Увод у *OpenGL (Open Graphics Library)*

Потребне технологије

За решавање овог проблема неопходно је поседовање следећих софтверских компоненти:

- C++ компајлер
- програмско окружење ***Visual Studio 2015***
- Програмске пакете ***FreeGLUT, GLFW, GLEW***

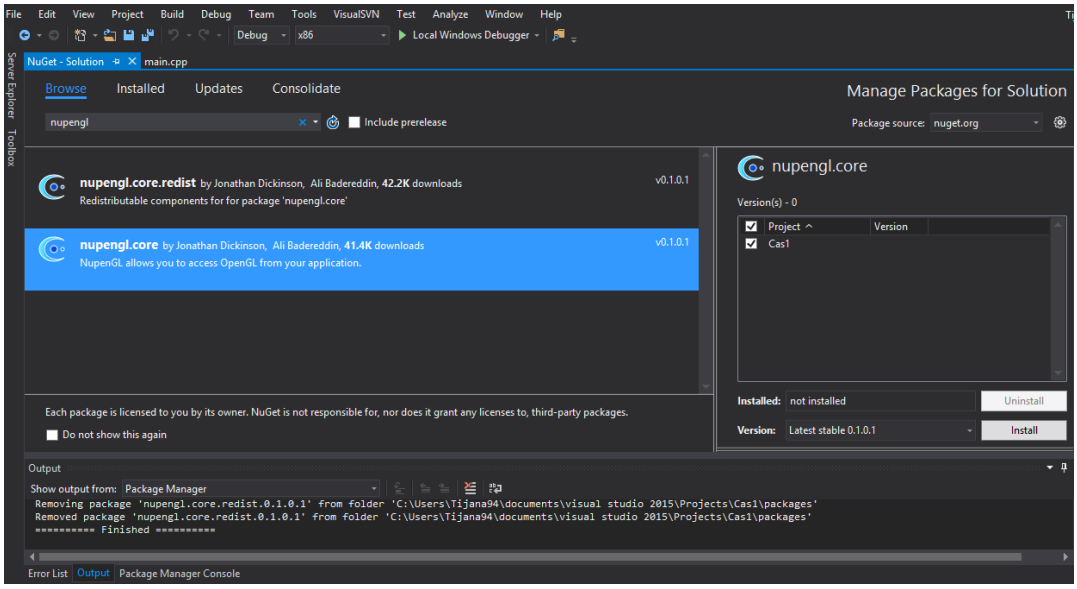
Напомена: Када се креира нови пројекат неопходно је укључити све наведене пакете у пројекат. То је могуће урадити на два начина.

Први начин: Отворити конзолу *Nuget Packet Manager*-а и унети команду следећу команду:

```
Install-Package nupengl.core
```

Ова команда ће укључити све наведене пакете у пројекат.

Други начин: У главном менију изабрати опцију *Tools > NuGet Package Manager > Manage NuGet Packages for Solutions…* Отворити картицу Browse и у делу за претрагу укуцати *nupengl.* Од понуђених резултата одабрати *nupengl.core.* Након одабира, са десне стране ће се појавити креирани пројекат. Потребно је означити цео пројекат као на слици и кликнути на *Install*. Након завршетка инсталације, сви пакети ће бити укључени у пројекат.



OpenGL

OpenGL је софтверски интерфејс ка графичком хардверу. Имплементира се за велики број различитих хардверских платформи. Користи се за писање програма који раде са дводимензионалном и тродимензионалном графиком. Интерфејс садржи преко 700

различитих функција које се могу користити за израду комплексних интерактивних апликација од једноставних примитива. ***OpenGL*** не садржи команде вишег нивоа за описивање тродимензионалних објеката. Тим командама би се могло једноставније креирати комплексан објекат као што су кола, авион, делови тела, међутим у ***OpenGL***-у то мора да се ради једноставним примитивама као што су тачка, линија и полигон. ***OpenGL*** је пандан мајкрософтовом ***DirectX***-у. Има употребу у индустрији видео игара, али се ипак много чешће користи у научне сврхе, у програмима виртуелне стварности као и у разним симулаторима.

Софистицирана библиотека која пружа напредне могућности, а изграђена у основи на библиотеци ***OpenGL*** јесте графичка библиотека ***OpenGL Utility Library (GLU).*** Она је стандардна за све имплементације ***OpenGL***-а. Обезбеђује велики број функција које користе основне методе ***OpenGL***-а за моделирање комплекснијих објеката. Између осталог, омогућава мапирање између координата екрана и светских координата, креирање комплексних површи, као и рутине за трансформацију објеката, позиције камере и то много једноставније за коришћење од оних које се налазе у склопу ***OpenGL***-а.

OpenGL не садржи команде за манипулацију прозорима оперативног система и препознавање догађаја са тастатуре и миша. ***OpenGL Utility Toolkit (GLUT)*** је библиотека која је задужена за комуникацију са оперативним системом. Такође омогућава креирање сложених објеката као што су сфера, торус, чајник и други.

Да бисте имали представу о томе шта се све може са библиотеком ***OpenGL*** може се видети на слици која представља прву страну књиге *OpenGL Programming Guide* („*Red Book*“), а креирана је користећи програмску библиотеку ***OpenGL***.

OpenGL омогућава креирање комплексних објеката помоћу геометријских примитива и њихово распоређивање на сцени. Такође, омогућава и постављање положаја посматрача у односу на сцену, постављање осветљења, као и постављање боја и текстура на моделе. На основу задатог осветљења и боја (текстура) ***OpenGL*** има могућност да на основу већ испрограмираних алгоритама одреди боје објеката на сцени. То се може урадити и екстерним шејдерима (*shaders*) где сами контролишемо то израчунавање. На основу претходно наведеног, аутоматски се врши креирање слика. Ту се долази до појма растеризације која представља конвертовање математичких објеката и њима придружених информација о бојама у пикселе на екрану. ***OpenGL*** омогућава и одсецање делова објеката који се не виде јер су скривени од неког другог објекта. Када се добију пиксели слике, могуће је вршити и операције над самим пикселима. **Пиксел** је најмањи графички елемент слике (***bitmap*** слике). Информације о пикселима се држе у меморији, тачније у битовским равнима (*bitplanes*). Битовске равни су делови меморије где сваки бит држи информације о једном пикселу на екрану.

Када се прича о графици, мора се споменути и процес рендеровања. **Рендеровање** је процес помоћу кога рачунар креира дводимензионалну слику на основу 3D модела.

Синтакса OpenGL-а:

- Све команде имају префикс ***gl*** `(glBegin()).`
- Све константе имају префикс ***GL_*** (GL_POLYGON)***.***
- У ***OpenGL***-у се прихватају стандардни типови података (byte,int,double…), док постоје и нови типови података из библиотеке ***OpenGL*** као што су ***GLbyte, GLint, GLdouble***….
- Поједине команде могу имати до три суфикса (`glVertex3fv`):
    - Први суфикс означава број аргумената (2,3,4)
    - Други означава тип аргумената (*s – short integer, i – integer, f – float, d - double*)
    - Трећи означава да ће се аргумент преносити као вектор (*v*), односно шаље се параметар типа *tip_parametra* (int*, double*,…).* Аргумент који се прослеђује функцији је низ чији број елемената зависи од суфикса који означава број аргумената.

Исход

На овим часовима, студент ће се први пут срести са програмским језиком **C++** и **OpenGL**-ом. Научиће основе **OpenGL**-а, и моћи ће да направи једноставне 2D моделе.

Опис проблема

Упознавање са основним функцијама **OpenGL**-а. Цртање произвољног квадрата, произвољне боје коришћењем софтверског интерфејса **OpenGL**.

Решење

Поступак решавања

Као што је наведено, библиотека ***Glut*** се између осталог користи за комуникацију ***OpenGL***-а са оперативним системом, и да би је користили потребно је прво извршити њену иницијализацију. То се врши командом

`glutInit(&argc, argv).`

Отварање прозора за цртање са задатим називом врши се командом

`glutCreateWindow("New window"),`

а као аргумент се прослеђује стринг са називом прозора. Могуће је контролисати димензију прозора као и позицију на екрану где ће се креирати. Величина прозора се одређује наредбом

`glutInitWindowSize(int width, int height).`

Вредности `width` и `height` задају у пикселима, а одређивање позиције прозора командом

`glutInitWindowPosition(int x, int y)`

где су `x` и `y` координате екрана у пикселима.У тачки на екрану са тим координатама наћи ће се горњи леви угао прозора. Координате горњег левог угла екрана су `(0,0)`, и прва координата означава удаљеност од горњег левог угла екрана по хоризонталној, а друга по вертикалној оси и то у смеру ка десно и доле. Подаци о пикселима који су приказани на екрану чувају се у одређеним деловима, најчешће RAM меморије. Ти делови меморије се називају бафери. При креирању прозора могуће је још дефинисати мод приказа командом

`glutInitDisplayMode(unsigned int mode).`

Променљива `mode` представља битовску мапу или комбинацију битовских мапа битовском операцијом OR (|) чиме се прецизније дефинишу бафери за чување података о пикселима слике. Битовске мапе ће бити објашњене детаљније на наредним часовима, а информације о свим битовским мапама се могу наћи на адреси

https://www.opengl.org/resources/libraries/glut/spec3/node12.html.

Пример креирања прозора:

```
    glutInit(&argc, argv);
    glutInitDisplayMode (GLUT_SINGLE | GLUT_RGBA);
    glutInitWindowSize (250, 250);
    glutInitWindowPosition (100, 80);
    glutCreateWindow ("Hello");
```

У наведеном примеру креира се прозор под називом "*Hello*", величине 250×250 пиксела, а позиција горњег левог угла прозора на екрану биће у тачки $(100,80)$. Прозор користи једноструки бафер (`GLUT_SINGLE`), односно цртање се врши директно у бафер који се приказује на екрану. Могуће је користити и двоструки бафер за цртање. Тада се врши исцртавање на једном баферу док је други приказан на екрану. Када се заврши рендеровање бафера који није приказан на екрану његово приказивање се врши командом `glutSwapBuffers()`.Битовска мапа `GLUT_RGBA` одређује да боје које се користе при исцртавању на екран могу имати $alpha$ параметар. Тачније, боја може бити задата са три или четири параметра. Прва три означавају интензитет црвене, зелене и плаве боје у боји коју задајемо. Четврти параметар одређује транспарентност изабране боје. Вредности сва четири параметра припадају интервалу $[0.0,1.0]$.

Припрема

Пре исцртавања потребно је поставити боју позадине екрана. Прво се бира боја командом

`glClearColor(double r, double g, double b, double alpha).`

Чишћење бафера врши се наредбом

`glClear(GLbitfield mask)`

где је $mask$ битовска мапа

(GL_COLOR_BUFFER_BIT, GL_DEPTH_BUFFER_BIT, GL_ACCUM_BUFFER_BIT, GL_STENCIL_BUFFER_BIT) или комбинација битовских мапа логичком операцијом or, а указије на бафере (*glClearColor, glClearIndex, glClearDepth, glClearStencil, glClearAccum*) које треба попунити одабраном бојом. Уколико је вредност параметра $mask = GL\_COLOR\_BUFFER\_BIT$ извршава се „чишћење“ бафера *glClearColor*и *glClearIndex* изабраном бојом.

Пре цртања још је потребно дефинисати који ће део модела бити приказан на екрану. То се врши функцијом

`glOrtho(double left, double right, double bottom, double top, double near, double far).`

На екрану ће бити приказан део модела коме је x координата из интервала $[left, right]$, y координата из интервала $[bottom, top]$, а z координата из интервала $[near, far]$. У случају дводимензионалних модела, z кооридате свих тачака које се креирају су 0.0, тако да се за вредности параметара $near$и farмогу изабрати -1.0 и 1.0, респективно.

Функције `glMatrixMode(GL_PROJECTION)` и `glLoadIdentity()` биће објашњене у оквиру наредних поглавља.

```
void init (void)
{
    glClearColor (0.0, 0.0, 0.0, 0.0);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(0.0, 1.0, 0.0, 1.0, -1.0, 1.0);
}
```

Цртање квадрата

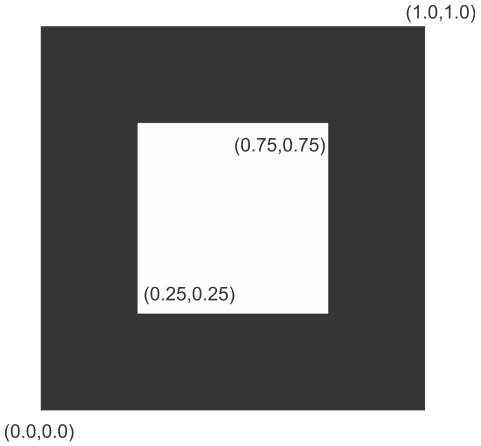
У примеру цртања квадрата врши се исцртавање полигона, и то у функцији `display()` креираној од стране корисника. Цртање објеката у ***OpenGL***-у врши се цртањем примитива (тачака, линија и полигона). Дефинисање боје објекта који се црта врши се пре дефинисања објекта функцијом

`glColor3f(float red, float green, float blue).`

Када се једном постави боја за цртање, сви објекти ће бити цртани том бојом док се она не промени поновним позивом са другим аргументима. Опис параметара функције је исти као и код функције `glClearColor()`. Навођење тачака (чворова) објекта врши се између команди `glBegin(GLenum mode)` и `glEnd()`, где параметар `mode` означава начин исцртавања

објекта. Свако навођење чворова ван овог опсега биће занемарено. О начинима исцртавања објеката биће детаљније речи на наредним часовима. При цртању квадрата у датом примеру коришћен је мод полигон, па се црта полигон дефинисан тачкама између команди `glBegin(GL_POLYGON)` и `glEnd()`. Чворови се дефинишу функцијом `glVertex3f(double x, double y, double z)`, где су x, y и z координате тачке која се дефинише. У задатом проблему цртања квадрата, креиране су четири тачке, где је приликом дефинисања тих тачака вођено рачуна о томе да добијени полигон буде баш квадрат.

```
glBegin(GL_POLYGON);
    glVertex3f (0.25, 0.25, 0.0);
    glVertex3f (0.75, 0.25, 0.0);
    glVertex3f (0.75, 0.75, 0.0);
    glVertex3f (0.25, 0.75, 0.0);
glEnd();
```



Да бисмо добили слику на екрану потребно је функцији `glutDisplayFunc(void (*func)(void))` проследитифункцију у којој се позива дефинисање свих објеката за исцртавање.

```
glutDisplayFunc(display)
```

Та функција се позива сваки пут при поновном исцртавању екрана. Поновно исцртавање екрана може бити позвано од стране система, нпр. при промени величине прозора, а може се захтевати и директно командом `GlutPostRedisplay()`. Иницијално исцртавање објеката као и обрада догађаја (тастатура, миш) почиње позивом функције `GlutMainLoop()`(*event processing loop*). Од тог тренутка се може обрадити сваки догађај миша или тастатуре и одреаговати на тај догађај, нпр. померити модел, променити боја модела и слично.

Преглед фунцкција:

- ***glClearColor(GLclampf red, GLclampf green, GLclampf blue,GLclampf alpha)*** – Постављање боје којом се може напунити неки од бафера. Сви параметри су из интервала [0,1].

Прва три параметра представљају интезитет црвене, зелене и плаве боје у боји која се дефинише. Уколико су сва три интезитета 1.0, добија се бела боја, а уколико су 0.0 добија се црна боја. Четврти параметар представља транспарентност (0.0 – потпуна прозирност, 1.0 – потпуна непрозирност)

- ***glClear(GlbitfieldmaskGlbitfield mask)*** – Испуњавање бафера ***mask*** бојом која је постављена командом ***glClearColor()***.
- ***glColor3f(float red, float green, float blue)*** – Постављање боје за исцртавање објеката – три параметра(red, green, blue). Сви параметри су такође из интервала [0,1].
- ***glutDislayFunc(void (*func)(void))*** – Постављање ***callback*** функције екрана. Функција која се позива приликом исцртавања екрана.
- ***GlutPostRedisplay()***–Позивање поновног исцртавање екрана.
- ***GlutMainLoop()***–event processing loop. Почиње исцртавање и обрада догађаја.

Програмски код

```
#include <GL/freeglut.h>

void display(void)
{
      glClear (GL_COLOR_BUFFER_BIT);
      glColor3f (1.0, 1.0, 1.0);

      glBegin(GL_POLYGON);
      glVertex3f (0.25, 0.25, 0.0);
      glVertex3f (0.75, 0.25, 0.0);
      glVertex3f (0.75, 0.75, 0.0);
      glVertex3f (0.25, 0.75, 0.0);
      glEnd();

      glFlush ();
}

void init (void)
{
    glClearColor (0.0, 0.0, 0.0, 0.0);
    glMatrixMode(GL_PROJECTION);
    glLoadIdentity();
    glOrtho(0.0, 1.0, 0.0, 1.0, -1.0, 1.0);
}

int main(int argc, char** argv)
{
    glutInit(&argc, argv);

    glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize (250, 250);
    glutInitWindowPosition (100, 80);
    glutCreateWindow ("hello");

    init ();
```

```
    glutDisplayFunc(display);
    glutMainLoop();

    return 0;
}
```