Analitička geometrija prostora u programskom jeziku C#

Benić, Leon

SVEUČILIŠTE JOSIPA JURJA STROSSMAYERA U OSIJEKU
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Sveučilišni studij

ANALITIČKA GEOMETRIJA PROSTORA U PROGRAMSKOM JEZIKU C#

Završni rad

Leon Benić

Osijek, 2023.

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

Obrazac Z1P - Obrazac za ocjenu završnog rada na preddiplomskom sveučilišnom studiju

Osijek, 25.08.2023.

Odboru za završne i diplomske ispite

Prijedlog ocjene završnog rada na preddiplomskom sveučilišnom studiju

Ime i prezime Pristupnika:	Leon Benić
Studij, smjer:	Računalno inženjerstvo
Mat. br. Pristupnika, godina upisa:	R4465, 27.07.2020.
OIB Pristupnika:	70230166773
Mentor:	doc. dr. sc. Tomislav Rudec
Sumentor:	izv. prof. dr. sc. Alfonzo Baumgartner
Sumentor iz tvrtke:	
Naslov završnog rada:	Analitička geometrija prostora u programskom jeziku C#
Znanstvena grana rada:	**Programsko inženjerstvo (zn. polje računarstvo)**
Zadatak završnog rad:	Student će izraditi aplikaciju za računanje osnovnih elemenata analitičke geometrije prostora u programskom jeziku C# Tema je zauzeta za Leona Benića. Sumentor s FERIT-a: Alfonzo Baumgartner.
Prijedlog ocjene završnog rada:	Izvrstan (5)
Kratko obrazloženje ocjene prema Kriterijima za ocjenjivanje završnih i diplomskih radova:	Primjena znanja stečenih na fakultetu: 3 bod/boda Postignuti rezultati u odnosu na složenost zadatka: 2 bod/boda Jasnoća pismenog izražavanja: 2 bod/boda Razina samostalnosti: 3 razina
Datum prijedloga ocjene od strane mentora:	25.08.2023.
Datum potvrde ocjene od strane Odbora:	08.09.2023.

Potvrda mentora o predaji konačne verzije rada:	*Mentor elektronički potpisao predaju konačne verzije.*
	Datum:

FERIT
FAKULTET ELEKTROTEHNIKE, RAČUNARSTVA I INFORMACIJSKIH TEHNOLOGIJA OSIJEK

IZJAVA O ORIGINALNOSTI RADA

Osijek, 08.09.2023.

Ime i prezime studenta:	Leon Benić
Studij:	Računalno inženjerstvo
Mat. br. studenta, godina upisa:	R4465, 27.07.2020.
Turnitin podudaranje [%]:	13

Ovom izjavom izjavljujem da je rad pod nazivom**: Analitička geometrija prostora u programskom jeziku C#**

izrađen pod vodstvom mentora doc. dr. sc. Tomislav Rudec

i sumentora izv. prof. dr. sc. Alfonzo Baumgartner

moj vlastiti rad i prema mom najboljem znanju ne sadrži prethodno objavljene ili neobjavljene pisane materijale drugih osoba, osim onih koji su izričito priznati navođenjem literature i drugih izvora informacija.
Izjavljujem da je intelektualni sadržaj navedenog rada proizvod mog vlastitog rada, osim u onom dijelu za koji mi je bila potrebna pomoć mentora, sumentora i drugih osoba, a što je izričito navedeno u radu.

Potpis studenta:

SADRŽAJ

1. UVOD 1

1.1. Zadatak završnog rada 1

2. POSTOJEĆA RJEŠENJA 2

2.1. ThinkCalculator 2

2.2. MathPortal 3

2.3. Open Omnia 3

2.4. OnlineMSchool 4

3. ANALITIČKA GEOMETRIJA PROSTORA 5

3.1. Ravnina 5

3.1.1. Opći oblik jednadžbe ravnine 5

3.1.2. Normalni oblik jednadžbe ravnine 6

3.1.3. Segmentni oblik jednadžbe ravnine 7

3.1.4. Jednadžba ravnine kroz tri točke 7

3.1.5. Dvije ravnine u prostoru 8

3.1.6. Udaljenost točke do ravnine 9

3.2. Pravac 9

3.2.1. Oblici jednadžbe pravca 9

3.2.2. Dva pravca u prostoru 11

3.2.3. Pravac i ravnina u prostoru 11

4. KORIŠTENE TEHNOLOGIJE 12

4.1. Programski jezik C# 12

4.2. Visual studio 13

4.3. Windows Forms 14

4.4. Canva 15

5. WINDOWS FORMS APLIKACIJA 16

5.1. Struktura aplikacije 16

5.1.1. Form1.cs 17

5.1.2. OsnovneOperacije.cs 18

5.1.3. AlgoritmiFirst.cs 19

5.1.4. AlgoritmiSecond.cs 20

5.2. Kut između dvije ravnine 21
5.2.1. Programski dio 21
5.2.2. Rješenje problema 22
5.3. Kut između ravnine i pravca 23
5.3.1. Programski dio 23
5.3.2. Rješenje problema 24
5.4. Prolazi li ravnina kroz ishodište? Pripada li točka ravnini? 25
5.4.1. Programski dio 25
5.4.2. Rješenje problema 26
5.5. Jednadžba pravca kroz dvije točke 27
5.5.1. Programski dio 27
5.5.2. Rješenje problema 29
5.6. Udaljenost točke od ravnine 29
5.6.1. Programski dio 29
5.6.2. Rješenje problema 29
5.7. Udaljenost između dvije točke 30
5.7.1. Programski dio 30
5.7.2. Rješenje problema 30
5.8. Udaljenost između ravnina 31
5.8.1. Programski dio 31
5.8.2. Rješenje problema 32
5.9. Jednadžba ravnine kroz tri točke 33
5.9.1. Programski dio 33
5.9.2. Rješenje problema 34
6. ZAKLJUČAK 35
LITERATURA 36
POPIS SLIKA 38
SAŽETAK 40
ABSTRACT 41

1. UVOD

Analitička geometrija prostora je grana geometrije u kojoj se geometrijski problemi prostora i ravnine rješavaju algebarskim metodama. Budući da algebarske metode znaju biti kompleksne i ručnim rješavanjem je lako doći do pogreške, teži se programskim rješenjima i aplikacijama za lakše računanje.

Problem ovog završnog rada je pojednostavniti i ubrzati rješavanje osnovnih operacija i algoritama analitičke geometrije prostora. Navedeni problemi završnog rada bit će izvedeni u obliku aplikacije sa kalkulatorom rješavanja.

Djelovanje aplikacije navedeno je u nekoliko poglavlja, koji se temelje na usporedbi sa već postojećim aplikacijama te pojašnjenjem teorijske podloge analitičke geometrije prostora. Zatim slijedi detaljan opis korištenih tehnologija i strukture aplikacije. U posljednjem poglavlju navodi se programsko rješenje za svaki problem te su prikazana rješenja osnovnih operacija u obliku kalkulatora.

1.1. Zadatak završnog rada

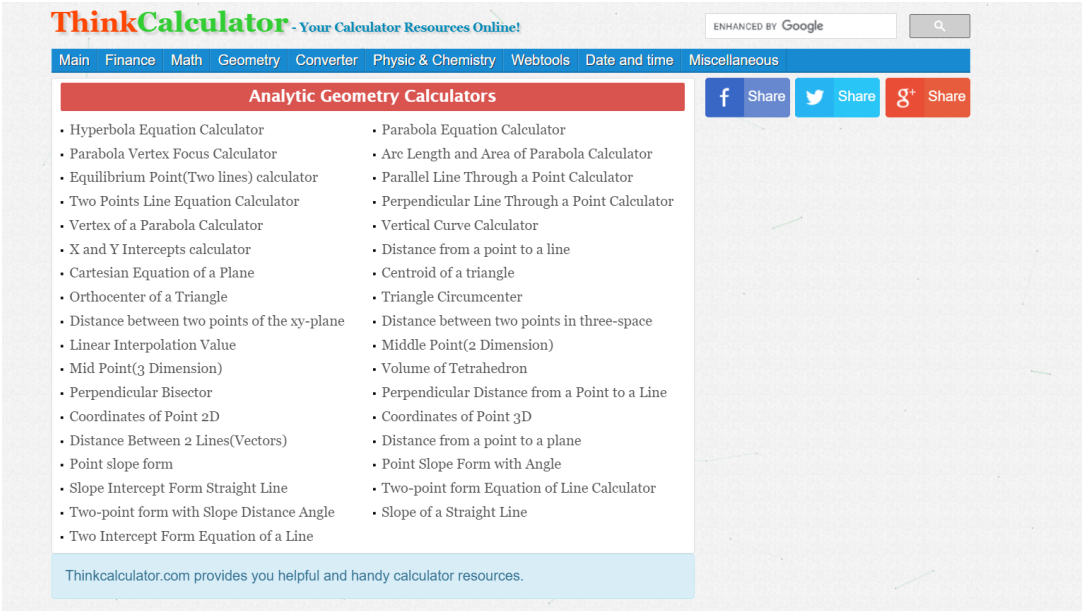
Zadatak završnog rada je napraviti aplikaciju za lakše rješavanje osnovnih problema analitičke geometrije prostora i provjera ručno riješenih zadataka. Također, cilj aplikacije je dati točna i brza rješenja na zadane probleme te navesti karakteristike korištenih tehnologija za izradu aplikacije.

2. POSTOJEĆA RJEŠENJA

Analitička geometrija prostora je jako širok pojam geometrije i računarstva, zbog toga postoji mnoštvo aplikacija koje su namijenjene učenju i bržem rješavanju zadataka. Analizirana su online rješenja pronađena na internetu.

2.1. ThinkCalculator

Prvi primjer je web aplikacija za rješavanje raznih matematičkih i ostalih problema koji zahtijevaju računanje nazvana *ThinkCalculator* [1]. Na slici 2.1 vidljivi su svi problemi koje aplikacija nudi za rješavanje.

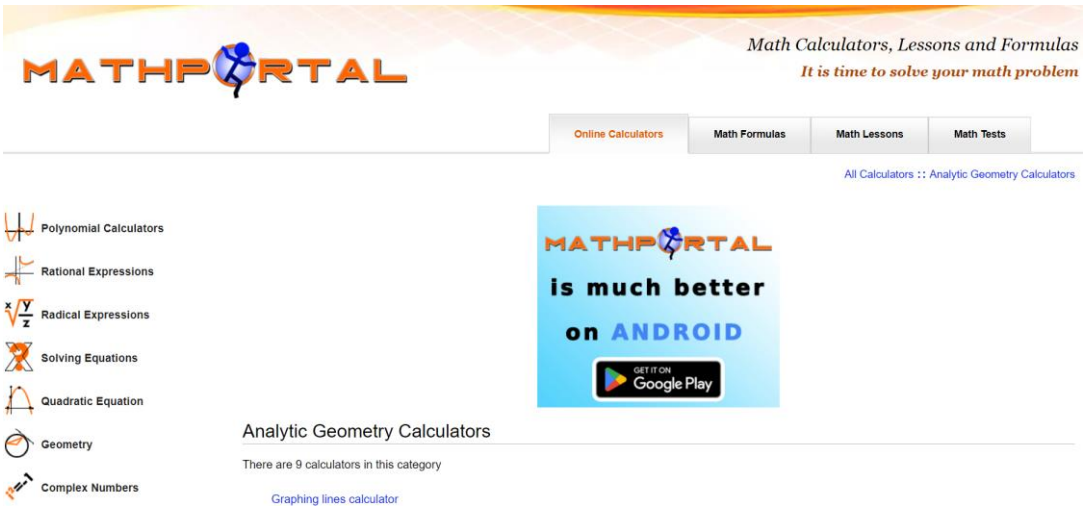


Slika 2.1. Izgled web aplikacije *ThinkCalculator*

Aplikacija nudi mnogo kategorija za rješavanje, od financija, matematike, geometrije i fizike. Navedena aplikacija ima mogućnost rješavanja raznih zadataka iz područja analitičke geometrije prostora. Aplikacija ima pregledno korisničko sučelje i laka je za korištenje. Klikom na jedan od problema otvara se nova kartica, koja sadrži problem rješavanja u obliku kalkulatora i navodi dio teorijske podloge vezan za taj problem.

2.2. MathPortal

Sljedeća aplikacija, *MathPortal* nudi rješavanje matematičkih problema, navodi formule za rješavanje i nudi matematičke testove za provjeru [2]. Slika 2.2. prikazuje grafičko sučelje web aplikacije *MathPortal*.

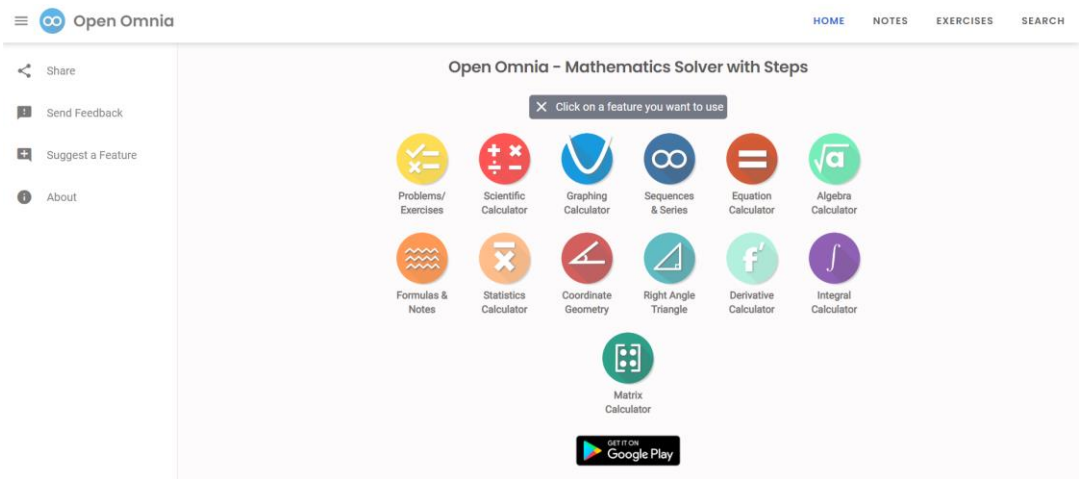


Slika 2.2. Izgled aplikacije *MathPortal*

Aplikacija *MathPortal* je realizirana kao web i android aplikacija. Aplikacija za rješavanje problema analitičke geometrije prostora nudi devet kalkulatora. Klikom na jedan kalkulator otvara se sučelje za upis brojeva kao koeficijenata jednadžbi pravaca i ravnina. Aplikacija nudi grafički prikaz zadanog problema, detaljno opisane korake rješavanja i primjere već ranije unesenih podataka i rješenja.

2.3. Open Omnia

Treće analizirana aplikacija, *Open Omnia* nudi kalkulatore za rješavanje matematičkih zadataka iz područja algebre, statistike, geometrije i rješavanje derivacija [3]. Open Omnia se može pronaći kao web i android aplikacija. Na slici 2.3. prikazano je grafičko sučelje aplikacije i sve kategorije. Klikom na jednu od kategorija otvara se kartica sa problemima koji se mogu riješiti.

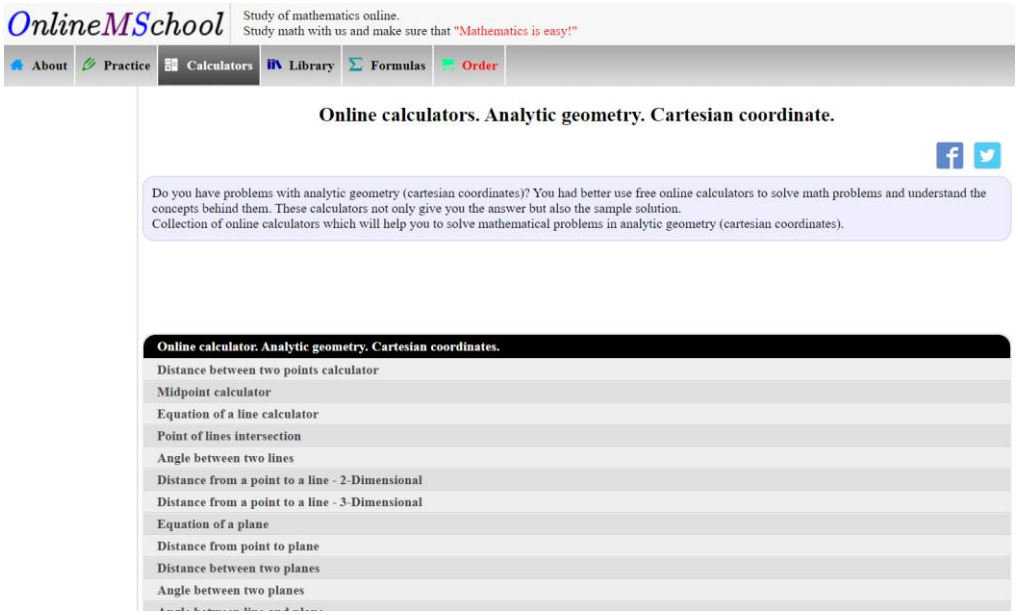


Slika 2.3. Izgled aplikacije *Open Omnia*

Područje analitičke geometrije prostora nudi pet osnovnih problema za rješavanje. Aplikacija je funkcionalna, daje dobre rezultate, ali sadrži jako jednostavne algoritme.

2.4. OnlineMSchool

Zadnje analizirana aplikacija, *OnlineMSschool* ne razlikuje se od drugih aplikacija, ali nudi više kategorija za rješavanje. Kategorije koje druge aplikacije ne sadrže su problemi iz područja kombinatorike, nizova i integriranja [4]. Na slici 2.4. prikazan je izgled aplikacije za područje analitičke geometrije prostora.



Slika 2.4. Izgled web aplikacije *OnlineMSchool*

Aplikacija sadrži nekoliko problema za područje analitičke geometrije prostora. Klikom na jedan od problema otvara se kalkulator za rješavanje sa teorijskom podlogom i slikom koja prikazuje problem koji se rješava. Iako aplikacija sadrži mnogo kategorija, grafičko sučelje je nabacano i jako nepregledno. Aplikacija daje točne rezultate i uz njih korake rješavanja.

3. ANALITIČKA GEOMETRIJA PROSTORA

Analitička geometrija prostora predstavlja granu geometrije u kojoj se geometrijski problemi rješavaju algebarskim metodama. Analitička geometrija prostora temeljena je na ideji da su točke prostora opisane brojevima koji se nazivaju koordinate, a geometrijske krivulje i plohe opisane algebarskim jednadžbama. Analitička geometrija je jako širok pojam, a za potrebe ovog rada obrađeni su ravnina i pravac.

3.1. Ravnina

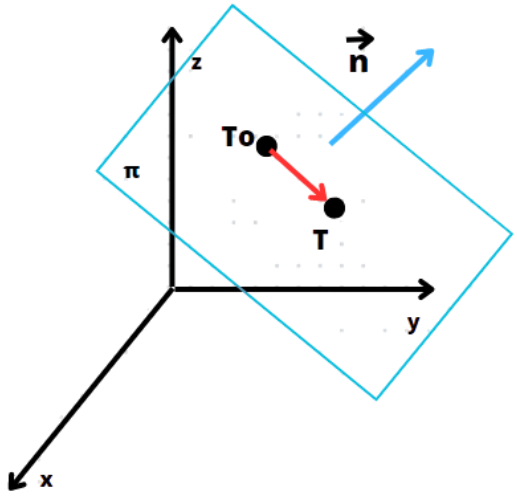
Ravnina je jedan od osnovnih pojmova u geometriji, ona predstavlja ravnu površinu sadržanu točkama kroz koje se može provući beskonačno mnogo različitih pravaca. Svaka linearna jednadžba koja sadrži tri nepoznanice opisuje neku ravninu u prostoru.

Ravnina je jednoznačno određena sa:

1. tri točke koje ne leže na istom pravcu
2. pravcem i točkom izvan tog pravca
3. dva paralelna različita pravca
4. dva pravca koja se sijeku [5].

3.1.1. Opći oblik jednadžbe ravnine

Ravninu je najjednostavnije prikazati pomoću točke koja leži na ravnini i vektora koji je okomit na tu ravninu, taj vektor se naziva normala ravnine. Iz slike 3.1. je vidljivo da točka T_0 leži u ravnini, a $\vec{n}$ je vektor normale te ravnine.



Slika 3.1. Ravnina je određena jednom točkom i vektorom normale

Neka su točka $T_0(x_0, y_0, z_0)$ i vektor normale $\vec{n} = A\vec{\imath} + B\vec{\jmath} + C\vec{k}$, a točka $T(x, y, z)$ predstavlja proizvoljnu točku ravnine. Tada za svaku proizvoljnu točku T se može reći da je vektor $\overrightarrow{T_0T} = (x - x_0)\vec{\imath} + (y - y_0)\vec{\jmath} + (z - z_0)\vec{k}$ okomit na normalu $\vec{n}$, te vrijedi izraz za okomitost $\overrightarrow{T_0T} \perp \vec{n} \Rightarrow \overrightarrow{T_0T} \cdot \vec{n} = 0$. Uvrštavanjem vektora i normale u izraz za okomitost i množenjem dobije se skalarna jednadžba ravnine ili jednadžba ravnine kroz točku [6].

$$A(x - x_0) + B(y - y_0) + C(z - z_0) = 0. \tag{3-1}$$

Raspisivanjem skalarne jednadžbe dobije se izraz

$$Ax + By + Cz - Ax_0 - By_0 - Cz_0 = 0. \tag{3-2}$$

Koeficijenti A, B, C i x_0, y_0, z_0 predstavljaju realne brojeve pa se uvodi oznaka koja predstavlja udaljenost ravnine od ishodišta $D = -Ax_0 - By_0 - Cz_0$. Uvrštavanjem oznake za udaljenost ravnine od ishodišta dobije se opći oblik jednadžbe ravnine

$$Ax + By + Cz + D = 0. \tag{3-3}$$

Iz opće jednadžbe ravnine mogu se lako očitati vektor normale sa komponentama A, B, C i uvrštavanjem dvije proizvoljne koordinate može se odrediti treću koordinatu te odrediti jednu točku koja pripada ravnini.

3.1.2. Normalni oblik jednadžbe ravnine

Neka je $\vec{n} = A\vec{\imath} + B\vec{\jmath} + C\vec{k}$ vektor normale ravnine i točka $T(x, y, z)$ proizvoljna točka koja pripada ravnini. Za vektor normale može se odrediti jedinični vektor u smjeru vektora normale. Prema formuli za određivanje jediničnog vektora [7]

$$\vec{n}_0 = \frac{\vec{n}}{|\vec{n}|}, \tag{3-4}$$

i uvrštavanjem vektora normale $\vec{n}$ te uvrštavanjem formule za duljinu vektora [7]

$$|\vec{n}| = \sqrt{(A^2 + B^2 + C^2}, \tag{3-5}$$

određuje se normirani oblik jednadžbe ravnine. Stoga normirani oblik jednadžbe ravnine izgleda

$$\frac{Ax + By + Cz + D}{\sqrt{(A^2 + B^2 + C^2}} = 0. \tag{3-6}$$

3.1.3. Segmentni oblik jednadžbe ravnine

Neka je $Ax + By + Cz + D = 0$ opći oblik jednadžbe ravnine. Segmentni oblik ravnine predstavlja posebni slučaj općeg oblika jednadžbe ravnine. Kako bi se odredio segmentni oblik jednadžbe ravnine potrebno je znati prolazi li ravnina kroz ishodište. Ukoliko je $D = 0$, ravnina prolazi kroz ishodište i nema smisla odrediti segmentni oblik. Ukoliko je $D \neq 0$, ravnina ne prolazi kroz ishodište i moguće je opći oblik podijeliti sa $-D$, zatim nepoznanice ostaviti u brojniku

$$\frac{Ax}{-D} + \frac{By}{-D} + \frac{Cz}{-D} - 1 = 0$$

$$\frac{x}{\frac{-D}{A}} + \frac{y}{\frac{-D}{B}} + \frac{z}{\frac{-D}{C}} = 1.$$

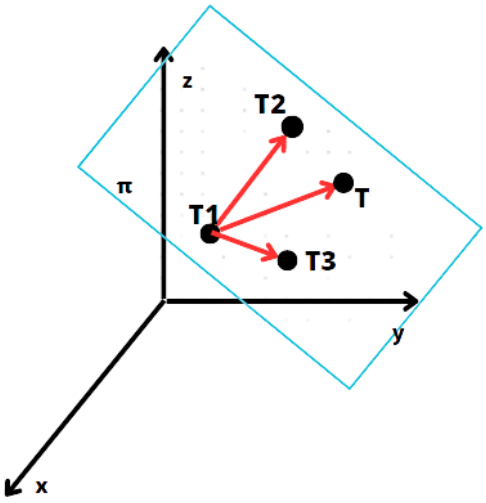
Uvođenjem odgovarajućih oznaka za segmente, odnosno sjecišta ravnine s koordinatnim osima, jednadžba ima oblik

$$\frac{x}{k} + \frac{y}{l} + \frac{z}{m} = 1, \tag{3-7}$$

ovaj oblik jednadžbe naziva se segmentni oblik jednadžbe ravnine.

3.1.4. Jednadžba ravnine kroz tri točke

Neka su izabrane tri točke $T_1(x_1, y_1, z_1), T_2(x_2, y_2, z_2), T_3(x_3, y_3, z_3)$. Kako bi ravnina bila određena sa te tri točke one moraju biti nekolinearne, ne leže na istom pravcu. Neka je $T(x, y, z)$ bilo koja proizvoljna točka koja leži u ravnini. Iz slike 3.2. se može primijetiti kako vektori $\overrightarrow{T_1T}, \overrightarrow{T_1T_2}, \overrightarrow{T_1T_3}$ leže u ravnini.



Slika 3.2. Ravnina kroz tri točke

Ako su dani vektori komplanarni, pripadaju istoj ili paralelnoj ravnini te ako im je mješoviti produkt jednak nuli, dobije se jednadžba

$$\left(\overrightarrow{T_1T} \times \overrightarrow{T_1T_2}\right) \cdot \overrightarrow{T_1T_3} = 0, \tag{3-8}$$

uvrštavanjem vektora i rješavanjem jednadžbe dobije se jednadžba ravnine kroz tri točke

$$\begin{vmatrix} x - x_1 & y - y_1 & z - z_1 \\ x_2 - x_1 & y_2 - y_1 & z_2 - z_1 \\ x_3 - x_1 & y_3 - y_1 & z_3 - z_1 \end{vmatrix} = 0. \tag{3-9}$$

3.1.5. Dvije ravnine u prostoru

Dvije ravnine u prostoru mogu imati različite položaje. Međusobni položaj dviju ravnina ovisi o točkama tih ravnina. Ravnine u prostoru mogu imati sve zajedničke točke, to znači da se ravnine podudaraju. Ravnine su paralelne kada nemaju niti jednu zajedničku točku, a ravnine se sijeku u jednom pravcu, koji se naziva presječnica ravnina.

Neka su π_1 ... $Ax + By + Cz + D = 0$ i π_2 ... $A'x + B'y + C'z + D' = 0$ opći oblici jednadžbi dvije ravnine. Ravnine π_1 i π_2 su paralelne ako su njihovi vektori normale kolinearni, tj. vrijedi

$$\vec{n} = \lambda\vec{n}'$$

$$\frac{A}{A'} = \frac{B}{B'} = \frac{C}{C'}. \tag{3-10}$$

Dvije ravnine π_1 ... $Ax + By + Cz + D = 0$ i π_2 ... $A'x + B'y + C'z + D' = 0$ su okomite ako su im vektori normala okomiti $\vec{n} \perp \vec{n}'$. Tada vrijedi uvjet okomitosti dvije ravnine

$$AA' + BB' + CC' = 0. \tag{3-11}$$

Između dvije ravnine moguće je izračunati kut. Kut između dvije ravnine je kut koji zatvaraju normale tih dviju ravnina. Pošto su normale vektori, kut se dobije iz skalarnog produkta tih vektora

$$\vec{n} \cdot \vec{n}' = |\vec{n}| \cdot |\vec{n}'| \cdot cos \measuredangle(\vec{n}, \vec{n}')$$

$$cos \measuredangle(\vec{n}, \vec{n}') = \frac{\vec{n} \cdot \vec{n}'}{|\vec{n}| \cdot |\vec{n}'|}, \tag{3-12}$$

gdje su vektori normale $\vec{n} = A\vec{\imath} + B\vec{\jmath} + C\vec{k}$ i $\vec{n}' = A'\vec{\imath} + B'\vec{\jmath} + C'\vec{k}$.

Između dvije ravnine moguće je izračunati udaljenost. Udaljenost između dvije ravnine moguće je izračunati samo ako su ravnine paralelne. Udaljenost dviju ravnina predstavlja udaljenost bilo koje točke ravnine do ortogonalne projekcije te točke na drugu ravninu. Udaljenost dviju ravnina $\pi_1 \dots Ax + By + Cz + D = 0$ i $\pi_2 \dots Ax + By + Cz + D' = 0$ računa se formulom

$$d(\pi_1, \pi_2) = \frac{|D - D'|}{\sqrt{(A^2 + B^2 + C^2)}}, \tag{3-13}$$

ukoliko se dvije ravnine sijeku, njihova udaljenost iznosi 0.

3.1.6. Udaljenost točke do ravnine

Udaljenost točke $T_1(x_1, y_1, z_1)$ do ravnine $\pi_1 \dots Ax + By + Cz + D = 0$ računa se formulom

$$d(T_1, \pi_1) = \frac{|Ax_1 + By_1 + Cz_1|}{\sqrt{(A^2 + B^2 + C^2)}}. \tag{3-14}$$

3.2. Pravac

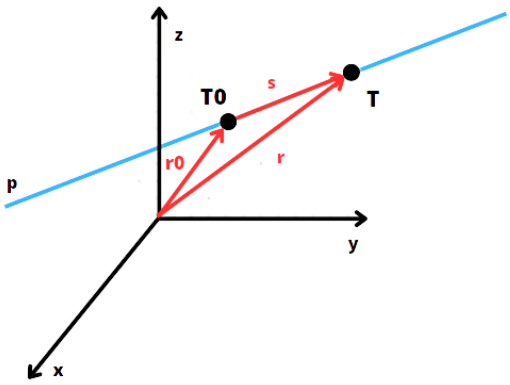
Pravac je osnovni element geometrije koji se ne definira kao element ravnine, već se definira svojstvima pravca koji se daju aksiomima. Pravac je jednoznačno određen s dvije točke i pravac može biti zadan jednom točkom i vektorom koji je paralelan s pravcem, taj vektor se naziva vektor smjera pravca. Vektor smjera pravca nije jedinstven, jedinstven mu je samo smjer [8].

3.2.1. Oblici jednadžbe pravca

Neka je pravac p zadan točkom $T_0(x_0, y_0, z_0)$ i vektorom smjera $\vec{s} = a\vec{\imath} + b\vec{\jmath} + c\vec{k}$. Za pravac p piše se $p = (T_0, \overrightarrow{s})$. Ako se na pravcu nalazi bilo koja druga točka $T_1(x_1, y_1, z_1)$, tada je vektor $\overrightarrow{T_0T_1}$ kolinearan sa vektorom smjera $\vec{s}$ pa vrijedi $\overrightarrow{T_0T_1} = t\vec{s}$. Postojanjem radij vektora točke T_0 i radij vektora točke T_1, može se vektor $\overrightarrow{T_0T_1}$ zapisati kao razlika radij vektora $r_1 - r_0$, stoga je vektorska jednadžba pravca oblika

$$p \dots r_1 = r_0 + t\vec{s}. \tag{3-15}$$

Iz slike 3.3 je vidljivo kako su vektori kolinearni te ih je moguće prikazati u kanonskoj bazi $\vec{\imath}, \vec{\jmath}, \vec{k}$.



Slika 3.3. Pravac određen jednom točkom i vektorom smjera

Uvrštavanjem koordinata sa slike u vektorsku jednadžbu pravca dobije se izraz

$$x\vec{\imath} + y\vec{\jmath} + z\vec{k} = x_0\vec{\imath} + y_0\vec{\imath} + z_0 + t(a\vec{\imath} + b\vec{\jmath} + c\vec{k}). \tag{3-16}$$

Iz izraza se može očitati kako koordinate točke pravca moraju zadovoljavati jednadžbe

$$p \ldots \begin{cases} x = x_0 + at, \\ y = y_0 + bt, \\ z = z_0 + ct, \end{cases} \tag{3-17}$$

a ovaj izraz predstavlja parametarski oblik jednadžbe pravca. Eliminacijom parametra t iz prethodnog izraza dobije se kanonski oblik jednadžbe pravca

$$p \ldots \frac{x - x_0}{a} = \frac{y - y_0}{b} = \frac{z - z_0}{c}. \tag{3-18}$$

Ukoliko na pravcu leže dvije točke $T_1(x_1, y_1, z_1)$ *i* $T_2(x_2, y_2, z_2)$, za vektor smjera se uzima vektor kolinearan sa vektorom između dvije točke $\overrightarrow{T_1T_2} = (x_2 - x_1)\vec{\imath} + (y_2 - y_1)\vec{\jmath} + (z_2 - z_1)\vec{k}$. Uvrštavanjem u kanonsku jednadžbu pravca dobije se jednadžba pravca kroz dvije točke

$$p \ldots \frac{x - x_1}{x_2 - x_1} = \frac{y - y_1}{y_2 - y_1} = \frac{z - z_1}{z_2 - z_1}. \tag{3-19}$$

3.2.2. Dva pravca u prostoru

Dva pravca u prostoru mogu biti paralelni, tada predstavljaju ravninu. Pravci se mogu sjeći u jednoj točki i pravci mogu biti mimoilazni. Za pravce koji su okomiti vrijedi $\vec{s}_1 \cdot \vec{s}_2 = 0$.

Kut između dva pravca je kut koji zatvaraju njihovi vektori smjerova $\measuredangle(p_1, p_2) = \measuredangle(\vec{s}_1, \vec{s}_2)$. Kut između dva vektora rješava se skalarnim produktom. Sređivanjem izraza dobije se formula za kut između dva pravca

$$\vec{s}_1 \cdot \vec{s}_2 = |\vec{s}_1| \cdot |\vec{s}_2| \cdot cos \measuredangle(\vec{s}_1, \vec{s}_2)$$

$$cos \measuredangle(\vec{s}_1, \vec{s}_2) = \frac{\vec{s}_1 \cdot \vec{s}_2}{|\vec{s}_1| \cdot |\vec{s}_2|}. \tag{3-20}$$

3.2.3. Pravac i ravnina u prostoru

Kada pravac i ravnina nemaju zajedničku točku ili kada pravac leži u ravnini onda su pravac i ravnina paralelni. Uvjet paralelnosti pravca i ravnine je $\vec{s} \cdot \vec{n} = 0$, gdje su $\vec{s}$ vektor smjera pravca i $\vec{n}$ vektor normale ravnine. Pravac i ravnina su okomiti ako su vektori smjera i vektor normale paralelni. Ako pravac nije paralelan ravnini, onda pravac probada ravninu u jednoj točki koje se naziva probodište pravca i ravnine.

Ukoliko se pravac i ravnina sijeku, udaljenost između pravca i ravnine iznosi 0. Ukoliko su pravac i ravnina paralelni, udaljenost između pravca i ravnine je bilo koja udaljenost točke s pravca od ravnine.

Kut između pravca i ravnine je kut između pravca i njegove ortogonalne projekcije na ravninu. Kut koji zatvaraju vektor normale ravnine i vektor smjera pravca φ može se izračunati pomoću skalarnog produkta. Budući da je vektor normale ravnine okomit na ravninu, može se zaključiti da je kut između pravca i ravnine $\psi = 90^0 - \varphi$.

Dvije ravnine koje nisu paralelne $\pi_1 \dots Ax + By + Cz + D = 0$ *i* $\pi_2 \dots A'x + B'y + C'z + D' =$

0 jednoznačno određuju pravac p. Tada je pravac p zadan kao presjek dviju ravnina i zapisuje se u obliku

$$p \dots \begin{cases} Ax + By + Cz + D = 0\,, \\ A'x + B'y + C'z + D' = 0. \end{cases} \tag{3-21}$$

4. KORIŠTENE TEHNOLOGIJE

Prije početka izrade aplikacije potrebno je odlučiti koja je najučinkovitija tehnologija za razvoj aplikacije. Za izradu aplikacije korišteno je razvojno okruženje *Visual studio*, a programski kod pisan je u programskom jeziku C#. Za izradu slika početnog zaslona aplikacije korištena je *Canva*, online besplatni uređivač slika.

4.1. Programski jezik C#

C# objektno je orijentirani programski jezik, koji omogućuje razvojnim programerima izradu snažnih aplikacija. C# je 2000. godine razvio *Microsoft* kao dio *.Net* platforme koja omogućuje pokretanje aplikacija u različitim operacijskim sustavima. C# je vrlo sličan programskim jezicima C, C++ i Java. C# sadrži snažne značajke koje nude lakšu izradu aplikacija i lakše korištenje memorije. Jedna od najvažnijih prednosti C# je rukovanje memorijom, programski jezik sam za programera raspodjeljuje memoriju i oslobađa od nekorištenih objekata. C# omogućava nasljeđivanje, polimorfizam i enkapsulaciju, temelje objektno orijentiranog programiranja. U programskom jeziku C# podržani su generički tipovi i metode, koji pružaju bolje performanse [9]. Slika 4.1 prikazuje kako izgleda glavna metoda (eng. Main method) u programskom jeziku C#, koja predstavlja ulaznu točku za aplikaciju.

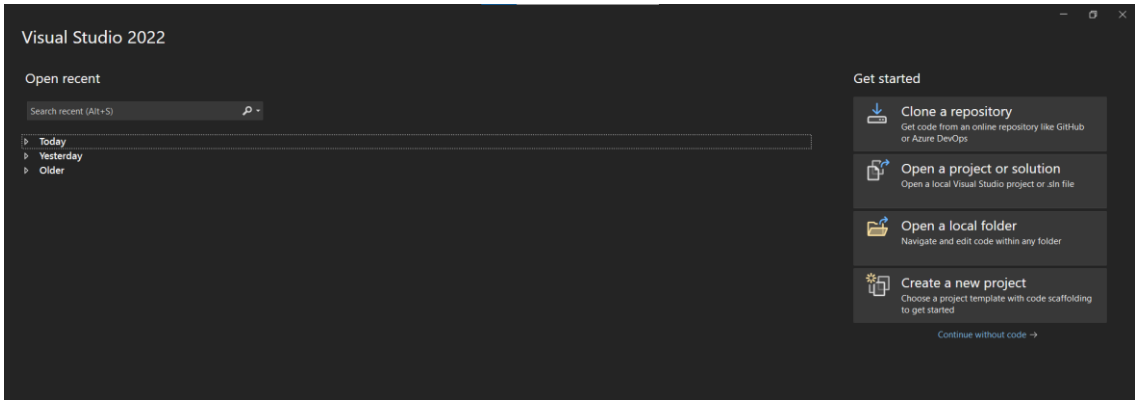
```
Program                                    Main(string[] args)
0 references
class Program
{
    0 references
    static void Main(string[] args)
    {

        Console.WriteLine(args.Length);
    }
}
```

Slika 4.1. Glavna metoda u programskom jeziku C#

Kada se aplikacija pokreće prvo se poziva glavna metoda. Iz slike je također vidljiva sintaksa programskog jezika C#, u kojemu je vrlo bitna razlika između velikog i malog slova. Prikazano je korištenje klasa i objekata, objektima se pristupa preko operatora točke. Programski kod korišten u završnom radu pisan je u obliku klasa programskog jezika C#. Korištene su već postojeće biblioteke koje su dostupne u C#, jedna od najkorištenijih je biblioteka za pozivanje matematičkih funkcija *Math*.

4.2. Visual studio

Visual studio je integrirano razvojno sučelje kojeg je razvio *Microsoft*. *Visual studio* se koristi za izradu računalnih programa, aplikacija i internetskih stranica. *Visual studio* koristi *Microsoftove* platforme za razvoj programske podrške kao što su *Windows API* i *Windows Forms*. *Visual studio* nudi uređivač programskog koda za pisanje i refaktoriranje koda. *Visual studio* podržava pisanje programa u 36 različitih programskih jezika. *Visual studio* je dostupan u 3 edicije. *Community* je besplatna edicija koja je namijenjena za edukaciju i akademska istraživanja. *Professional* je komercijalna edicija koja je izašla 2010. godine te uz sve prednosti omogućuje i izradu mobilnih aplikacija. *Enterprise* je posljednja edicija, ona uz sve prednosti koje omogućava *Professional* edicija nudi novi set alata za razvoj softvera, alata za testiranje i izradu baza podataka [10]. Slika 4.2. predstavlja početnu stranicu razvojnog okruženja *Visual studio*.



Slika 4.2. Razvojno okruženje *Visual studio*

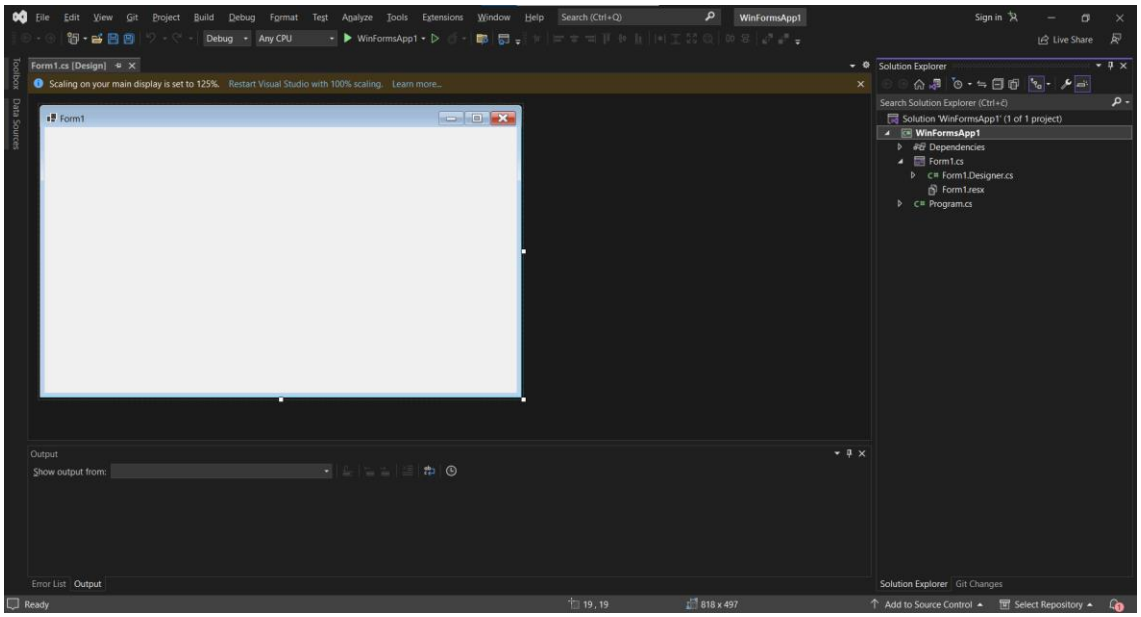
Početna stranica omogućava kreiranje novog projekta i otvaranje projekta iz postojeće datoteke. Prikazani su i nedavno korišteni projekti koje je moguće otvoriti. Kreiranjem novog projekta otvara se prozor koji omogućava biranje predložaka (engl. *Template*) za kreiranje aplikacija. Odabirom jednog od predložaka otvara se prozor za upis naziva projekta, lokacije gdje će projekt biti spremljen i naziv rješenja (engl. *Solution*), rješenje je spremnik za jedan ili više projekata u *Visual studiu*. Dalje se otvara prozor za odabir *.Net* programskog okvira (engl. *.Net Framework*). Kreiranjem projekta otvara se korisničko sučelje projekta, generira se klasa *Program* koja sadrži glavnu metodu. Korisničko sučelje omogućava dodavanje klasa i ostalih datoteka koje *Visual studio* omogućuje. Za potrebe završnog rada koristi se predložak *Windows Forms App*.

4.3. Windows Forms

Windows Forms je okvir korisničkog sučelja (engl. *User interface framework*) koji se koristi za izgradnju korisničkih aplikacija za Windows. Windows forme omogućuju brojne značajke za rad sa aplikacijama. Neke od značajki su jednostavno kreiranje korisničkog sučelja aplikacije, koji je omogućen povuci-i-pusti (engl. *Drag-and-drop*) tehnikom te korisnički unos podataka. Prednosti kreiranja *Windows Forms* aplikacije su lako ažuriranje i mogućnost rada sa i bez konekcije na internet.

Forma (engl. *Form*) je vizualna površina koja se koristi za prikazivanje informacija korisniku. *Windows Forms* sadrže mnogo kontrola (engl. *controls*) koje se mogu dodati formi, neke od najvažnijih su tekstualni okviri (engl. *text boxes*) i gumbi (engl. *buttons*). *Windows Forms* omogućuju i izgradnju korisničkih kontrola korištenjem odgovarajuće klase.

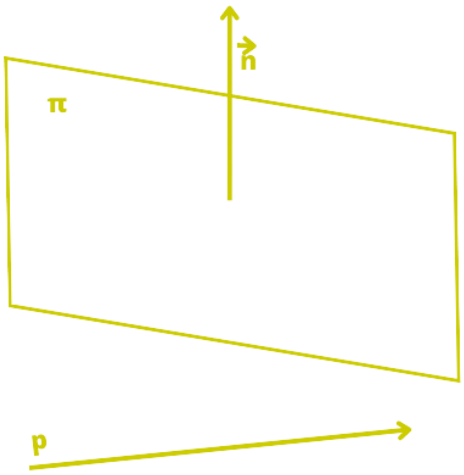
Windows Forms forma se dodaje u projekt kao datoteka jedinstvenog imena. Kada je forma dodana otvara se dizajner kojim se uređuje izgled forme. Dodana forma sadrži datoteku sa resursima, C# klasu u koju se upisuje kod za formu i datoteku koja predstavlja dizajner. Slika 4.3 predstavlja izgled prazne forme i osnovnog projekta u *Windows Forms* [11].



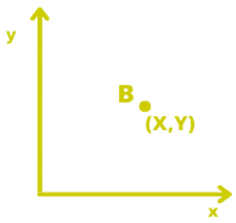
Slika 4.3. *Windows Forms* forma

4.4. Canva

Canva je online platforma za grafički dizajn namijenjena kreiranju prezentacija i grafike. Aplikacija nudi predloške za korištenje. Neki od predložaka koje aplikacija nudi su dokument oblika A4 i objava na društvenoj mreži. *Canva* je besplatna za korištenje, no neki visoko kvalitetni elementi su obvezni za plaćanje ako se žele koristiti. Korisnici mogu platiti izradu fizičkih proizvoda kao što su pozivnice, posjetnice i jelovnici [12]. Za početni izgled aplikacije napravljene su slike ravnine i pravca te koordinatnog sustava sa točkom i njenim koordinatama. Slike 4.4. i 4.5. prikazuju slike napravljene u *Canvi* za početni zaslon aplikacije.



Slika 4.4. Pravac i ravnina



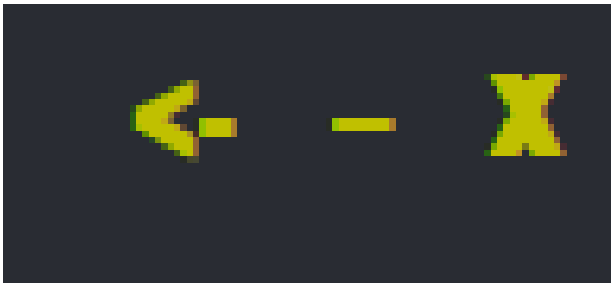
Slika 4.5. Koordinatni sustav sa točkom

5. WINDOWS FORMS APLIKACIJA

Praktični dio rada predstavlja *Windows Forms* aplikacija koja se koristi za rješavanje problema analitičke geometrije prostora. Aplikacija omogućava korisniku odabir algoritama koji korisnik želi riješiti, unos parametara jednadžbe, pravca ili točke.

5.1. Struktura aplikacije

Aplikacija se sastoji od glavne forme *Form1.cs* , forme za odabir algoritama OsnovneOperacije.cs te dvije forme koje sadrže algoritme koji se računaju *AlgoritmiFirst.cs* i *AlgoritmiSecond.cs*. Forme sadrže prilagođenu strelicu za povratak na prethodnu formu, prilagođeni gumb za smanjivanje forme i prilagođeni gumb za izlazak iz aplikacije. Prilagođeni gumbi su napravljeni tako da se u svojstvo gumba *Text* napiše izgled i promijeni boja. Kako bi gumbi mogli biti vidljivi te kako bi se maknuli obrub i gumbi za smanjivanje, uvećavanje i izlazak iz forme, potrebno je u svojstvima svake forme postaviti svojstvo *FormBordedStyle* na *None*. Forme nemaju mogućnost uvećavanja. Slika 5.1. predstavlja prilagođene gumbe.



Slika 5.1. Prilagođeni gumbi

Kako bi prilagođeni gumbi bili funkcionalni potrebno je za svaki dodati programski kod.

```
private void buttonMin_Click(object sender, EventArgs e)
        {
            WindowState = FormWindowState.Minimized;
        }
```

Izvorni kod 5.1. Smanjivanje forme

Izvorni programski kod 5.1. predstavlja metodu za gumb smanjivanja forme. Klikom na gumb svojstvo forme *WindowState* postavlja se na *Minimized* te se aplikacija smanjuje na alatnu traku.

Programski kod 5.2. predstavlja metodu za prilagođeni gumb za zatvaranje aplikacije.

```
private void buttonClose_Click(object sender, EventArgs e)
        {
            Application.Exit();
        }
```

Izvorni kod 5.2. Zatvaranje aplikacije

Klikom na gumb za izlazak iz aplikacije poziva se metoda koja objektu klase *Application* poziva metodu *Exit()* čime se aplikacija zatvara.

Gumb za povratak sadrže sve forme osim glavne forme jer je ona početna.

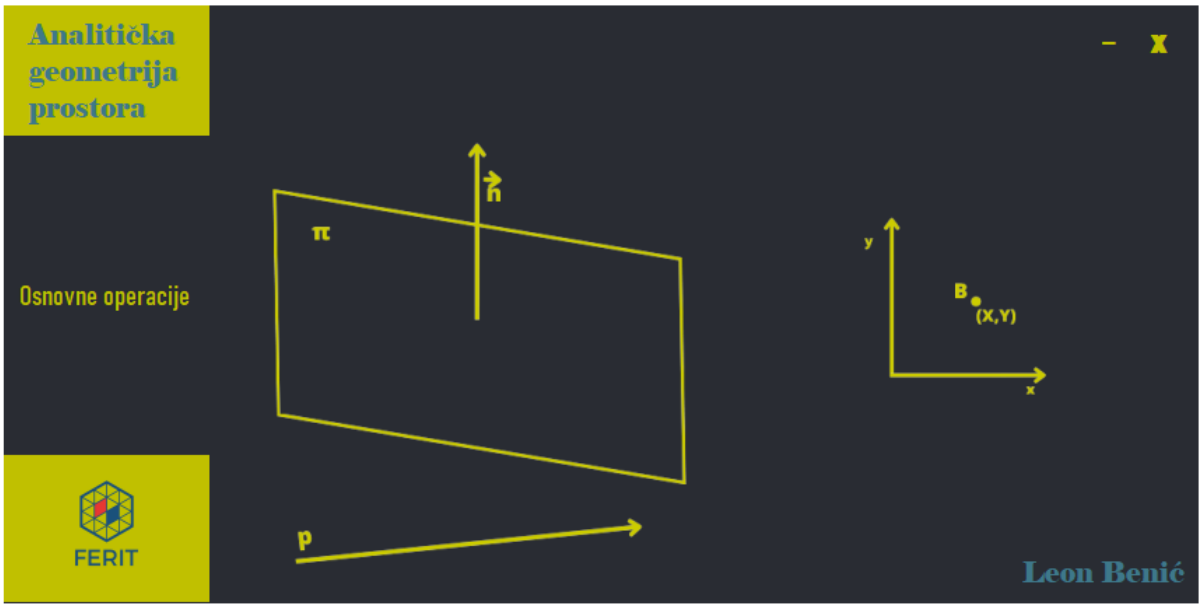
```
private void button1_Click(object sender, EventArgs e)
        {
            Form1 form = new Form1();
            form.Show();
            Visible = false;
        }
```

Izvorni kod 5.3. Povratak iz forme OsnoveOperacije.cs

Programski kod 5.3. prikazuje metodu gumba za povratak iz forme OsnovneOperacije.cs u glavnu formu. Instanca klase *Form1* predstavlja formu koja se klikom prikazuje, a forma se prikazuje pozivanjem metode *Show()* nad objektom. Kako bi bila prikazana samo jedna forma, potrebno je postaviti svojstvo forme *Visible* na *false*. Forme *AlgoritmiFirst* i *AlgoritmiSecond* klikom na strelicu vraćaju korisnika na formu za odabir algoritama OsnovneOperacije.cs.

5.1.1. Form1.cs

Forma *Form1.cs* predstavlja početni izgled aplikacije te se prva otvara pokretanjem aplikacije. Slika 5.2. predstavlja početni izgled aplikacije. Sa slike je vidljivo da forma sadrži prilagođene gumbe za smanjivanje i izlazak iz aplikacije, slike 4.4. 4.5. te gumb sa tekstom Osnovne operacije. Klikom na gumb Osnovne operacije otvara se forma za odabir algoritama.



Slika 5.2. Početni izgled aplikacije

5.1.2. OsnovneOperacije.cs

Forma OsnovneOperacije.cs predstavlja izbor mogućih problema za izračun. Na slici 5.3. se mogu vidjeti svi algoritmi koji su dani za izračun.

Slika 5.3. Osnovne operacije i algoritmi

Klikom na jedan od devet algoritama otvara se nova forma gdje se može izračunati problem koji je opisan tekstom gumba, problemi koji se računaju su obrađeni u teorijskom dijelu analitičke geometrije prostora.

5.1.3. AlgoritmiFirst.cs

Forma *AlgoritmiFirtst.cs* predstavlja kalkulator za izračun pet problema. Problemi koji se mogu riješiti u formi su:

- Kut između dvije ravnine
- Kut između ravnine i pravca
- Pripada li točka ravnini?
- Prolazi li ravnina kroz ishodište?
- Jednadžba pravca kroz dvije točke

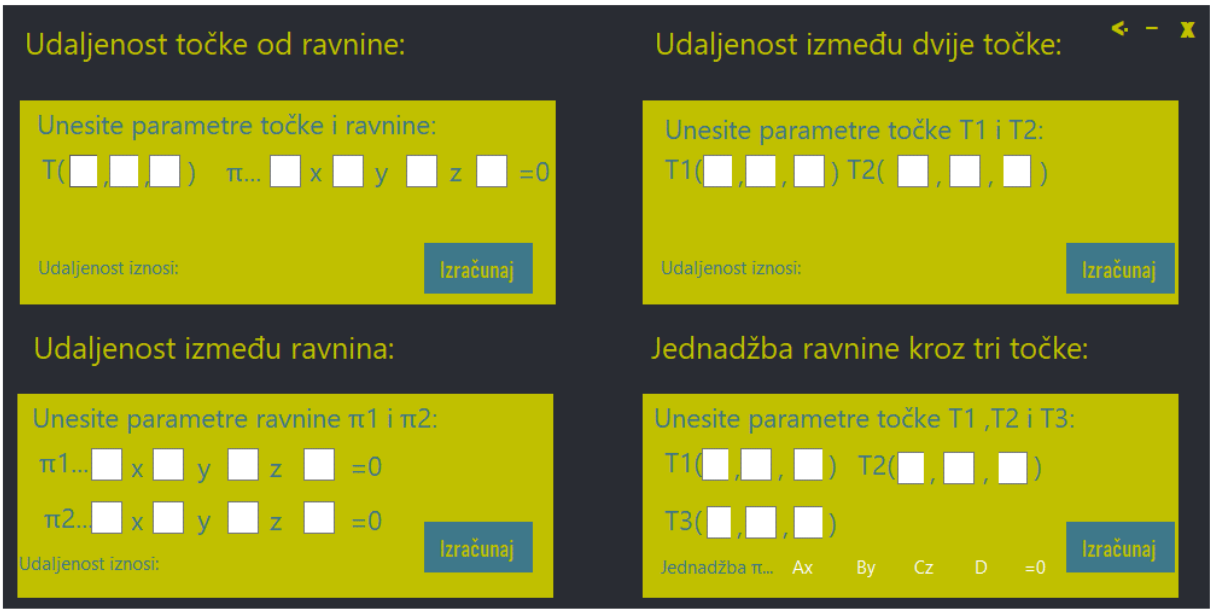
Forma se sastoji od četiri jednaka *Panela* koji predstavljaju zasebne probleme za izračun. Na slici 5.4. se vidi izgled forme te algoritmi koji su nabrojani.

Slika 5.4. Forma AlgoritmiFirst.cs

5.1.4. AlgoritmiSecond.cs

Forma *AlgoritmiSecond.cs* napravljena je isto kao i forma *AlgoritmiFirst.cs* samo sadrži ostale probleme:

- Udaljenost točke od ravnine
- Udaljenost između dvije točke
- Udaljenost između ravnina
- Jednadžba ravnine kroz tri točke



Slika 5.5. Forma AlgoritmiSecond.cs

5.2. Kut između dvije ravnine

5.2.1. Programski dio

Prvi problem koji se nudi za računanje je izračunavanje kuta između dvije ravnine. Izvorni programski kod 5.4 predstavlja računanje kuta između dvije ravnine.

```
private void button8_Click(object sender, EventArgs e)
        {
            double n1i;
            double n1j;
            double n1k;


            double n2i;
            double n2j;
            double n2k;

            if (!double.TryParse(aCordinatePlane1.Text, out n1i) ||
!double.TryParse(bCordinatePlane1.Text, out n1j) ||
!double.TryParse(cCordinatePlane1.Text, out n1k) ||
!double.TryParse(aCordinatePlane2.Text, out n2i) ||
!double.TryParse(bCordinatePlane2.Text, out n2j) ||
!double.TryParse(cCordinatePlane2.Text, out n2k))
            {
                MessageBox.Show("Unos mora biti broj i ne smije biti
prazan", "Upozorenje!", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }

            //formula za kut
            double cosine = ((n1i * n2i)+ (n1j * n2j) + (n1k * n2k)) /(
Math.Sqrt(Math.Pow(n1i, 2) + Math.Pow(n1j, 2) + Math.Pow(n1k, 2)) *
Math.Sqrt(Math.Pow(n2i, 2) + Math.Pow(n2j, 2) + Math.Pow(n2k, 2)));
            cosine= Math.Acos(cosine);
            //pretvorba u stupnjeve
            cosine = cosine * 180 / Math.PI;
            cosine = Math.Round(cosine, 4);
            cosineValue.Text=cosine.ToString()+ "°";



        }
```

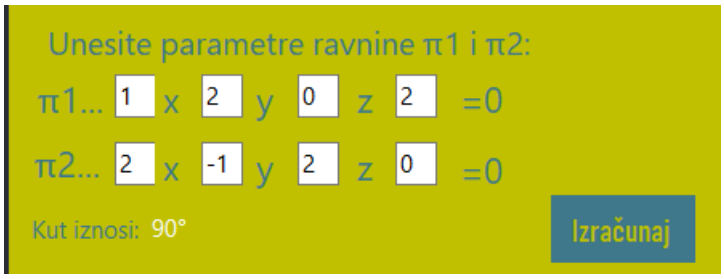
Izvorni kod 5.4. Kut između dvije ravnine

Objekti *double* tipa podatka n1 i n2 predstavljaju norme dvije ravnine, a njihove komponente su dobivene preko *Text* svojstva *TextBoxa* u koji se brojevi unose. Prvo se sprječava pogrešan unos podataka, prazan *TextBox* ili da unos nije broj. Ovo se može provjeriti funkcijom *TryParse()*. Funkcija se predaje tipu podataka koji se traži, u ovom slučaju *double* te se provjerava ako nije *double* podatak u *Text* svojstvu, ulati se u grananje i ispisuje se poruka preko *Show()* funkcije. Ako je unos valjan onda parametar *out* za svaku komponentu normale postavlja uneseni broj i računa

se kut. Kut se računa prema formuli (3-12) , gdje se za izračun korijena i kvadriranja koriste funkcije klase Math, *Math.Sqrt()* i *Math.Pow()*. Brojnik predstavlja skalarni produkt dvije normale pa se on računa po formuli za skalarni produkt. U nazivnik se raspisuje norma svake normale prema formuli (3-5). Kosinus kuta se računa funkcijom *Acos()*, gdje se kao parametar predaje *double* tip podataka koji predstavlja račun kuta. Kako bi se dobio ispis kuta u stupnjevima potrebno je pretvoriti kut koji je u radijanima i stupnjeve prema formuli za pretvorbu. ToString() metoda se koristi jer je *Text* svojstvo *string* podatak, a kut je *double* tip podatka. Klikom na gumb izračunaj poziva se dana metoda i ispisuje se kut između dvije ravnine.

5.2.2. Rješenje problema

U ovom poglavlju prikazana je funkcionalnost aplikacije te primjer rješavanja problema u dva primjera. Zadaci su uzeti iz [13]. Prvi primjer za problem kut između dvije ravnine je izračunati kut koji zatvaraju ravnine $\pi_1 \dots x + 2y + 2 = 0$ i $\pi_2 \dots 2x - y + 2z = 0$. Na slici 5.6 je prikazano kako se unose podaci te rješenje problema nakon što je kliknut gumb izračunaj.



Slika 5.6. Izračun problema kut između dvije ravnine

Aplikacija je izračunala kut od 90^o, računskom provjerom i provjerom iz skripte [13], vidljivo je kako aplikacija daje točno rješenje. Na slici 5.7. je prikazan pogrešan unos, prazna komponenta i *string* podatak. Klikom na gumb U redu omogućava se ponovni unos podataka.

Slika 5.7. Pogrešan unos

5.3. Kut između ravnine i pravca

5.3.1. Programski dio

Kako je objašnjeno u teorijskom dijelu, kut između pravca i ravnine je oblika $\psi = 90^0 - \varphi$. Izvorni programski kod 5.5. opisuje rješenje problema kuta između ravnine i pravca.

```
private void button2_Click(object sender, EventArgs e)
        {
            double A;
            double B;
            double C;
            double D;


            double x;
            double y;
            double z;


            if (!double.TryParse(planeA.Text, out A) ||
!double.TryParse(planeB.Text, out B) || !double.TryParse(planeC.Text, out
C) || !double.TryParse(planeD.Text, out D) || !double.TryParse(lineX.Text,
out x) || !double.TryParse(lineY.Text, out y) ||
!double.TryParse(lineZ.Text, out z))
            {
                MessageBox.Show("Unos mora biti broj i ne smije biti
prazan", "Upozorenje!", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }

            //formula za kut

            double cosine = ((A * x) + (B * y) + (C * z)) /
(Math.Sqrt(Math.Pow(A, 2) + Math.Pow(B, 2) + Math.Pow(C, 2)) *
Math.Sqrt(Math.Pow(x, 2) + Math.Pow(y, 2) + Math.Pow(z, 2)));
            cosine = Math.Acos(cosine);
            //pretvorba u stupnjeve

            cosine = cosine * 180 / Math.PI;
            //kut između pravca i ravnine je 90-kut između normale i pravca
smjera

            cosine = Math.Abs(90 - cosine);
            cosine = Math.Round(cosine, 4);
            cosineValue2.Text = cosine.ToString() + "°";

        }
```

Izvorni kod 5.5. Kut između ravnine i pravca

Programski kod je sličan kodu za kut između dvije ravnine, sadrži sprječavanje pogrešnog unosa, računanje kuta prema formuli , ali računa se kut između normale ravnine i vektora smjera. Zatim pretvorba u stupnjeve te onda računanje kuta. Funkcija *Abs*() klase *Math* koristi se kako kut između

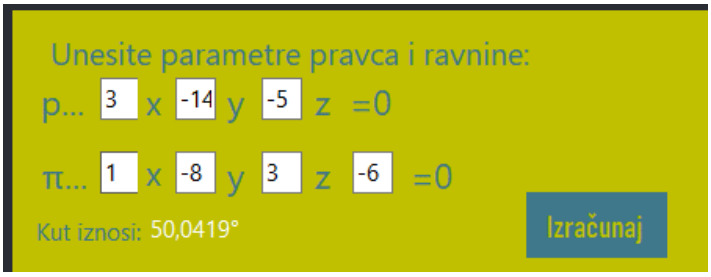
ravnine i pravca ne bi bio negativan. Funkcijom *Round(cosine,4)* zaokružuje se dani kut na četiri decimalna mjesta zbog preglednijeg ispisa.

5.3.2. Rješenje problema

Problem koji aplikacija rješava je odrediti kut između pravca i ravnine :

$$p \ldots \frac{x-1}{3} = \frac{y}{-14} = \frac{z-1}{-5} \quad , \ \pi_1 \ldots x - 8y + 3z - 6 = 0.$$

Pravac se u aplikaciju unosi kao komponente vektora smjera pravca očitane iz kanonskog oblika, kako je prikazano na slici 5.8.



Slika 5.8. Izračun problema kut između ravnine i pravca

Zadatak je uzet iz [14], provjerom je dokazano da aplikacija daje točno rješenje.

5.4. Prolazi li ravnina kroz ishodište? Pripada li točka ravnini?

5.4.1. Programski dio

Problemi Prolazi li ravnina kroz ishodište i pripada li točka ravnini realizirani zajedno.

```
private void button4_Click(object sender, EventArgs e)
        {
            double A;
            double B;
            double C;
            double D;


            double x;
            double y;
            double z;


            if (!double.TryParse(planeValueA.Text, out A) ||
!double.TryParse(planeValueB.Text, out B) ||
!double.TryParse(planeValueC.Text, out C) ||
!double.TryParse(planeValueD.Text, out D) || !double.TryParse(pointX.Text,
out x) || !double.TryParse(pointY.Text, out y) ||
!double.TryParse(pointZ.Text, out z))
            {
                MessageBox.Show("Unos mora biti broj i ne smije biti
prazan", "Upozorenje!", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }

            double sum = A * x + B * y + C * z + D;

            if(sum == 0)

            {
                valuePripada.Text = "Točka pripada danoj ravnini";
            }
            else
            {
                valuePripada.Text = "Točka ne pripada danoj ravnini";
            }



            if (D == 0)
            {
                valueKoordinatneOsi.Text = "Ravnina prolazi kroz
ishodište";
            }
            else
            {
                valueKoordinatneOsi.Text = "Ravnina ne prolazi kroz
ishodište";
            }

        }
```

Izvorni kod 5.6. Prolazi li ravnina kroz ishodište i pripada li točka ravnini

Izvorni programski kod 5.6. sadrži provjeru pogrešnog unosa za koordinate točke i parametre ravnine i algoritme za provjeru. Kada se koordinate točke uvrste u jednadžbu ravnine zbroj mora biti jednak nuli, tada točka pripada ravnini. Ovo je realizirano preko objekta *double* tipa podatka *sum*. *If else* grananjem se provjerava je li zbroj jednak nuli i ispisuje se poruka.

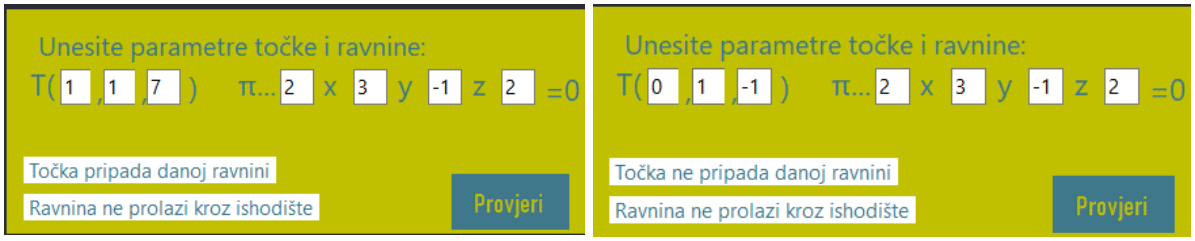
Kako bi se odredilo prolazi li ravnina kroz ishodište ili ne potrebno je provjeriti je li parametar D jednak nuli ili nije, ovisno o rezultatu mijenja se *Text* svojstvo.

5.4.2. Rješenje problema

Za rješavanje ova dva problema potrebno je unijeti koordinate točke i jednadžbu ravnine. Primjeri problema koje aplikacija rješava su:

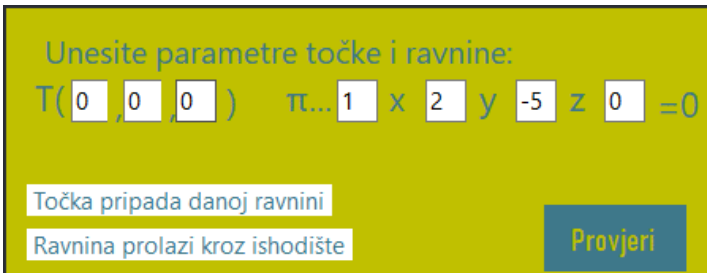
1. Pripada li točka $T(1,1,7)$ ravnini $\pi_1 \dots 2x + 3y - z + 2 = 0$,
2. Pripada li točka $T(0,1,-1)$ ravnini $\pi_1 \dots 2x + 3y - z + 2 = 0$,
3. Prolazi li ravnina $\pi_1 \dots 2x + 3y - z + 2 = 0$ kroz ishodište,
4. Prolazi li ravnina $\pi_1 \dots x + 2y - 5z = 0$ kroz ishodište.

Zadaci su uzeti iz [15]. Na slici 5.9. je vidljivo kako točka $T(1,1,7)$ ne pripada danoj ravnini, a točka $T(0,1,-1)$ pripada te kako ravnina $\pi_1 \dots 2x + 3y - z + 2 = 0$ ne prolazi kroz ishodište.



Slika 5.9. Izračun 1.,2. i 3. problema prolazi li ravnina kroz ishodište i pripada li točka ravnini

Na slici 5.10. vidljivo je da ravnina $\pi_1 \dots x + 2y - 5z = 0$ prolazi kroz ishodište, gdje točka $T(0,0,0)$ predstavlja ishodište.



Slika 5.10. Izračun 4. problema prolazi li ravnina kroz ishodište i pripada li točka ravnini

Provjerom rezultata utvrđeno je da aplikacija daje točne rezultate na zadane probleme.

5.5. Jednadžba pravca kroz dvije točke

5.5.1. Programski dio

Izvorni programski kod 5.7. za računanje pravca kroz dvije točke podijeljen je u dva dijela. Prvi dio je računanje vektora kolinearnog sa vektorom smjera pravca i uvrštavanje jedne točke, a drugi dio je programski kod kako zapisati u parametarski oblik jednadžbe pravca i uljepšati ispis.

```
private void button6_Click(object sender, EventArgs e)
        {
            double x1;
            double y1;
            double z1;

            double x2;
            double y2;
            double z2;

            if (!double.TryParse(point1x.Text, out x1) ||
!double.TryParse(point1y.Text, out y1) || !double.TryParse(point1z.Text,
out z1) || !double.TryParse(point2x.Text, out x2) ||
!double.TryParse(point2y.Text, out y2) || !double.TryParse(point2z.Text,
out z2))
            {
                MessageBox.Show("Unos mora biti broj i ne smije biti
prazan", "Upozorenje!", MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }


            double x12 = x2 - x1;
            double y12 = y2 - y1;
            double z12 = z2 - z1;

            double s1 = x12;
            double s2 = y12;
            double s3 = z12;
            string s1text;
            string s2text;
            string s3text;
```

Izvorni kod 5.7. Pravac kroz dvije točke-računanje

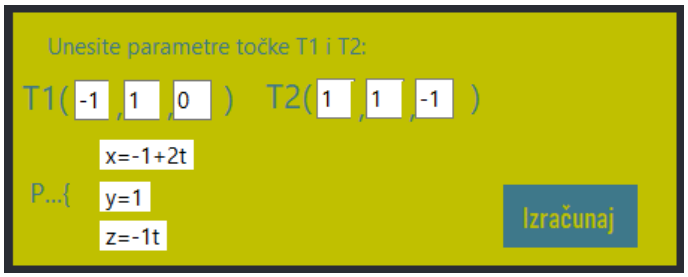
Prvi dio programskog koda sadrži sprječavanje pogrešnog unosa, računanje vektora kolinearnog vektora po formuli $\overrightarrow{T_1T_2} = (x_2 - x_1)\vec{\imath} + (y_2 - y_1)\vec{\jmath} + (z_2 - z_1)\vec{k}$. Drugi dio prikazan je kodom 5.8. . *If else* grananjem provjerava se jesu li parametri vektora smjera veći ili jednak nuli. Ako je parametar veći dodaje se *string* +, jer u ispisu pozitivan broj ne sadrži predznak, a ako je parametar jednak nuli tekst postaje prazan *string* jer je uobičajeno da se nula ne piše. *String* t dodaje se na kraju teksta jer parametarski oblik sadrži t kao parametar. Parametarski oblik sadrži i jednu točku koja pripada pravcu, uobičajeno prvu točku koja je navedena. Zbog toga se *if else* grananjem provjerava jesu li koordinate točke jednake nuli, ako jesu postaju prazan *string* i nisu prikazane. Na kraju se *Text* svojstvu dodaje tekst koordinate točke i vektora smjera.

```
if (s1 > 0)
{
     s1text = "+" + s1.ToString()+"t";
}
else if(s1==0)
{
    s1text = "";
}else
{
    s1text =s1.ToString() + "t";
}
if (s2 > 0)
{
    s2text = "+" + s2.ToString() + "t";
}
else if (s2 == 0)
{
    s2text = "";
}else
{
    s2text = s2.ToString() + "t";
}
if (s3 > 0)
{
    s3text = "+" + s3.ToString() + "t";
}
else if (s3 == 0)
{
    s3text = "";
}else
{
    s3text = s3.ToString() + "t";
}
string x,y,z;
if (x1 == 0)
{
    x = "";
}else
{
    x= x1.ToString();
}
if (y1 == 0)
{
    y = "";
}else
{
    y = y1.ToString();
}
if (z1 == 0)
{
    z = "";
}else
{
    z = z1.ToString();
}
xvalue.Text = "x=" + x + s1text;
yvalue.Text = "y=" + y + s2text;
zvalue.Text = "z=" + z + s3text;}
```

Izvorni kod 5.8. Pravac kroz dvije točke-uljepšavanje ispisa

5.5.2. Rješenje problema

Primjer problema koji aplikacija rješava je odrediti jednadžbu pravca između točaka $T(-1,1,0)$ i $T(1,1,-1)$. Na slici 5.11. je vidljivo rješenje problema te ručnim rješavanjem je utvrđeno da aplikacija daje točnu parametarsku jednadžbu pravca.



Slika 5.11. Izračun problema pravac kroz dvije točke

5.6. Udaljenost točke od ravnine

5.6.1. Programski dio

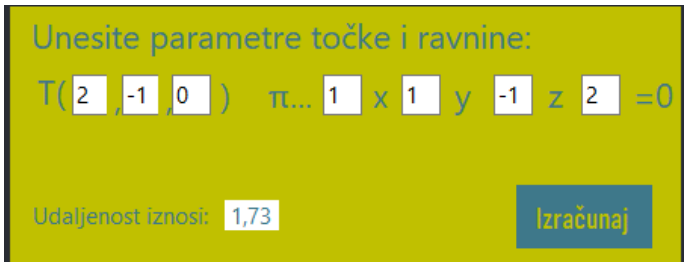
Izvorni programski kod 5.9. za udaljenost točke od ravnine realiziran je pomoću funkcija *Math* biblioteke prema formuli (3-14).

```
            double distance
=Math.Abs(A*x+B*y+C*z+D)/(Math.Sqrt(Math.Pow(A,2)+Math.Pow(B,2)+Math.Pow(C,
2)));
            distance = Math.Round(distance, 2);

            distancePointPlane.Text=distance.ToString();
```

Izvorni kod 5.9. Udaljenost točke od ravnine

5.6.2. Rješenje problema

Primjer problema koji aplikacija rješava je izračunati udaljenost točke $T(2,-1,0)$ od ravnine $\pi_1 \ldots x + y - z + 2 = 0$. Slika 5.12. prikazuje rješenje udaljenosti koje je provjereno i točno je rješenje.



Slika 5.12. Izračun problema udaljenost točke od ravnine

5.7. Udaljenost između dvije točke

5.7.1. Programski dio

Udaljenost između dvije točke $A(x_1, y, z_1)$ i $B(x_2, y_2, z_2)$ se računa prema formuli:

$$|AB| = \sqrt{((x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2)} \tag{5-1}$$

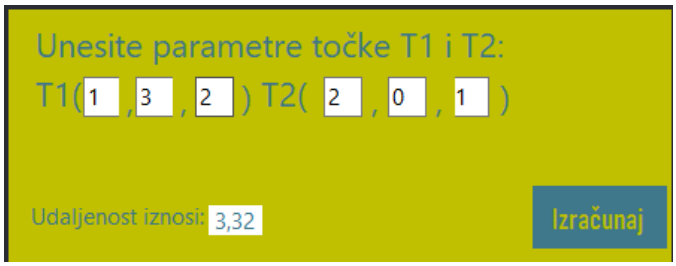
Izvorni programski kod 5.10. predstavlja realizaciju formule (5-1).

```
double distanceOfPoints =Math.Sqrt(Math.Pow(x2-x1,2)+Math.Pow(y2-
y1,2)+Math.Pow(z2-z1,2));
            distanceOfPoints = Math.Round(distanceOfPoints, 2);
            distancePoints.Text = distanceOfPoints.ToString();
```

Izvorni kod 5.10. Udaljenost između dvije točke

5.7.2. Rješenje problema

Na slici 5.13 prikazano je rješenje problema udaljenosti između točaka $T(1,3,2)$ i $T(2,0,1)$.



Slika 5.13. Izračun problema udaljenost između dvije točke

5.8. Udaljenost između ravnina

5.8.1. Programski dio

Udaljenost između ravnina se računa po formuli (3-13), programski kod 5.11. predstavlja formulu.

```
if (paralel1 && paralel2 && paralel3)
            {
                double distanceOfPlanes = Math.Abs(D1 - D2) /
(Math.Sqrt(Math.Pow(A1, 2) + Math.Pow(B1, 2) + Math.Pow(C1, 2)));
                distanceOfPlanes = Math.Round(distanceOfPlanes, 2);
                distancePlanes.Text = distanceOfPlanes.ToString(); }
            else{
                distancePlanes.Text = "0";}
```

Izvorni kod 5.11. Udaljenost između ravnina-formula

Udaljenost između dvije ravnine može se izračunati samo ako su one paralelne, zbog toga preko *if else* grananja provjerava se uvjet paralelnosti dvije ravnine prema formuli (3-10). Izvorni programski kod 5.12. provjerava parametre ravnine, ako su nula ili ako su djeljivi onda je *bool* tip podataka postavljen na *true* i može se računati udaljenost, ako nisu postavljen je na *false.*

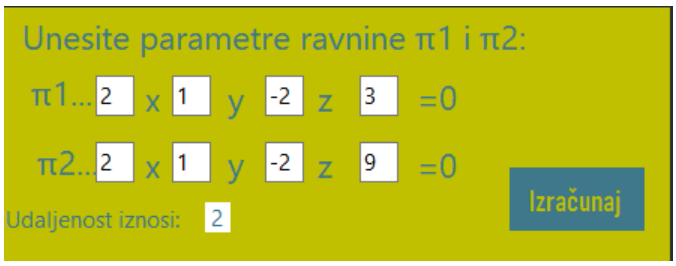
```
bool paralel1, paralel2, paralel3;


             if (A1==0 && A2==0)
            {
                paralel1 = true;

            }else if (  A1 / A2 == 1)
            {
                paralel1 = true;

            }else
            {
                paralel1= false;

            }

            if (B1 == 0 && B2 == 0)
            {
                paralel2 = true;

            }
            else if (B1 / B2 == 1)
            {
                paralel2 = true;

            }else
            {
                paralel2 = false;

            }
```

```
if (C1 == 0 && C2 == 0)
            {
                paralel3 = true;

            }
            else if (C1 / C2 == 1)
            {
                paralel3 = true;

            }
            else
            {
                paralel3 = false;
```

Izvorni kod 5.12. Udaljenost između ravnina-uvjet paralelnosti

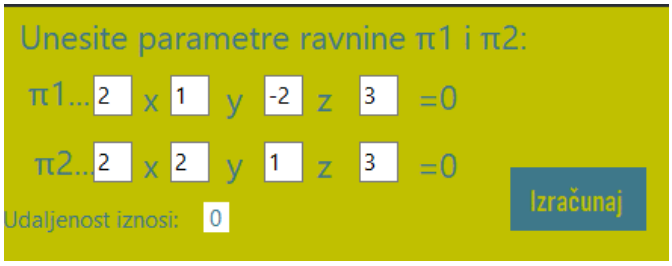
5.8.2. Rješenje problema

Primjer rješavanja problema je prikazan u dva slučaja, kada su ravnine paralelne i kada nisu. Potrebno je izračunati udaljenost između ravnina $\pi_1 \ldots 2x + y - 2z + 3 = 0$ i $\pi_2 \ldots 2x + y - 2z + 9 = 0$ te udaljenost između ravnina $\pi_1 \ldots 2x + y - 2z + 3 = 0$ i $\pi_1 \ldots 2x + 2y + z + 3 = 0$. Na slici 5.14. je prikazano rješenje problema kada su ravnine paralelne i aplikacija daje točne rezultate.



Slika 5.14. Izračun problema udaljenost između paralelnih ravnina

Na slici 5.15. je prikazano rješenje problema kada ravnine nisu paralelne i rješenje je očekivana udaljenost nula.



Slika 5.15. Izračun problema udaljenost između ravnina koje nisu paralelne

5.9. Jednadžba ravnine kroz tri točke

5.9.1. Programski dio

```
float v1X = tx2 - tx1;
float v1Y = ty2 - ty1;
float v1Z = tz2 - tz1;

float v2X = tx3 - tx1;
float v2Y = ty3 - ty1;
float v2Z = tz3 - tz1;
            //Vektor normale dobiven vektorskim produktom
float i = v1Y * v2Z - v2Y * v1Z;
float j = (v1X * v2Z - v2X * v1Z) * -1;
float k = v1X * v2Y - v2X * v1Y;
            //D varijabla ravnine
float d = (i * tx1 + j * ty1 + k * tz1) * -1;
```

Izvorni kod 5.13. Jednadžba ravnine kroz tri točke

Programski kod je podijeljen u dva dijela, računanje vektorskog produkta i namještanje teksta za ispis. Izvorni kod 5.13. predstavlja računanje x, y, z koordinati vektora $\overrightarrow{T_1T_2}, \overrightarrow{T_2T_3}$ i zatim računanje vektorskog produkta kako bi se dobila normala ravnine. Kako bi se dobio parametar D potrebno je uzeti jednu točku koja pripada toj ravnini, općenito točku T_1 i pomnožiti sa vektorom normale. Dobivene brojeve treba formirati u ispis opće jednadžbe ravnine. Isto kao i kada se računala jednadžba pravca treba se provjeriti je su li koeficijenti veći od nule i jednaki nuli. Ako su koeficijenti jednaki nuli tekst je prazan *string* , a ako su veći od nule dodaje se znak +. U svojstvo *Text* dodaju se tekstovi prema *if else* grananju. Prethodno napisano je prikazano izvornim kodom 5.14.

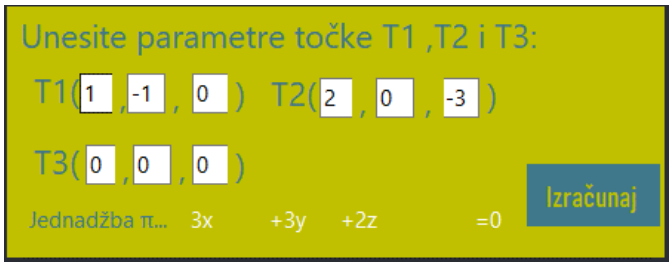
```
            if (i == 0)
            {
                A.Text = "";
            }
            else
            {
                A.Text = i.ToString() + "x";
            }
            if (j > 0)
            {
                B.Text = "+" + j.ToString() + "y";
            } else if (j == 0)
            {
                B.Text = "";
            }else{
             B.Text = j.ToString() + "y";}
```

```
if (k > 0){C.Text = "+" + k.ToString() + "z";
} else if (k == 0)
{
    C.Text = "";
}else
{
    C.Text = k.ToString() + "z";
}

if (d > 0)
{
    D.Text = "+" + d.ToString();
}else if (d == 0)
{
    D.Text = "";
}
else
{
    D.Text = d.ToString();}
```

Izvorni kod 5.14. Jednadžba ravnine kroz tri točke-uljepšavanje ispisa

5.9.2. Rješenje problema

Primjer problema kojeg aplikacija rješava je odrediti jednadžbu ravnine kroz točke: $T_1(1,-1,0), T_2(2,0,-3)$, $T_3(0,0,0)$. Na slici 5.16 prikazano je rješenje problema.



Slika 5.16. Izračun problema ravnina kroz tri točke

Na slici se može primijetiti kako jednadžba ravnine prolazi kroz ishodište pa je parametar D jednak nuli. Sa slike je vidljivo kako se svojstvo *Text* mijenja zbog dijela programskog koda gdje se namješta ljepši ispis. Dobivena jednadžba ravnine prikazana je u općem obliku i provjerom je aplikacija dala točno rješenje. Najveći nedostatak aplikacije je nemogućnost vizualnog prikaza problema.

6. ZAKLJUČAK

Cilj rada bio je napraviti aplikaciju za brže i lakše rješavanje osnovnih problema koji se pojavljuju u području analitičke geometrije prostora. Za izradu aplikacije korišten je programski jezik C# zbog brzine i moćnog razvojnog okruženja. Aplikacija daje točna rješenja i moguće ju je koristiti kao provjeru ručno riješenih zadataka. Nedostatak aplikacije je nemogućnost da korisniku vizualno prikaže problem te da prikaže rješenje koje je dobiveno. Nedostatak aplikacije moguće je riješiti budućim proširenjima i radom nad aplikacijom. Unatoč nedostatku aplikacija je korisna i brzo ostvaruje rješenja.

LITERATURA

[1] ThinkCalculator:Analytic Geometry Calculators,Thinkcalculator, 2006., dostupno na: https://www.thinkcalculator.com/analytical/ [svibanj, 2023.]

[2] M.P., Analytic Geometry- Online Calculator:MathPortal,MathPortal.org,dostupno na:https://www.mathportal.org/calculators/analytic-geometry/index.php [svibanj, 2023.]

[3] Open Omnia - Mathematics Solver with Steps,Open Omnia, 2019., dostupno na: https://openomnia.com/ [svibanj, 2023.]

[4] D.M., Online calculators:Analytic geometry. Cartesian coordinate.OnlineMSchool, 2011. dostupno na: https://onlinemschool.com/math/assistance/cartesian_coordinate/ [svibanj, 2023.]

[5] N. Elezović, LINEARNA ALGEBRA, Fakultet elektrotehnike računarstva, Zagreb, 2006.

[6] J. Sedlar, Analitička geometrija i linearna algebra 2016.-2017., Fakultet građevinarstva, arhitekture i geodezije u Splitu, Split, 2016., dostupno na:

https://gradst.unist.hr/Portals/9/docs/katedre/Matematika/PSGG%20AGLA/AGLA%20-%20skripta%20%282016-17%29_vIP.pdf [svibanj, 2023.]

[7] I.S., Duljina vektora, jedinični vektor, kut između vektora i kosinusi smjerova, Matematika1, Fakultet elektrotehnike, strojarstva i brodogradnje, Split, 2018., dostupno na: http://www.mathematics.digital/matematika1/predavanja/node56.html [svibanj, 2023.]

[8] S.Ž, Jednadžba pravca u prostoru, Prirodoslovni matematički fakultet, Zagreb, 2020. dostupno na:https://web.math.pmf.unizg.hr/~szunar/m1_materijali/210122%20-%206.3%20-%20Jednadzba%20pravca%20u%20prostoru.pdf [svibanj 2023.]

[9] Microsoft, A tour of C#:Overview, Microsoft Build, Redmond, 2023., dostupno na: https://learn.microsoft.com/en-us/dotnet/csharp/tour-of-csharp/ [svibanj, 2023.]

[10] Visual Studio, Wikipedia, 2023., dostupno na: https://en.wikipedia.org/wiki/Visual_Studio [svibanj, 2023.]

[11] Microsoft, Desktop Guide:Windows Forms .NET, Microsoft Build, Redmond, 2023., dostupno na:

https://learn.microsoft.com/en-us/dotnet/desktop/winforms/overview/?view=netdesktop-7.0 [svibanj, 2023.]

[12] P. M.,Canva,Sydney, 2023., dostupno na: https://www.canva.com/ [svibanj, 2023.]

[13] N. Elezović; A. Aglić, LINEARNA ALGEBRA ZBIRKA ZADATAKA, Fakultet elektrotehnike i računarstva, Zagreb, 2006.

[14] A. Šteko, auditorne vježbe #4, 5 ANALITIČKA GEOMETRIJA PROSTORA, Fakultet elektrotehnike, računarstva i informacijskih tehnologija Osijek, Osijek, 2020.

[15] J. S., Ravnina i pravac, Fakultet građevinarstva, arhitekture i geodezije u Splitu, Split, 2016., dostupno na:

https://gradst.unist.hr/Portals/9/docs/katedre/Matematika/PSGG%20AGLA/T02_Ravnina%20i%20pravac.pdf [svibanj, 2023.]

POPIS SLIKA

2.1. Izgled web aplikacije ThinkCalculator 2

2.2. Izgled aplikacije MathPortal 3

2.3. Izgled aplikacije OpenOmnia 3

2.4. Izgled web aplikacije OnlineMSchool 4

3.1. Ravnina je određena jednom točkom i vektorom normale 5

3.2. Ravnina kroz tri točke 7

3.3. Pravac određen jednom točkom i vektorom smjera 10

4.1. Glavna metoda u programskom jeziku C# 12

4.2. Razvojno okruženje Visual studio 13

4.3. Windows Forms forma 14

4.4 Pravac i ravnina 15

4.5. Koordinatni sustav sa točkom 15

5.1. Prilagođeni gumbi 16

5.2. Početni izgled aplikacije 18

5.3. Osnovne operacije i algoritmi 18

5.4. Forma AlgoritmiFirst.cs 19

5.5 Forma AlgoritmiSecond.cs 20

5.6 Izračun problema kut između dvije ravnine 21

5.7. Pogrešan unos 22

5.8. Izračun problema kut između ravnine i pravca 28

5.9. Izračun 1.,2. i 3. problema prolazi li ravnina kroz ishodište i pripada li točka ravnini 26

5.10. Izračun 4. problema prolazi li ravnina kroz ishodište i pripada li točka ravnini 26

5.11. Izračun problema pravac kroz dvije točke 29

5.12. Izračun problema udaljenost točke od ravnine 30

5.13. Izračun problema udaljenost između dvije točke 30

5.14. Izračun problema udaljenost između paralelnih ravnina 32

5.15. Izračun problema udaljenost između ravnina koje nisu paralelne 32

5.16. Izračun problema ravnina kroz tri točke 34

SAŽETAK

U ovom radu opisan je teorijski dio analitičke geometrije prostora te je napravljena aplikacija za rješavanje problema analitičke geometrije prostora. Aplikacija se sastoji od devet osnovnih algoritama koji su realizirani u obliku kalkulatora, gdje korisnici unose jednadžbe i koeficijente. U radu su objašnjene tehnologije koje su se koristile za izradu aplikacije. Objašnjeni su svi algoritmi problema koji su dani za rješavanje te je dan uz njih primjer rješavanja. Programski kodovi su objašnjeni redom i realizirani su po formulama analitičke geometrije prostora. Dano je i objašnjenje programskog koda za uljepšavanje teksta, zbog algoritama koji zahtijevaju ispis jednadžbe pravca i ravnine. U radu su objašnjene prednosti i nedostaci aplikacije. Za jednostavniju izradu i korištenje aplikacije korišten je programski jezik C# i implementirana je Windows Forms aplikacija s grafičkim sučeljem i četiri forme.

Ključne riječi: analitička geometrija prostora, aplikacija, C#, kalkulator

ABSTRACT

Analytic geometry of space in programming language C#

In this paper, the theoretical part of the analytical geometry of space is described and an application for solving problems of analytical geometry of space is made. The application consists of nine basic algorithms that are realized in the form of a calculator, where users enter equations and coefficients. The paper explains the technologies that were used to create the application. All the problem algorithms that are given to be solved are explained and an example of their solution is given. The program codes are explained in order and are realized according to the formulas of the analytical geometry of space. There is also an explanation of the program code for beautifying the text, due to the algorithms that require the equation of the line and the plane to be printed. The paper explains the advantages and disadvantages of the application. For simpler creation and use of the application, the C# programming language was used and a Windows Forms application with a graphical interface and four forms was implemented.

Keywords: analytic geometry of space, application, C#, calculator