Факултет инжењерских наука Универзитета у Крагујевцу

Бојан В. Николић

Софтвер за графички приказ симулације кретања летелице

завршни рад

Крагујевац, 2015.

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: 712/2014

Бојан В. Николић

Софтвер за графички приказ симулације кретања летелице

завршни рад

Комисија за преглед и одбрану:

1. **Др Ненад Филиповић, проф. - ментор**
2. _____
3. _____

Датум одбране:_____

Оцена:_____

У оквиру овог завршног рада кандидат треба да

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

Препоручена литература:

[1] .....

[2] .....

[3] .....

...

Крагујевац, датум

ментор:

Др Ненад Филиповић, проф.

_____

Резиме рада: У овом раду коришћењем Visual Studio 2015 развојног окружења и програмског језика C# је приказан развој задате апликације . Представљен је поступак креирања апликације која симулира летелицу која се креће лево, десно, горе и доле притом на њу налецу разни објекти. Циљ је да летелица избегене или уништи пуцањем објекте. Рад је употпуњен графичким приказом корака од почетка израде, па све до стартовања и тестирања апликације.

Кључне речи: Visual studio 2013, c#, OPENGL, Windows, Објектно орјентисано програмирање.

Summary: In this paper, using the Visual Studio 2015 development environment and programming language C# shows the evolution of the default applications . It was introduced in the process of creating an application that simulates the spacecraft that is moving left, right , up and down while it runs into various objects . The goal is to craft izbegene or destroyed by shooting objects. The work is complemented by a graphic representation of the steps from the beginning of creation , to start up and testing applications.

Keywords : Visual studio 2013, c++, OPENGL, Windows, Object-Oriented Programming.

САДРЖАЈ

УВОД

Рачунарски систем се састоји од хардвера и софтвера. Хардвер чине делови рачунара, а софтвер програми које рачунар извршава. Рачунар је машина без интелигенције, он извршава само оно што му је задато и то на начин како му је задато. Програмом задајемо рачунару начин на који извршава послове. Да бисмо решили неки проблем коришћењем рачунара, морамо рачунару прецизно описати све кораке- инструкције (наредбе) које он извршава задатим редоследом. Скуп инструкција написан за решавање неког проблема назива се програм, а писање инструкција програмирање.

Програмирање или рачунарско програмирање јесте вештина помоћу које корисник ствара и извршава алгоритме користећи одређене програмске језике да би направио рачунарски програм. Програмирање садржи елементе уметности, науке, математике и конструисања. А основно средство за комуницирање и рачунарско програмирање је језик.

Сваки језик користи другачији скуп правила и синтаксу која одређује правило за грађење одређене наредбе. Успех комуникације између учесника у процесу комуникације зависи од језика који мора бити разумљив. Начин комуникације између човека и компјутера је специфичан, тако да директна комуникација између човека и компјутера није могућа. Језик компјутера састоји се из симбола (0 и 1), а језик човека је богат и разноврстан.

Оваквим језиком је потребно одрадити апликацију која симулирати игрицу покретање авиона у присуству разних објеката. Такве апликације нису новина, па ћу у наредном раду приказати једну такву апликацију.

1
ПРОГРАМИРАЊЕ И ПРОГРАМСКИ ЈЕЗИЦИ

Програмирање је значајна компонента, присутна у готово свим гранама рачунарства. Често изједначавање рачунарске науке са програмирањем занемарује друге битне аспекте рачунарства као што су пројектовање хардвера, архитектура система, пројектовање нивоа оперативних система, структуирање база података за специфичне апликације, валидације модела, итд. Програмирање је у најопштијем смислу активност израде програма за електронску рачунску машину (рачунар).

Програмирање, поред изучавања важних класа алгоритама и програма који се, као честе компоненте других програма могу примменити у решавању нових задатака, мора да укључи и методе развоја алгоритама и програма са унапред познатим понашањем, тј. Методе које омогућују да се, заједно са извођењем програма изводи и доказ његове коректности (његова семантика), што приближава развој програма доказу теореме у математици.Понашање програма сада се изводи из самог текста програма, без интерпретације извршног кода, тј. без тестирања. Последњих деценија, програмски системи код којих је критична безбедност (нпр. контрола лета), развијају се управо коришћењем оваквих метода. Оне (за разлику од тестирања) могу егзактно да докажу коректност програма, али и да помогну у откривању евентуалних грешака.

У мањим развојним пројектима фазе креирања програма нису одвојене и изводе се синхроно - реализација зависи од концепције и обратно. У већим развојним пројектима су те фазе прилично одвојене. Фаза израде концепта се у тим случајевима назива дизајн, а реализације имплементација.

Програмерима се тада, када су циљеви и методе реализације у претходној фази (дизајну и спецификацији) прецизно одређени, даје мање слободе при имплементацији и његова креативност се заснива првенствено на проналажењу најбољих и најефикаснијих алгоритама и метода реализације појединих задатака (израде компоненти), као и оптимизирање рада тих компоненти.1

1.1 Програмско окружење

Било ко ко жели да програмира суочава се са проблемом избора такозваног програмског окружења. Под програмским окружењем подразумева се скуп софтверских алата помоћу којих програмер пројектује и развија програме. Програмско окружење увек укључује едитор, компајлер, библиотеку предходноразвијених програма, дибагер (Debugger). Поред овога,програмеру могу бити на располагању и други софтверски алати као што су алатиза

анализу и пројектовање система (UML, на пример), алати за графички дизајн корисничког интерфејса итд.

Могу се разликовати две врсте програмских окружења: интегрисана окружења(integrated development enviroment, или IDE скраћено) и линијски едитори команди. Интегрисано окружење је графички кориснички интерфејс који садржи све потребне алате за развој програма као што су едитор, компајлер, графички дизајнер (за форме и сл..), дибагер, појект менаџер итд. Окружење са линијским едитором команди је једнсотавно окружење где програмер коришћењем једноставних текст едитора уноси наредбе програмског кода, наредбе за компилацију и извршавање.

Данас се за програмирање најчешће користе интегрисана окружења. За програмере почетнике се међутим препоручује коричћење једноставних линијских едитора како би програмери почетници могли да прате све кораке у изради програма који су у ИДЕ понекад сакривени. Наравно, ИДЕ су знатно погоднија за развој сложених програмских система јер укључују и елементе управљања пројектима.2

1.2 Програмски језици

Програмски језик је средство којим човек саопштава рачунару програм. Човек и рачунар комуницирају помоћу програмског језика. Приридни језици допуштају неједнозначност и непрецизност. Рачунар може „разумети“ само формални запис, не толеришући ни најмање непрецизности. Записи низа инструкција у програмском језику не могу се вишезначно протумачити. Програмски језик је скуп правила којим се рачунару представљају инструкције и описују подаци. Делимо их по степену зависности програмског језика и рачунара на:

- Машински језик
- Симболички језик
- Језици вишег нивоа

Машински, симболички и макро језици су зависни од рачунара –машински зависни језици. Они су тесно повезани за конкретну машину и узимају у обзир специфичности архитектуре одговарајућег процесора. То су језици ниског нивоа.

Језици вишег нивоа припадају групи машински независних језика. Граде се независно од рачунара на коме ће се извршавати. Они су средство у коме програмер записује своје идеје не упуштајући се у детаље архитектуре рачунара. Машински независни језици су намењени применама у различитим делатностима и за сваку од тих примена граде се посебне класе програмских језика високог нивоа.1

1.3 Visual studio 2013 развојно окружење.

Visual studio 2013 је развојни алат, потписан од стране Microsoft корпорације, који корисницима омогућује креирање широког спектра апликација и као такав засигурно представља најкомплетније ИДЕ (integrated development environment) окружење данашњице. Његовим коришћењем можемо развијати мобилне апликације, апликације за destop и cloud окружења, разне врсте сервиса, web апликације итд. Visual Studio подржава различите програмске језике и дозвољава уреднику кôда и debuggeru да подржава (у различитој мери) скоро било који програмски језик, под условом да сервис за тај језик постоји. Уграђни језици су C, C++ и C++/CLI (преко Visual C++), VB.NET (преко Visual Bsic.NET)-a, C# (преко Visual C#) и f# (почевши од програма Visual Studio 2010). Подршка за остале програмске језике попут M, Python, и Rybu-ja као и осталих је доступан инсталацијом језичких сервиса који се могу засебно инсталирати. Такође подржава XML/XSLT, HTML/XHTML, JavaScript и CSS.[3]

2
OPENGL biblioteka

У наредном пројекту коришћен је додатни алат OpenGl (енг. Open Graphic Library) назив је једног од најраширенијих АПИ-а (Application Programming Interface) за развој богатих 2D/3D апликација. Првобитно развијен од стране Silicon Graphics Inc. Данас његов изворни кôд доступан је за преузимање и даље прилагођавање потребама девелопера и крајњих корисника. Његова примена сеже од CAD (Computer Aided Design) програма до хитова из индустрије игара.

Откад је изашао у 1992. години, био је водећи API за развој графичких апликација. Убрзо након изласка Windows-а '95, Microsoft (првобитно члан АРБа) издаје свој мултимедијални API за развој богатих 3D апликација специфично за свој операциони састав с циљем да конкурише OpenGlu-u и привуче више корисника на Windows OS. Коначно повлачење Microsofta из OpenGl-a ARB-а догађа се 2003. године. Предност OpenGl-а лежи у чињеници да је OpenGl независан о платформи на којој се користи док му је евентуална мана то што је OpenGl графички АПИ, за разлику од DirectXa који се састоји од више модула (графика, video streaming, audio, networking, input). Но, OpenGl се лагано може упарити са веома квалитетном алтернативном библиотеком, SDL (Simple DirectMedia Layer), која му омогућује лакшу контролу над улазом корисника, манипулише звуком и др.

Овај текст специфично ће се бавити имплементацијом OpenGl-а у ваше апликације те ће настојати понудити квалитетни увод у програмирање 3D апликација са OpenGl –ом.у. Прво ћемо поставити OpenGl и „Синхронизовати“ га с Visual Studiom, а након тога ћемо кренути у занимљив свет дизајнирања и програмирања 3D апликација користећи OpenGl АПИ.[4]

2.1 ПОСТАВЉАЊЕ OpenGl-а.

OpenGl за разлику од DirectXa нема поједностављену инсталацију која ће одмах покренути податке на рачунару и додати потребне директоријуме у Visual C# Directories, него је све потребно направити ручно. Нема потребе за претрагом хрпе страница за ГЛУТ-ом (Graphics Library Utility Kit) којег ћемо специфично користити за развој ове апликације, већ скинути са интернета бесплатну верзију.

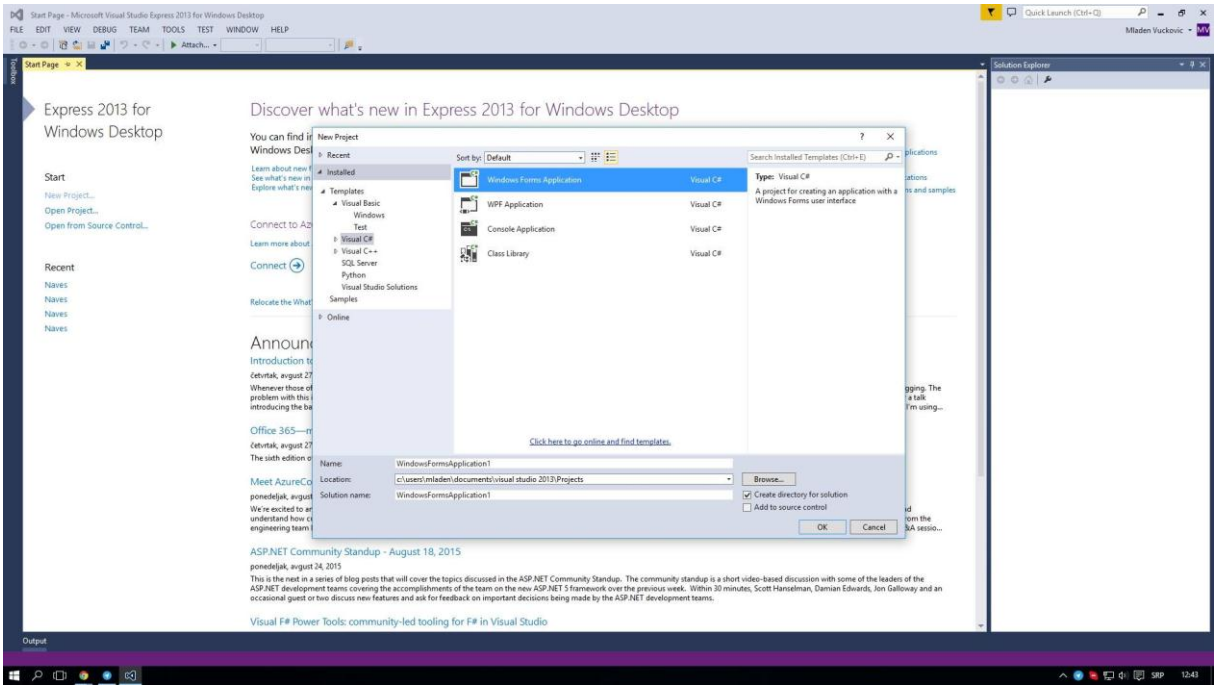
OpenGl Utility Kit (GLUT) је библиотека комуналних услуга за OpenGl програме , које првенствено обављају систематски ниво I/O са главног оперативног система . Резултати те фукнције су прозор , контролни прозор и праћење тастатуре и миша улаза .

Постуно ћу објаснити корак по корак.

- Распакована је архива у рачунару, па направљен фолдер негде на HDD-у (нпр. C:\GLUT) те у њему направљено два засебна фолдера „include“ и „lib“.
- Отворите include, па у њему направити још један фолдер назван „GL“. Ту је копиран из распаковане архиве Фајла зван glut.x, а остали фајлови постављено је у Lib фолдер.
- Покренут је Visual Studio 2013
- Отворити Tools -> Options
- Сада на Project and Solutions -> VS# Directories, горе из падајућег менија одабран је Include Files и додат је нови део (path) и онда кликом на адресу (C:\GLUT\include\GL\)
- Након тога идемо поново на падајући мени и одаберемо Library files. Сада се понавља претходни поступак само ићи на директорију Lib(C:\ГЛУТ\либ\)
- И то је то, постављен је OpenGl.

3
ИЗРАДА АПЛИКАЦИЈЕ

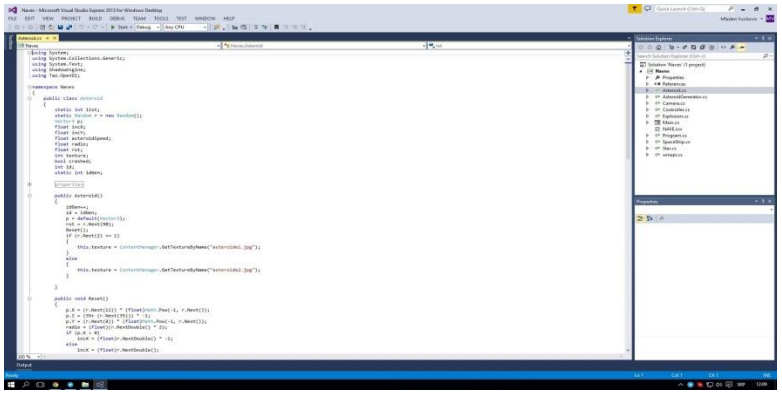
Покренут је Visual Studio 2013, кликом Windows form aplication. Затим се врши упис имена апликације. Слика 1.



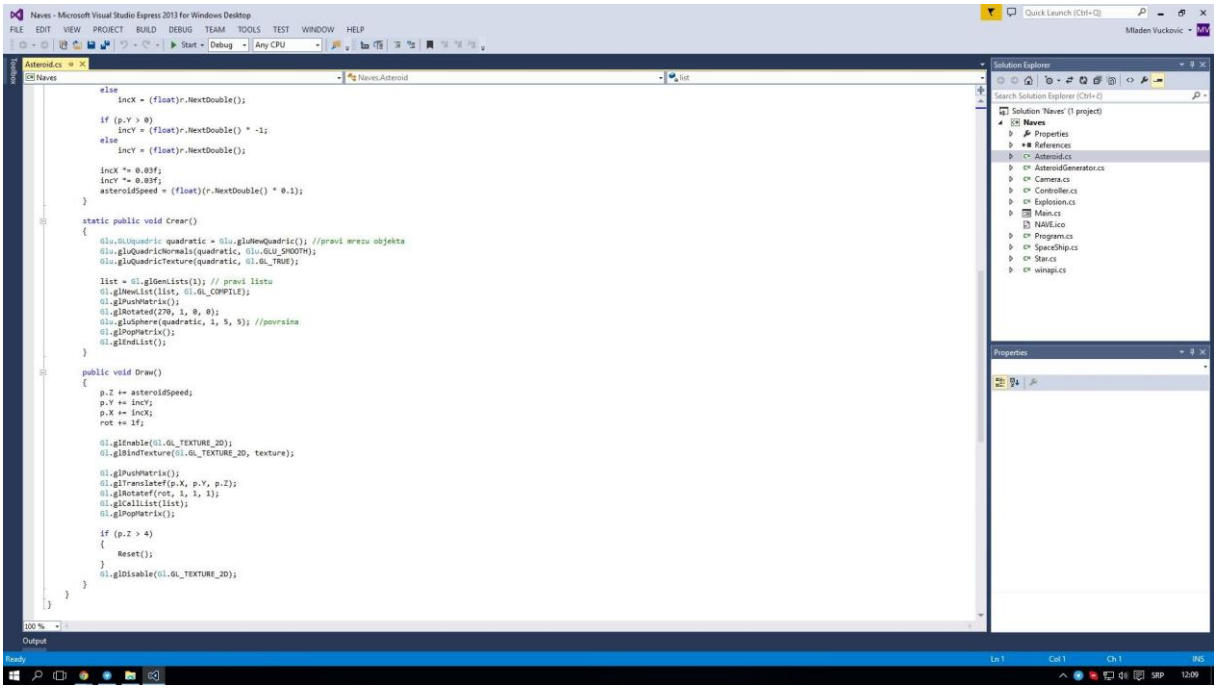
Слика 1.

3.1 Код Апликације

Сам код је рађен по класама, прва класа кода је израда астероида, облика и кретања и боје. Слика 2. И Слика 3.

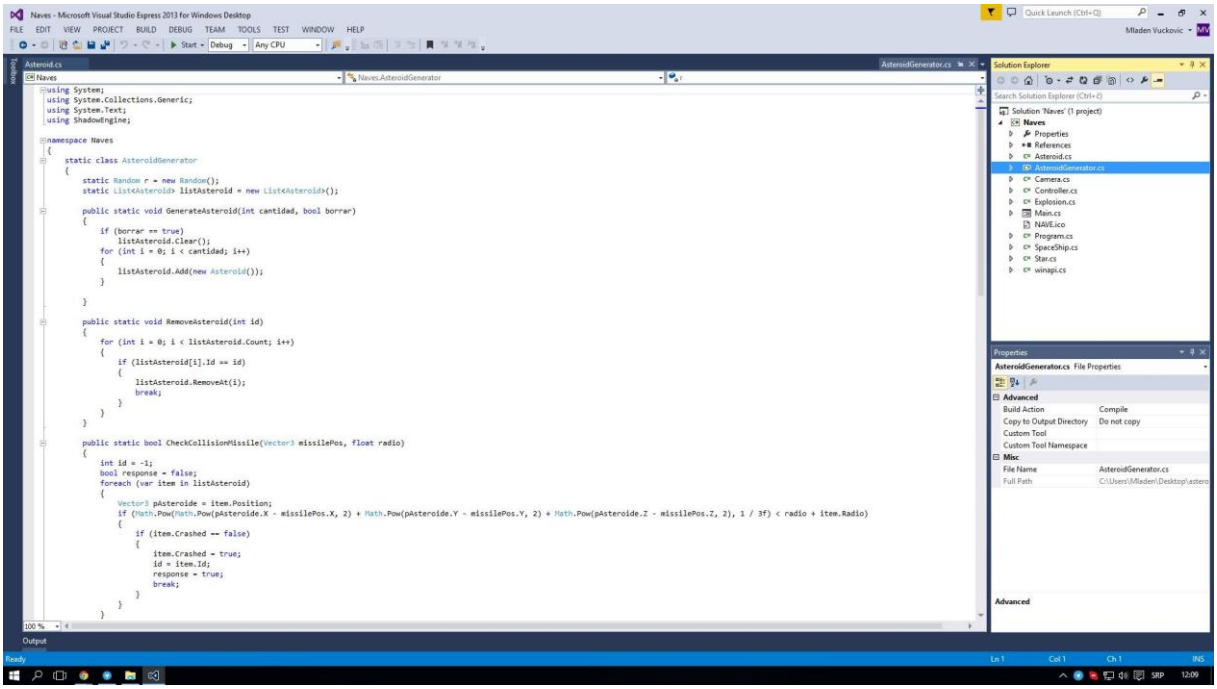


слика 2.

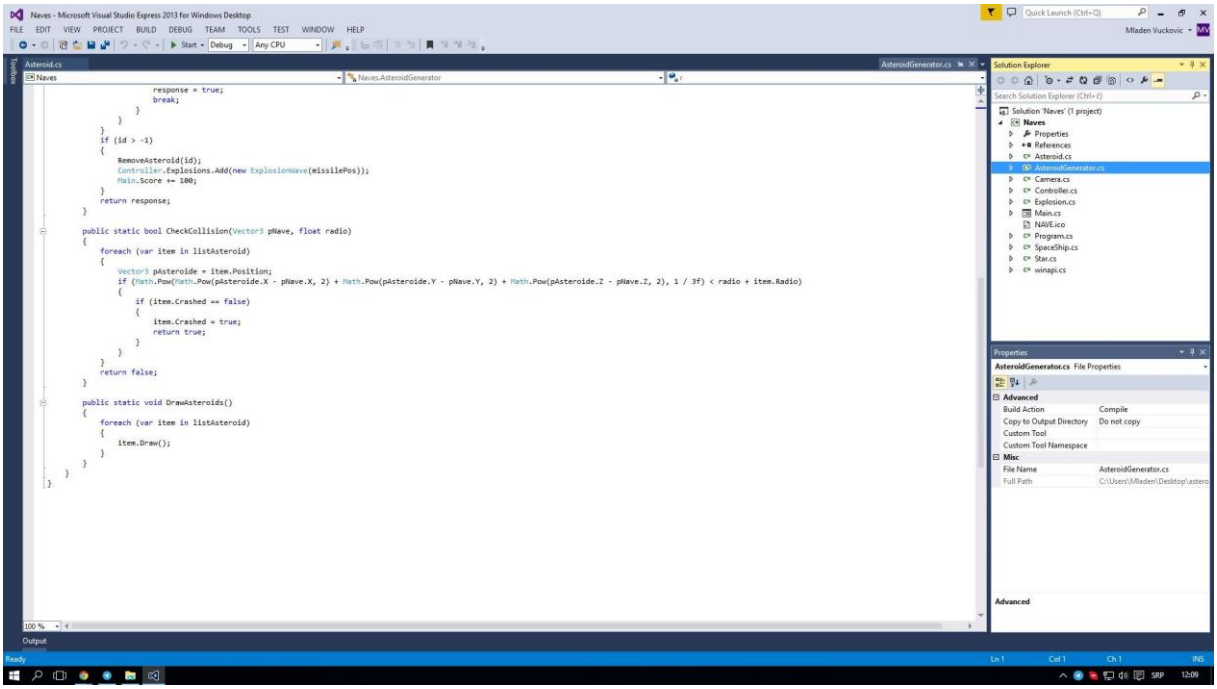


Слика 3.

Следећа класа је код за поновно исцртавање астероида и повећавање броја у зависности од нивоа. Слика 4. И слика 5.

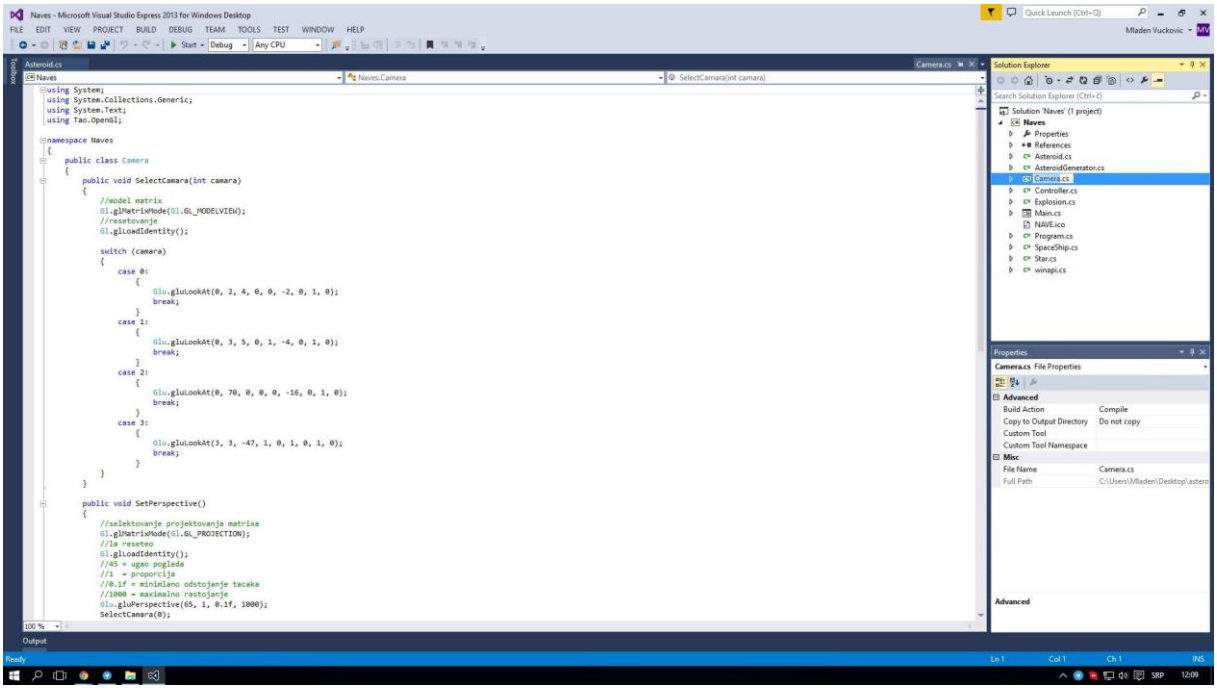


Слика 4.



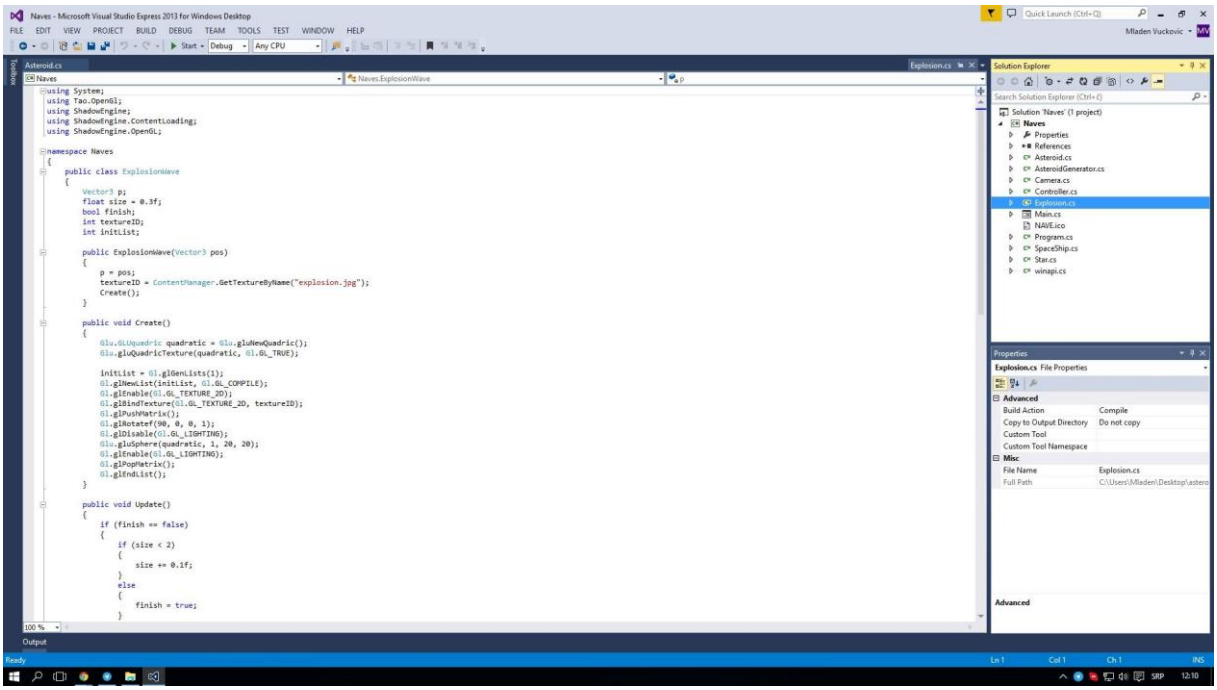
Слика 5.

У следећем делу одрађен је код за креирање камере и саму визуализацију. Слика 6.

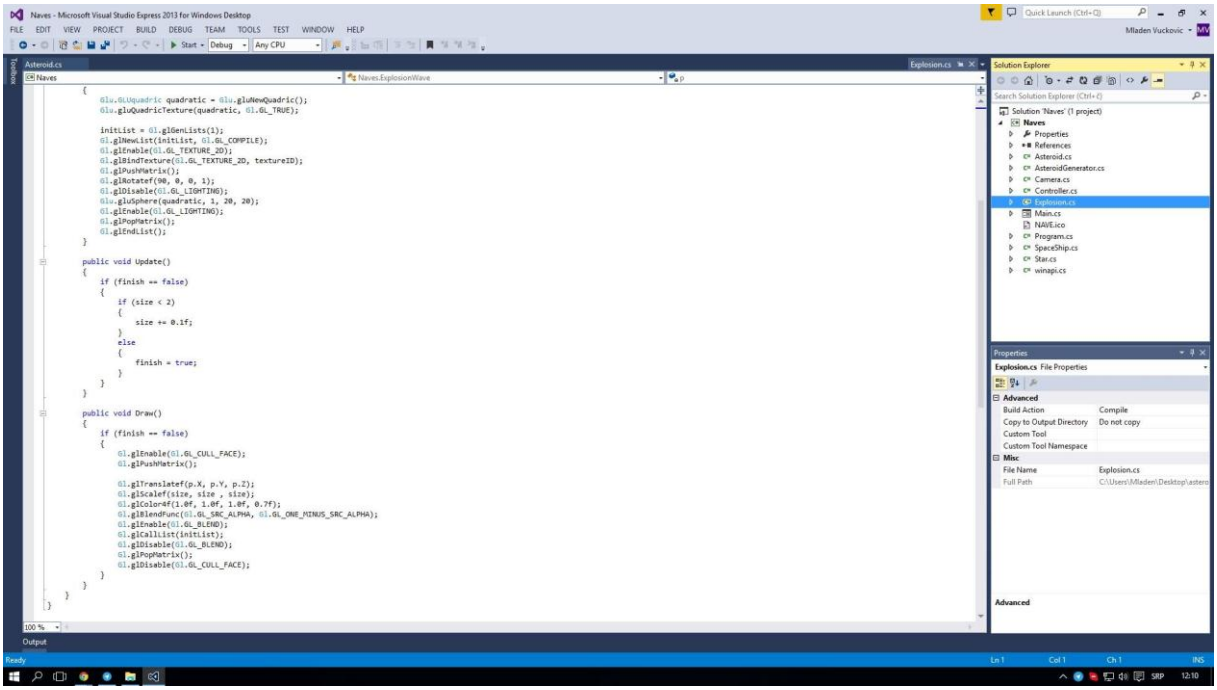


Слика 6.

Наредна класа описује код који ствара експлозије. Слика 7. И слика 8.

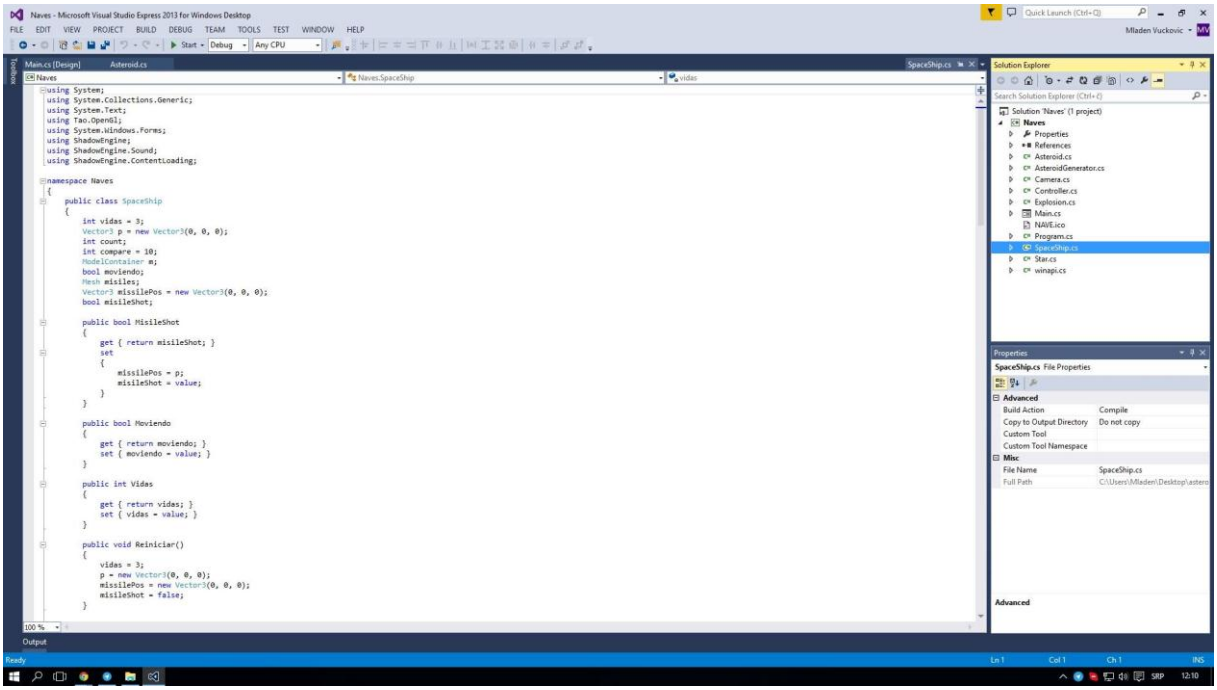


Слика 7.



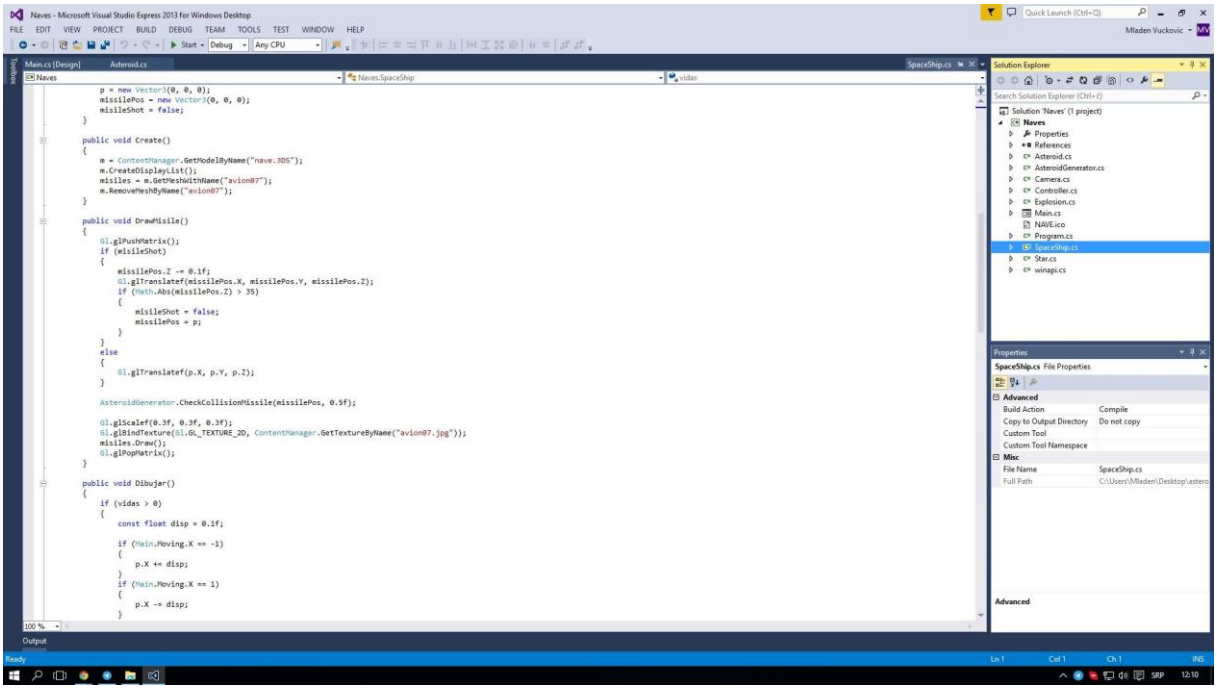
Слика 8.

У наредном коду је креиран авион и изглед авиона. Слика 9, слика 10, слика 11.



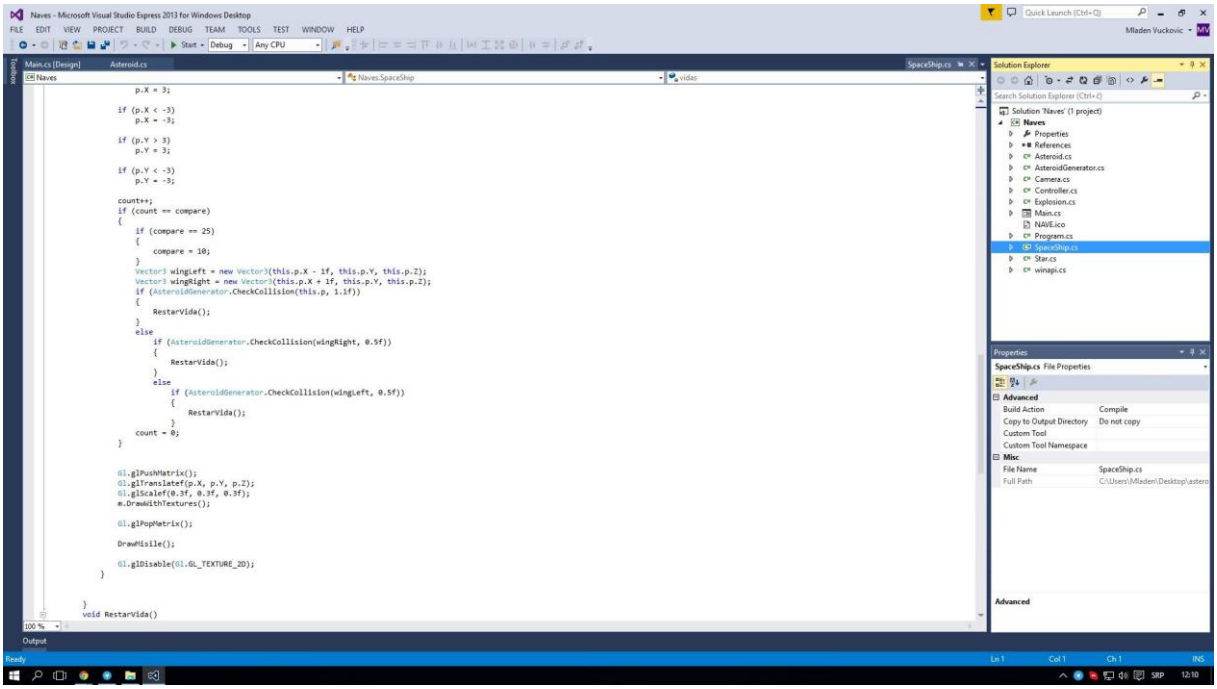
```
using System;
using System.Collections.Generic;
using System.Text;
using Tao.OpenGl;
using System.Windows.Forms;
using ShadowEngine;
using ShadowEngine.Sound;
using ShadowEngine.ContentLoading;

namespace Naves
{
    public class SpaceShip
    {
        int vidas = 3;
        Vector3 p = new Vector3(0, 0, 0);
        int count;
        int compare = 10;
        ModelContainer m;
        bool moviendo;
        Mesh misiles;
        Vector3 missilePos = new Vector3(0, 0, 0);
        bool misileShot;

        public bool MisileShot
        {
            get { return misileShot; }
            set
            {
                missilePos = p;
                misileShot = value;
            }
        }

        public bool Moviendo
        {
            get { return moviendo; }
            set { moviendo = value; }
        }

        public int Vidas
        {
            get { return vidas; }
            set { vidas = value; }
        }

        public void Reiniciar()
        {
            vidas = 3;
            p = new Vector3(0, 0, 0);
            missilePos = new Vector3(0, 0, 0);
            misileShot = false;
        }
```

Слика 9.



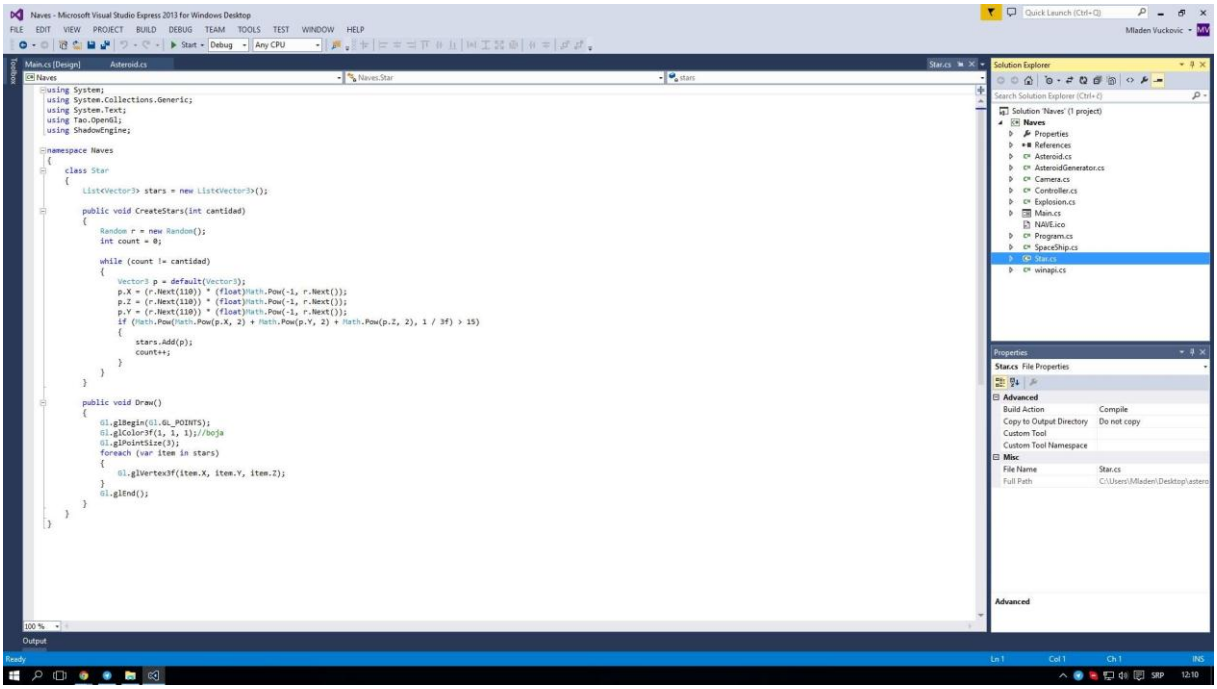
```
            p = new Vector3(0, 0, 0);
            missilePos = new Vector3(0, 0, 0);
            misileShot = false;
        }

        public void Create()
        {
            m = ContentManager.GetModelByName("nave.3DS");
            m.CreateDisplayList();
            misiles = m.GetMeshWithName("avion07");
            m.RemoveMeshByName("avion07");
        }

        public void DrawMisile()
        {
            Gl.glPushMatrix();
            if (misileShot)
            {
                missilePos.Z -= 0.1f;
                Gl.glTranslatef(missilePos.X, missilePos.Y, missilePos.Z);
                if (Math.Abs(missilePos.Z) > 35)
                {
                    misileShot = false;
                    missilePos = p;
                }
            }
            else
            {
                Gl.glTranslatef(p.X, p.Y, p.Z);
            }

            AsteroidGenerator.CheckCollisionMissile(missilePos, 0.5f);

            Gl.glScalef(0.3f, 0.3f, 0.3f);
            Gl.glBindTexture(Gl.GL_TEXTURE_2D, ContentManager.GetTextureByName("avion07.jpg"));
            misiles.Draw();
            Gl.glPopMatrix();
        }

        public void Dibujar()
        {
            if (vidas > 0)
            {
                const float disp = 0.1f;

                if (Main.Moving.X == -1)
                {
                    p.X += disp;
                }
                if (Main.Moving.X == 1)
                {
                    p.X -= disp;
                }
```

Слика 10.



Слика 11.

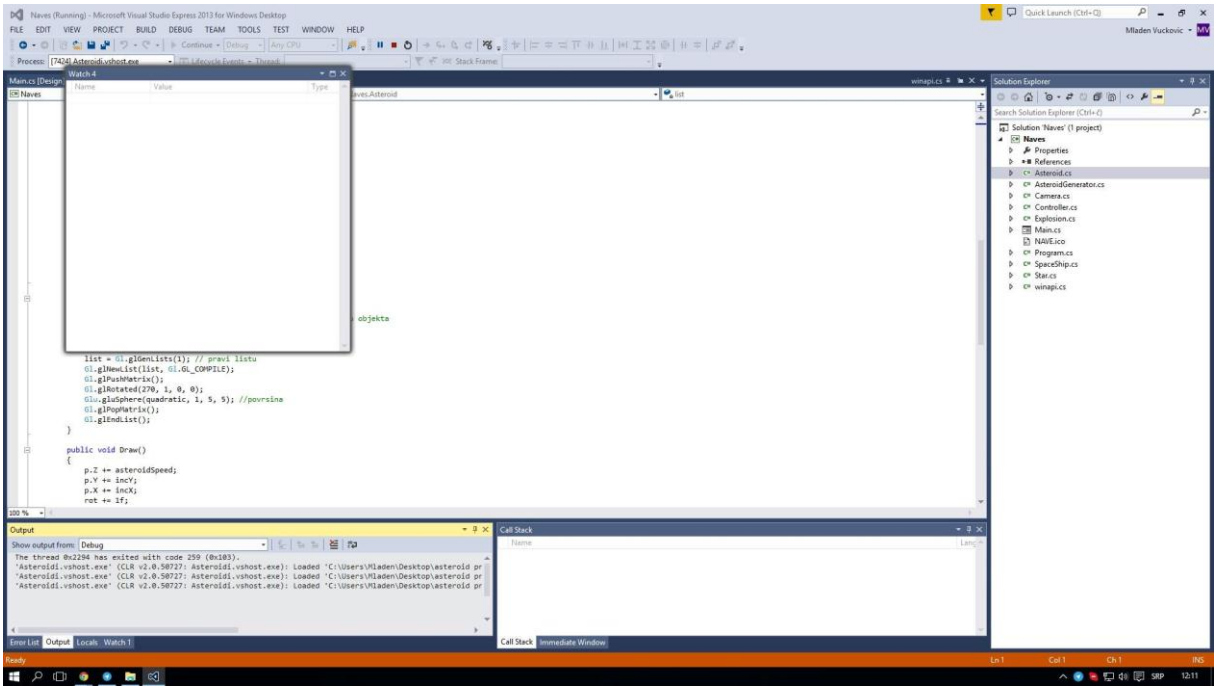
За креирање позадине и звезда у позадини куцан је следећи код. Слика 12.



Слика 12.

3.2 Build апликације

После израде кода апликације, мора се сачувати код. Затим клик на дугме Build (Слика 13). Уколико нема грешака иде се на опцију Debug а затим Start Debug.



Слика 13.

4
ТЕСТИРАЊЕ И ВАЛИДАЦИЈА АПЛИКАЦИЈЕ

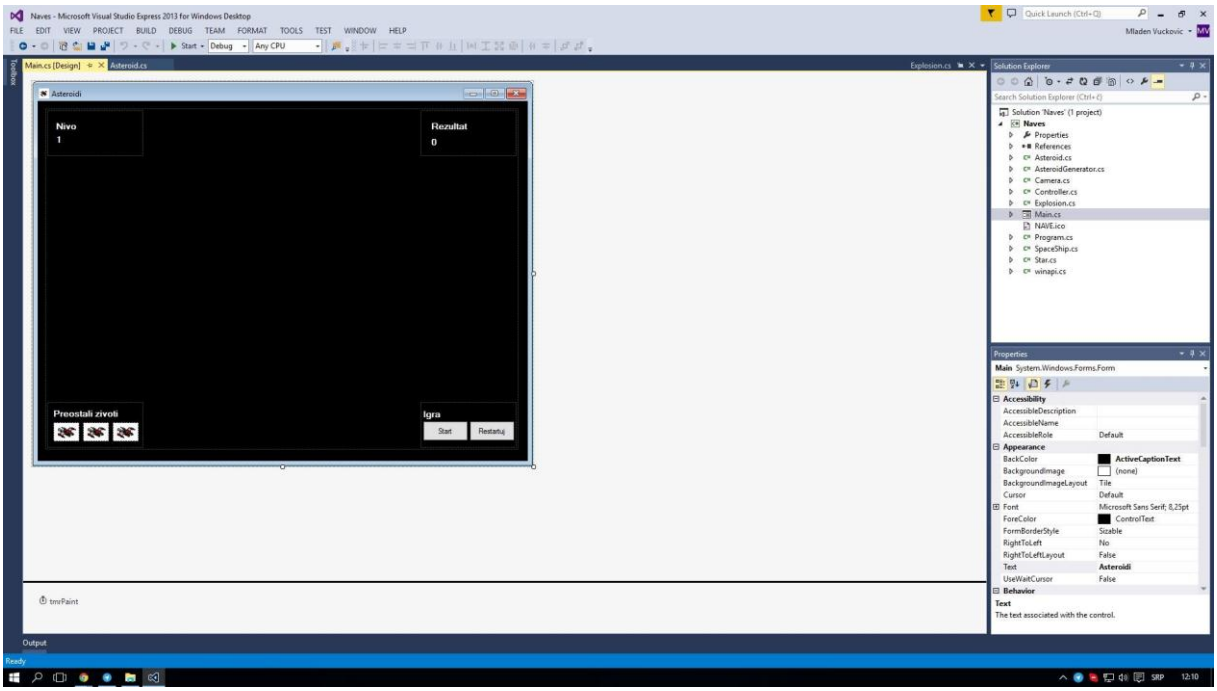
Сваки програмски производ па и ова симулација игрице која пушта летелицу у простор треба да прође и кроз фазу тестирања и валидације. Намена тестирања је да покаже да апликација ради онозаште је намењена и да утврди апликационе дефекте пре него што се стави у употребу. Када се тестира програмирани производ, тада се извршава тест, где се тестирање врши вештачким подацима. Тестирање је генералнији део свих процеса валидације, који укључују и статичке технике валидације. Резултати теста треба да покажу грешке, аномалије или друге информације о апликацији које указују на непредвиђењо понашање. [5]

Апликација се покреће коришћењем тест података где се помно прати њено понашање и анализирају сви детаљи који могу да покажу одређено неприкладно понашање, и на крају теста систем треба да покаже да може да обезбеди одређене функционалности, перформансе и да је стабилан у току нормалне употребе.

Тестирање апликације у овом случају би требало да открије грешке или отказе, и прикаже да ли постоји неко неприкладно понашање које није дефинисано спецификацијом.Постоје неколико врста тестирања:

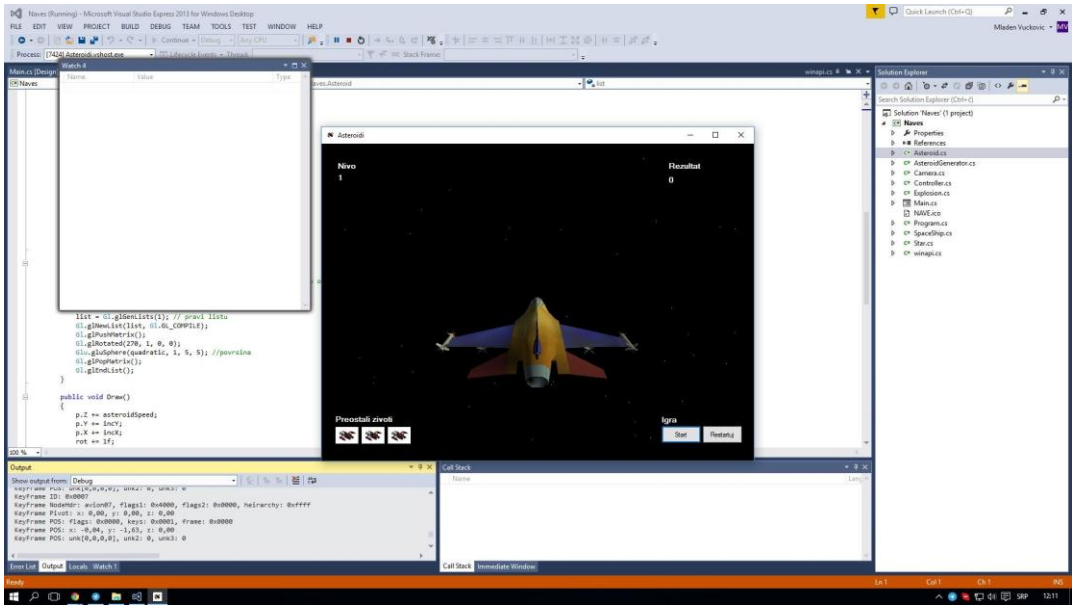
- Развојно тестирање-тестира се у току развоја производа да би се открили багови или други дефекти**.** Врсте развојног тестирања:
  - Тестирање јединица, тестирају се индивидуалне програмске јединице и класе.
  - Тестирање система, систем се тестира као целина.
  - Тестирање компоненти, неколико посебних јединица се интегришу како би креирале сложене компоненте. Фокус тестирања је тестирање итерфејса.
- Тестирање производа-одређен тим тестира комплетну верзију апликације пре него што се она испоручи у одређеном окружењу за тестирање.
- Корисничко тестирање-корисници, или потенцијални корисници апликације тестирају производ, и овај начин је најбитнији, разлог је тај што корисниково радно окружење има велике ефекте на поузданост, перформансе, употребљивост, итд., а ово се не може рефлектовати у окружењу за тестирање. Врсте корисничког тестирања:
  - Алфа тестирање, корисници раде са развојним тимом да би тестирали софтвер на развојним страницама.
  - Бета тестирање, корисницима се испоручује производ како би могли да експериментишу са њим и установе проблеме које развојни тим није пронашао.
  - Тестирање прихватљивости, корисници тестирају систем да би одлучили да ли је спреман да буде прихваћен у корисничко окружење за употребу. [5]

На самом почетку тестирања прво отворити фаил где је сачуван целокупан рад. Ту се налази иконица „asteroid.exe“. Двокликом на њу покрећемо нашу апликацију „Asteroid“. Отвара се прозор са почетком игрице. Слика .



Слика

Кликом на дугме Start покрећемо игрицу и вршимо проверу свих покрета и захтева. Слика



Слика

ЗАКЉУЧАК

У овом задатку анализиран је начин креирања апликације, 3D симулације авиона у простору, као и алатке које су коришћене. Начин рада у програмском језику и самом програмирању. Програмирање је вештина израде рачунарских апликација која представља једно од најперспективнијих занимања и грана привреде у модерном друштву. Тежећи друштву знања и конкурентној привреди заснованој на знању.

Као сто каже један од највећи програмера до сад, Donald Knuth, процес стварања програма и апликација за компјутере је изузетно атрактиван, не само због своје економске и научне исплативости, већ и зато што може бити уметничко искуство налик компоновању поезије или музике.

ЛИТЕРАТУРА

[1] https://infossremac.wordpress.com/informatika-iii-i-iv/programiranje/

[2] Osnove programiranja Dr Milan Popović,

[3] www.wikipedia.rs

[4]http://www.bug.hr/forum/topic/programiranje/opengl-kroz-c--tutorial-/3643.aspx

[5] др Зоран Јеремић, *Тестирање софтвера и управљање квалитетом*