetOpenFileName( &ofns );
    return buffer;
}
```

Слика 11. Функција за отварање фајл дијалога

Након одабира фајла покреће се функција која конвертује текст у бројне вредности, чува их у објекту типа *model* и на тај нацин се добијају координате темена елементарних троуглова од којих је сачињен модел који ће бити исцртан.

4.5 Подешавање *OpenGL*-а

Како би се вршило исцртавање модела, најпре се мора подесити *OpenGL* за дете прозор (слика 12). Ово је веома важан корак јер ће свако исцртавање, ротација, зумирање модела бити контролисани функцијама из ове библиотеке.

```
void EnableOpenGL(HWND hwnd, HDC* hDC, HGLRC* hRC) {
    PIXELFORMATDESCRIPTOR pfd;

    *hDC = GetDC(hwnd);

    ZeroMemory(&pfd, sizeof(pfd));
    pfd.nSize = sizeof(pfd);
    pfd.nVersion = 1;
    pfd.dwFlags = PFD_DRAW_TO_WINDOW |
                  PFD_SUPPORT_OPENGL |
                  PFD_DOUBLEBUFFER;
    pfd.iPixelType = PFD_TYPE_RGBA;
    pfd.cColorBits = 24;
    pfd.cDepthBits = 32;
    pfd.iLayerType = PFD_MAIN_PLANE;

    SetPixelFormat(*hDC, ChoosePixelFormat(*hDC, &pfd), &pfd);
    *hRC = wglCreateContext(*hDC);
    wglMakeCurrent(*hDC, *hRC);
}
```

Слика 12. Функција за подешавање *OpenGL*-а

4.6 Исцртавање модела

Исцртавање модела врши се у дете прозору, периодичним позивањем *DrawModel()* функције:

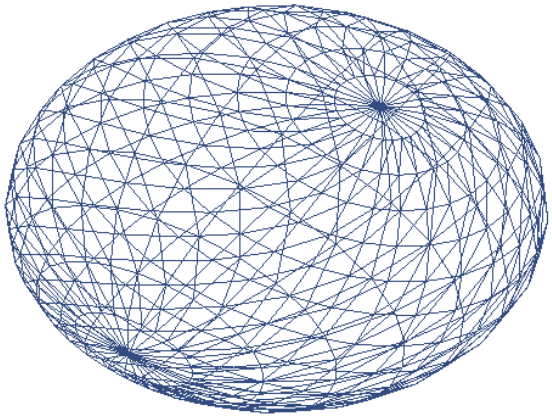
```
void DrawModel(){
    int i = 0;
    if(isMesh){
        glPolygonMode(GL_FRONT, GL_LINE);
        glPolygonMode(GL_BACK, GL_LINE);
    }
    glBegin(GL_TRIANGLES);
    while(i < Triangle->count){
        glNormal3f(Triangle->facet[i][0],Triangle->facet[i][1],Triangle->facet[i][2]);
        glColor3f(0.2,0.3,0.5);

        glVertex3f(Triangle->coordinate[i][0],Triangle->coordinate[i][1],Triangle->coordinate[i][2]);
        glVertex3f(Triangle->coordinate[i][3],Triangle->coordinate[i][4],Triangle->coordinate[i][5]);
        glVertex3f(Triangle->coordinate[i][6],Triangle->coordinate[i][7],Triangle->coordinate[i][8]);

        i++;
    }
    glEnd();
    if(isMesh){
        glPolygonMode(GL_FRONT, GL_FILL);
        glPolygonMode(GL_BACK, GL_FILL);
    }
}
```

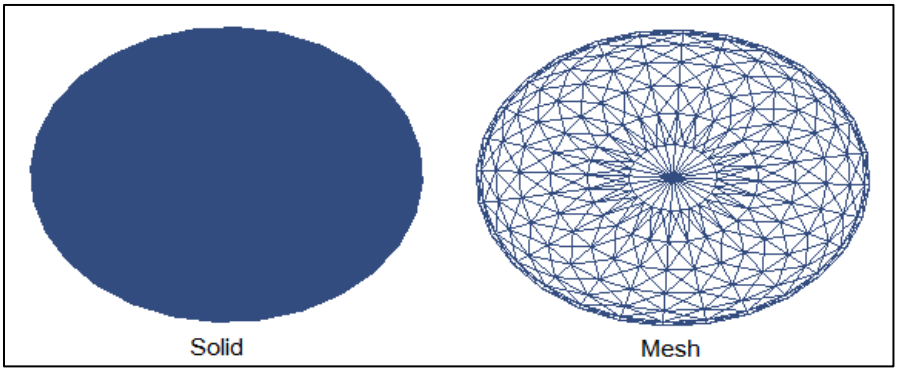
Слика 13. *DrawModel()* функција

Функција исцртава елементарне троуглове користећи координате добијене конверзијом *STL* фајла. Троуглови се исцртавају један по један и на тај начин се формира модел (слика 14).



Слика 14. Пример модела сачињеног од елементарних троуглова

У зависности да ли је изабран *Solid* или *Mesh Mode*, функција врши другачије исцртавање модела:



Слика 15. Пример *Solid* модела (лево) и *Mesh* модела (десно)

4.7 Остале команде

Притиском на дугме *Ortho* укључује се и искључује ортогонална пројекција:

```
case ID_BUTTON_02:
{
    if (!isLoaded) break;
    isMousemode = false;
    if (isOrtho) {
        isOrtho = false;
        SetPlane();
    }
    else {
        isOrtho = true;
    }

    ClearText(hwnd);

    SetFocus(hwnd);
}
break;
```

Слика 16. Код везан за дугме *Ortho*

Уколико је ортогонална пројекција укључена, притиском на дугме *Mouse* улази се у *Mouse Mode*, што омогућава кориснику да моделом управља помоћу миша. Притиском на исто дугме или притиском на тастер *ESC* (*Escape*) на тастарури излази се из *Mouse Mode*:

```
case ID_BUTTON_03:
{
    if (!isOrtho || !isLoaded) break;
    if (isMousemode) {
        isMousemode = false;
        planeId = 0;
        SetPlane();
    }
    else {
        isMousemode = true;
    }

    ClearText(hwnd);

    SetFocus(hwnd);
}
break;
```

Слика 17. Код везан за дугме *Mouse*

Док је *Mouse Mode* активан, корисник може да врши ротацију модела померањем миша (слика 18) и његово окретањем точкића (*eng. Mousewheel*) (слика 19).

```
case WM_TIMER:
{
    if (!isLoaded) break;
    Reshape(WC_WIDTH, WC_HEIGHT);
    Display();

    if (!(isOrtho && isMousemode)) break;
        POINT p;
        GetCursorPos(&p);
        ScreenToClient(hwnd, &p);

        if (prevCursorX != p.x) xRot += (int)((prevCursorX - p.x) * CURSOR_VEL);

        if (prevCursorY != p.y) yRot += (int)((prevCursorY - p.y) * CURSOR_VEL);

        prevCursorX = p.x;
        prevCursorY = p.y;

        ResetCamera();
}
break;
```

Слика 18. Код за праћење померања миша

```
case WM_MOUSEWHEEL:
{
    if (!(isOrtho && isMousemode)) break;
        short s = HIWORD(wParam);
        if (s > 0) {
            if(dValue < maxZoom) dValue += zoomValue;
        }
        else if (s < 0){
            if(dValue > minZoom) dValue -= zoomValue;
        }
}
break;
```

Слика 19. Код везан за *Mousewheel*

Дугме *Axis* омогућава кориснику да искључи односно укључи приказ оса координатног система:

```
case ID_BUTTON_04:
{
    if (!isLoaded) break;
    if (showAxis) {
        showAxis = false;
    }
    else {
        showAxis = true;
    }

    SetFocus(hwnd);
}
break;
```

Слика 20. Код везан за дугме *Axis*

Дугме *Plane* омогућава кориснику промену погледа:

```
case ID_BUTTON_05:
{
    if (!isLoaded) break;
    planeId++;
    SetPlane();

    if (planeId == 9) planeId = -1;

    SetFocus(hwnd);
}
break;
```

Слика 21. Код везан за дугме *Plane*

Дугме *Mesh* омогућава кориснику да измени начин исцртавања између *Solid* и *Mesh Mode*. Разлика између *Solid* и *Mesh Mode* може се видети на слици 15.

```
case ID_BUTTON_06:
{
    if (!isLoaded) break;
    if (isMesh) {
        isMesh = false;
    }
    else {
        isMesh = true;
    }

    ClearText(hwnd);

    SetFocus(hwnd);
}
break;
```

Слика 22. Код везан за дугме *Mesh*

Притиском на дугме *Rot X* (слика 23) или тастер ***X*** на тастатури (слика 24) врши се ротација модела по X оси:

```
case ID_BUTTON_07:
{
    if (!isLoaded) break;

    xRot += rotValue;

    SetFocus(hwnd);
}
break;
```

Слика 23. Код везан за дугме *Rot X*

```
case WM_KEYDOWN:
{
    switch (wParam)
    {
        case VK_KEY_X:
        {
            if (!isLoaded) break;

            xRot++;
        }
        break;
```

Слика 24. Код везан за тастер ***X***

Притиском на дугме *Rot Y* (слика 25) или тастер ***Y*** на тастатури (слика 26) врши се ротација модела по Y оси:

```
case ID_BUTTON_08:
{
    if (!isLoaded) break;

    yRot += rotValue;

    SetFocus(hwnd);
}
break;
```

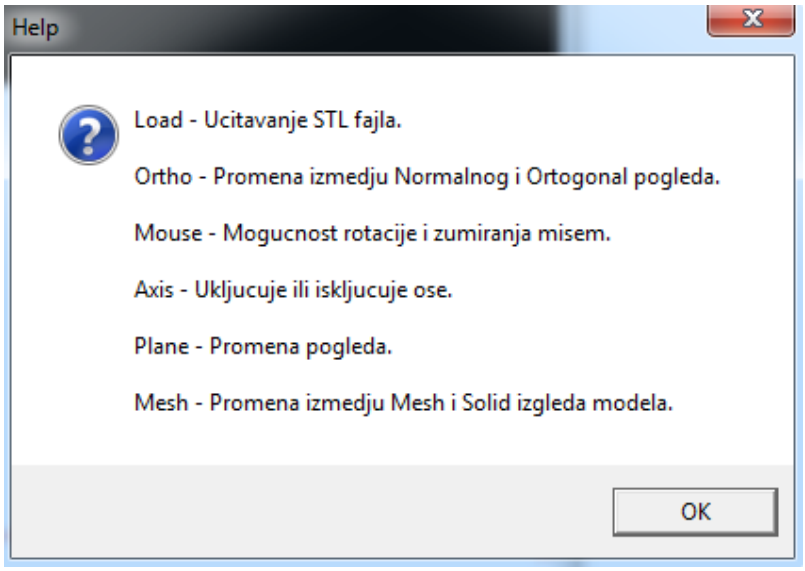
Слика 25. Код везан за дугме *Rot* ***Y***

```
case WM_KEYDOWN:
{
    switch (wParam)
    {
        case VK_KEY_Y:
        {
            if (!isLoaded) break;

            yRot++;
        }
        break;
```

Слика 26. Код везан за тастер ***Y***

Притиском на дугме *Help?* појављује се *MessageBox* који описује функционалност сваке команде, што може помоћи кориснику при коришћењу апликације (слика 27).



Слика 27. *Help* прозор

5. ЗАКЉУЧАК

Иако данашњи *CAD* (*eng. computer-aided design*) софтверски пакети подржавају многе формате, STL формат представља лак и једноставан начин чувања података о моделу систем и веома је распрострањен. Многи *CAM* (*eng. computer-aided manufacturing*) системи користе моделе који су сачињени од елементарних троуглова. *STL* формат није најефикаснији и најбржи метод за чување података, али се најчешће користи за трансфер троугаоне геометрије у *CAM*. Овим радом представљена је апликација која која учитава *STL* формат и на основу истог исцртава *3D* модел. У апликацију су укључене и основне функционалности *CAD* софтвера, као што су ротација, управљање мишем, зумирање, промена погледа.

Јасно је да ни један алгоритам за манипулацију моделом и његово исцртавање не може бити савршен у апсолутном смислу, јер генерално гледајући, постоји много начина који могу довести до сличних или истих резултата. То је један од разлога због којих постоји велики број *3D* графичких библиотека као што су ***OpenGL***, ***Direct3D***, ***Glide API***, ***OGRE***, ***Horde3D***, ***Crystal Space*** и свака има своје предности и мане.

У овом раду описан је алгоритам који је једноставан и поуздан и тежи да произведе што боље резултате. Недостатак је нешто дуже време обраде података и ограничена функционалност, што је и разумљиво јер су коришћени базни алгоритми најједноставнији за разумевање.

6. ЛИТЕРАТУРА

[1] Ненад Филиповић (2006), Објектно орјентисано програмирање, ФТН Чачак.

[2] *The Red Book*. OpenGL Programming Guide, 8th Edition. A tutorial and reference book.

[3] http://stackoverflow.com

[4] http://www.codeproject.com/

[5] https://msdn.microsoft.com/en-us/default.aspx

[6] https://www.youtube.com/user/thecplusplusguy