LATVIJAS UNIVERSITĀTE

DATORIKAS FAKULTĀTE

**RĪKS BEZJĒ LĪKŅU KONSTRUĒŠANAI UN MODIFICĒŠANAI**

KVALIFIKĀCIJAS DARBS

Autors: **Elīza Gaile**

Studenta apliecības Nr.: eg17035

Darba vadītājs: doc., Dr. dat. Vineta Arnicāne

RĪGA 2019

**ANOTĀCIJA**

Kvalifikācijas darbā ir aprakstīta sistēma interaktīvam grafisku objektu rīkam, kas paredzēta kubisku Bezjē līkņu konstruēšanai, modificēšanai un dzēšanai izmantojot datorpeli, datora tastatūru vai teksta failus. Sistēma piedāvā ģenerēt četru dažādu veidu Bezjē līknes, tai skaitā interpolētas līknes un saliktas līknes ar nepārtrauktību (B-līnijas), papildus izmantojot trīs dažādu veidu parametrizācijas metodes. Iespējams arī izvadīt konstruēto Bezjē līkņu kontrolpunktus un/vai mezglu punktus, kā arī citus līkni raksturojošos lielumus. Līkņu vizuālai salīdzināšanai iespējams augšuplādēt fona attēlu.

Sistēma implementēta C# programmēšanas valodā, izmantojot Microsoft .NET satvaru un tajā esošo Windows Forms bibliotēku.

Atslēgas vārdi: C#, Microsoft .NET, Windows Forms, Bezjē līknes, interpolācija.

**ABSTRACT**

TOOL FOR CONSTRUCTION AND MODIFICATION OF BÉZIER CURVES

Qualification paper describes a system for interactive tool of graphical objects, meant for construction, modification and deletion of Bézier curves using mouse, keyboard or text files. System allows to create four types of curves, including Bézier curve fitting and composite Bézier curves with continuity (B-splines), and to choose from three different parameterization methods. Additionally, it is possible to output control points, knot points and other descriptive variables of constructed Bézier curves. For visual comparison of curves, it is possible to upload a background image.

System is implemented in C# programming language, using Microsoft .NET framework and Windows Forms library included in the framework.

Keywords: C#, Microsoft .NET, Windows Forms, Bézier curves, interpolation.

SATURA RĀDĪTĀJS

[IEVADS 6](#_Toc9534161)

[APZĪMĒJUMU SARAKSTS 8](#_Toc9534162)

[1. VISPĀRĒJS APRAKSTS 10](#_Toc9534163)

[1.1. Esošā stāvokļa apraksts 10](#_Toc9534164)

[1.2. Pasūtītājs 10](#_Toc9534165)

[1.3. Produkta perspektīva 10](#_Toc9534166)

[1.4. Darījumsprasības 10](#_Toc9534167)

[1.5. Sistēmas lietotāji 10](#_Toc9534168)

[1.6. Vispārējie ierobežojumi 10](#_Toc9534169)

[1.7. Pieņēmumi un atkarības 11](#_Toc9534170)

[2. PROGRAMMATŪRAS PRASĪBU SPECIFIKĀCIJA 12](#_Toc9534171)

[2.1. Funkcionālās prasības 12](#_Toc9534172)

[2.2. Nefunkcionālās prasības 12](#_Toc9534173)

[3. PROGRAMMATŪRAS PROJEKTĒJUMA APRAKSTS 13](#_Toc9534174)

[3.1. Projektējuma matemātiskais pamatojums 13](#_Toc9534175)

[3.1.1. Kubisku Bezjē līkņu izmantošanas pamatojums 13](#_Toc9534176)

[3.1.2. Bezjē līkņu pamatteorija 14](#_Toc9534177)

[3.1.3. Kubiskās Bezjē līknes interpolācija caur četriem punktiem 15](#_Toc9534178)

[3.1.4. Tuvākās līknes metodes izmantošanas pamatojums un teorija 16](#_Toc9534179)

[3.1.5. Saliktas līknes izmantošanas pamatojums un teorija 16](#_Toc9534180)

[3.1.6. B-līniju konstruēšanas un modificēšanas metode 17](#_Toc9534181)

[3.1.7. Parametrizācijas metodes 18](#_Toc9534182)

[3.1.8. Līkņu modificēšanas metodes 20](#_Toc9534183)

[3.2. Lietotāja saskarņu projektējums 22](#_Toc9534184)

[3.2.1. Galvenais logs 22](#_Toc9534185)

[3.2.2. Koordinātu logs 25](#_Toc9534186)

[3.3. Funkciju projektējums 26](#_Toc9534187)

[3.3.1. Galvenā loga pamatfunkcijas 27](#_Toc9534188)

[3.3.2. Jaunu līkņu pievienošana 28](#_Toc9534189)

[3.3.3. Līkņu konstruēšana 29](#_Toc9534190)

[3.3.4. Līkņu modificēšana 34](#_Toc9534191)

[3.3.5. Esošu līkņu funkcijas 37](#_Toc9534192)

[3.3.6. Koordinātu logs 38](#_Toc9534193)

[4. TESTĒŠANAS DOKUMENTĀCIJA 40](#_Toc9534194)

[4.1. Testpiemēru projektējums 40](#_Toc9534195)

[4.2. Testēšanas žurnāls 41](#_Toc9534196)

[5. PROJEKTA ORGANIZĀCIJA 42](#_Toc9534197)

[5.1. Darbietilpības novērojums 42](#_Toc9534198)

[5.1.1. Prognozētā darbietilpība 42](#_Toc9534199)

[5.1.2. Reālā darbietilpība 43](#_Toc9534200)

[5.2. Kvalitātes nodrošināšana 44](#_Toc9534201)

[5.3. Konfigurāciju pārvaldība 44](#_Toc9534202)

[6. REZULTĀTI UN SECINĀJUMI 45](#_Toc9534203)

[PATEICĪBAS 46](#_Toc9534204)

[IZMANTOTĀ LITERATŪRA UN AVOTI 47](#_Toc9534205)

[PIELIKUMI 48](#_Toc9534206)

[1. Programmatūras koda fragments 48](#_Toc9534207)

[2. Mazāko kvadrātu metodes pierādījums Bezjē līknes interpolācijai 49](#_Toc9534208)

[2.1. Skalārā reizinājuma atvasinājums 52](#_Toc9534209)

# IEVADS

**Nolūks**

Kvalifikācijas darba nolūks ir iepazīstināt sistēmas izstrādātājus un lietotājus ar sistēmas prasībām, projektējumu, testēšanas dokumentāciju un projekta organizāciju, funkcijām un rīka funkcionalitātes pielietojumiem, kā arī nodrošināt prasību saskaņošanu turpmākajā izstrādes un uzturēšanas procesā.

**Darbības sfēra**

Kvalifikācijas darbā aprakstītā sistēma paredzēta iekšējām pasūtītāja uzņēmuma vajadzībām, lai manuāli konstruētu un salīdzinātu līknes apģērba piegrieztnēs. Sistēmas lietotāji izstrādātajā rīkā var konstruēt, modificēt un dzēst četru veidu kubiskas Bezjē līknes, kas atšķiras ar konstruēšanas veidu:

1. uzdodot četrus kontrolpunktus,
2. uzdodot četrus punktus, caur kuriem jāizvelk līkne (interpolēta Bezjē līkne),
3. uzdodot vairāk kā četrus punktus, caur kuriem pēc iespējas tuvu jāizvelk Bezjē līkne (interpolācija izmantojot mazāko kvadrātu metodi),
4. saliktu Bezjē līkņu konstruēšana, nodrošinot nepārtrauktību katrām divām secīgām līknēm.

Veicot interpolēto līkņu konstruēšanu, iespējams izvēlēties piemērotāko no trim piedāvātajām Bezjē līkņu parametrizācijas metodēm. Iespējams veikt vizuālu piegrieztņu līkņu salīdzināšanu ar jau esošu piegrieztni.

**Saistība ar citiem dokumentiem**

Dokuments tika izstrādāts saskaņā ar programmatūras prasību specifikācijas standartu LVS 68:1996 „Programmatūras prasību specifikācijas ceļvedis” [1] un saskaņā ar programmatūras projektējuma apraksta standartu LVS 72:1996 “Ieteicamā prakse programmatūras projektējuma aprakstīšanai” [2].

**Pārskats**

Kvalifikācijas darba dokuments ir sadalīts sešās galvenajās daļās:

* Vispārējs apraksts – apraksta produktu vispārējā līmenī un iekļauj informāciju par sistēmas lietotājiem, darījumprasībām. Aprakstīti arī ierobežojumi, pieņēmumi un atkarības.
* Programmatūras prasību specifikācija – ietver funkcionālās un nefunkcionālās prasības.
* Programmatūras projektējuma apraksts – ietver daļēju funkciju un saskaņu projektējumu.
* Programmatūras testēšana – ietver testpiemēru projektējumu un testēšanas žurnālu ar rezultātiem.
* Projekta organizācija – apraksta programmatūras izstrādes procesus, iekļauj arī darbietilpības novērtējumu, kvalitātes nodrošināšanu un konfigurāciju pārvaldību.
* Rezultāti un secinājumi – apraksta izstrādās sistēmas rezultātus un secinājumus, kas radušies pēc kvalifikācijas darba izstrādes.

# APZĪMĒJUMU SARAKSTS

**B-līnija** Parametriska līkne, kas veidota gludi savienojot patvaļīgu skaitu Bezjē līkņu.

**Beigu punkts** Bezjē līknes pēdējais kontrolpunkts.

**Bezjē līkne** Parametriska līkne, ko uzdod izmantojot kontrolpunktus; populārs līkņu uzdošanas veids datorgrafikā.

**Blakus rokturis** Saliktas līknes rokturi sauc par mezgla punkta blakus rokturi, ja tas ir iepriekšējais vai nākamais kontrolpunkts skaitot no kontrolpunkta, kas sakrīt ar minēto mezglu.

**Blakus mezgls** Iepriekšējais vai nākamais mezgls kādam mezglam saliktā Bezjē līknē.

**nepārtrauktība** Funkcijai piemīt nepārtruktība, ja tās pirmās un otrās kārtas atvasinājumi eksistē un ir nepārtraukti. Raksturo vizuāli gludu funkciju.

**Galapunkts** Bezjē līknes pirmais vai pēdējais kontrolpunkts.

**Git** Versiju kontroles sistēma.

**Interpolācija** Punktu un līkņu konstruēšanas metode, izmantojot galīgu skaitu zināmu punktu.

**Kontrolpunkts** Elements no punktu kopas, kas nosaka Bezjē līknes formu; -tās pakāpes Bezjē līknei ir kontrolpunkti.

**Mezgla punkts** Punkts, caur kuru tiek interpolēta Bezjē līkne.

**NURBS** Nevienmērīgas racionālas B-līnijas – matemātisks līkņu aprakstīšanas modelis, ar ko iespējams definēt jebkādu patvaļīgu līniju.

**Pretējais rokturis** Saliktas līknes rokturus sauc par pretējiem, ja tie ir iepriekšējais un nākamais kontrolpunkts skaitot no divu secīgu segmentu savienojuma kontrolpunkta (kas ir arī mezglu punkts). Saliktas līknes pirmajam un pēdējam rokturim nav pretējā roktura.

**px** Pikselis, vismazākais attēla elements.

**Rokturis** Saliktas Bezjē līknes kontrolpunkts, kas nav segmenta galapunkts.

**Sākumpunkts** Bezjē līknes pirmais kontrolpunkts.

**.txt** Teksta failu veids.

Šajā dokumentā tiek lietoti sekojoši apzīmējumi:

* vektori tiek apzīmēti ar mazo burtu treknrakstā, piemēram, ;
* matricas tiek apzīmētas ar lielo burtu treknrakstā, piemēram, ;
* skalāri lielumi tiek apzīmēti ar mazo burtu slīprakstā, piemēram, ;
* funkcijas tiek apzīmētas ar lielo burtu sliprakstā, piemēram, ;
* vienkāršības labad, punkts tiek uztverts kā vektors.

# VISPĀRĒJS APRAKSTS

## Esošā stāvokļa apraksts

Pieejamas vairākas programmatūras (piemēram, Adobe Illustrator [3], GIMP [4], Inkscape [3]), kas piedāvā līkņu konstruēšanas un modificēšanas funkcionalitātes, tomēr šīm programmatūrām ir ierobežotas interpolācijas un līkņu daudzveidības iespējas.

## Pasūtītājs

Sistēma aprakstīta un izstrādāta pēc uzņēmuma SIA “FitDex” pasūtījuma studiju kursa “Prakse” (DatZ2033) ietvaros.

## Produkta perspektīva

Kvalifikācijas darbs ir neatkarīga sistēma, kas apvieno pasūtītājam nepieciešamās funkcionalitātes, lai veiktu apģērba piegrieztņu līkņu konstruēšanu, modificēšanu un vizuālu salīdzināšanu.

## Darījumsprasības

Programmatūrai jānodrošina iespēja veikt gludu un apģērbu piegrieztnēm līkņu interpolāciju, ģenerēto līkņu modificēšanu un raksturojošo lielumu ievades un izvades veikšanu ar datorpeli, datora tastatūru vai nepieciešamo datu ielasīšanu no faila. Jānodrošina iespēja vizuāli salīdzināt konstruētās līknes ar līknēm jau esošā piegrieztnē.

## Sistēmas lietotāji

Sistēmai paredzēts viens lietotāju veids. Lietotājs ir saistīts ar pasūtītāja uzņēmumu, ir pazīstams ar izstrādātā rīka specifiku (tai skaitā ar izmantoto Bezjē līkņu specifiku), un izmanto šo rīku, lai konstruētu līknes apģērbu piegrieztnēs. Lietotājs izmanto rīka saskarni un tam ir pieejamas visas programmatūras funkcionalitātes.

## Vispārējie ierobežojumi

Nav.

## Pieņēmumi un atkarības

Lai izmantotu visas programmatūras funkcionalitātes, lietotājam jābūt nodrošinātam ar datoru, ekrānu, datora tastatūru un datorpeli. Sistēma jāizstrādā C# programmēšanas valodā.

# PROGRAMMATŪRAS PRASĪBU SPECIFIKĀCIJA

## Funkcionālās prasības

Programmatūrai jānodrošina pasūtītāja prasības:

1. Līkņu interpolācija caur patvaļīgi uzdotiem punktiem. Punkti var tikt ievadīti manuāli ar tastatūru, atzīmēti ar datorpeli vai ielasīti no faila.
2. Konstruētajām līknēm jābūt gludām (tām jāpiemīt nepārtrauktībai) un tām jābūt apģērbu piegrieztnēm piemērotām – .
3. Konstruētās līknes jāspēj interaktīvi modificēt izmantojot datorpeli vai ievadot interpolējamo punktu izmaiņas ar tastatūru.
4. Jāspēj izvadīt konstruēto līkņu parametrus uz ekrāna un failā, lai nepieciešamības gadījumā varētu rekonstruēt izveidoto līkni.
5. Jānodrošina iespēja vizuāli salīdzināt rīkā konstruētās līknes ar līknēm esošā piegrieztnē.
6. Dažādu nozīmju grafiskajiem objektiem (piemēram, interpolācijas mezglu punktiem un palīgpunktiem) jābūt atšķiramiem ar vizuālo izskatu, piemēram, krāsojumu.
7. Jānodrošina iespēja konstruētās līknes dzēst.

## Nefunkcionālās prasības

Grafisko objektu konstruēšanai, modificēšanai, dzēšanai un datu izvadei jānotiek bez pamanāmas aiztures – ne ilgāk kā 0,2 sekunžu laikā [5].

# PROGRAMMATŪRAS PROJEKTĒJUMA APRAKSTS

## Projektējuma matemātiskais pamatojums

### Kubisku Bezjē līkņu izmantošanas pamatojums

Lai konstruētu divdimensionālus datorgrafiskus attēlus, parasti tiek izmantoti punkti, taisnes un līknes. Atšķirībā no dotiem punktiem un taisnēm, konkrētu līkņu attēlošana datorā nav triviāla, jo nepieciešama precīza matemātiska funkcija, kas šo līkni apraksta. Nepieciešamās līkņu funkcijas iespējams uzdot dažādos veidos – piemēram, ar polinomu un trigonometriskajām funkcijām, kā arī ar dažādām parametriskajām līknēm.

NURBS ir parametrisku līkņu uzdošanas veids, kas salīdzinoši viegli aprakstāms datoram un ir intuitīvi saprotams cilvēkam [6]. NURBS veidotās līknes ir gludas, paredzamas un ērti modificējamas, turklāt tām piemīt liela ekspresivitāte (atšķirībā no, piemēram, polinomu funkcijām, kur katram argumentam atbilst viena vērība).

Viens no visplašāk izmantotajiem līkņu uzdošanas veidiem datorgrafikā ir Bezjē līknes [3], kas ir NURBS līkņu apakškopa. Lai gan mazāk elastīgas par NURBS līknēm, arī Bezjē līknes ir plaši pielāgojamas un paredzamas, turklāt samazinātās sarežģītības dēļ, tās ir vieglāk implementējamas, prasa mazāk resursus un ir vēl intuitīvākas. Galvenokārt tieši šo īpašību dēļ, Bezjē līknes ir implementētas lielākajā daļā grafisko programmatūru, piemēram, Adobe Illustrator, GIMP, Inkscape un citās.

Bezjē līkni iespējams definēt jebkurai pakāpei. Pirmās kārtas Bezjē līkne ir nogrieznis, savukārt ceturtās un augstāku kārtu Bezjē līknes patērē daudz skaitļošanas resursu. Tādēļ datorgrafikā vispopulārākās ir kvadrātiskās un kubiskās Bezjē līknes. Atceroties, ka līknes pakāpe nosaka arī tās maksimālo ekstrēmu un pārliekuma punktu skaitu, var secināt, ka kubiskās Bezjē līknes piedāvā lielāku pielietojamību nekā kvadrātiskās, turklāt tās ir implementētas C# System.Drawing bibliotēkā. Ņemot vērā šos apsvērumus, programmatūra tiek projektēta izmantojot kubiskās Bezjē līknes.

Šajā programmatūrā Bezjē līknes var tikt konstruētas četros veidos:

1. Uzdodot četrus kontrolpunktus (<4 cPoints>);
2. Uzdodot četrus mezglu punktus (<4 pPoints>);
3. Atrodot tuvāko līkni vairāk nekā četriem uzdotiem mezglu punktiem (<Least Square>);
4. Veidojot saliktu Bezjē līkni (<Composite>).

### Bezjē līkņu pamatteorija

Bezjē līkne ir definēta jebkurai pakāpei un-tās kārtas Bezjē līkne pierakstāma ar formulu:

Kubiska Bezjē līkne ir izsakāma formā:

Ar tiek apzīmēti kontrolpunkti. Kontrolpunkti ir mainīgie, kas nosaka, kā Bezjē līkne tiek izliekta. Katru kontrolpunktu komplekts nosaka tieši vienu līkni. Varam ievērot, ka kontrolpunkti sākas ar indeksu 0.

Trešās pakāpes Bezjē līknei ir četri kontrolpunkti, divus no tiem sauc par galapunktiem (, ) un divus par rokturiem (, ). Līknes sākumpunkts ir un beigu punkts ir , līdz ar to šie punkti pieder līknei. Atšķirībā no galapunktiem, patvaļīgi izvēlēti rokturu punkti līknei nepieder. Skatīt 3.1. att.

*A picture containing sky

Description automatically generated*

3.1. att. **Bezjē līkne un tās kontrolpunkti**

Bezjē līknes iespējams reprezentēt arī matricu formā, kas atvieglo dažādu aprēķinu veikšanu. Kubisku Bezjē līkni var izteikt kā trīs matricu **T**, **M**, **C** reizinājumu:

### Kubiskās Bezjē līknes interpolācija caur četriem punktiem

Lai veiktu interpolāciju ar Bezjē līkni –, lai uzkonstruētu līkni, kas iet cauri dotiem mezglu punktiem –, jāiegūst atbilstošas kontrolpunktu koordinātas. Trešās kārtas gadījumā līkne precīzi var iziet caur ne vairāk kā četriem patvaļīgiem mezglu punktiem.

Ja lietotājs vēlas novilkt līkni caur tieši četriem punktiem, iespējams atrast atbilstošos kontrolpunktus. Viens no variantiem, kā atrast šādu kontrolpunktu koordinātas, ir izmantojot mazāko kvadrātu metodi. Šī metode atrod kontrolpunktu koordinātas, minimizējot distanci starp dotajiem mezglu punktiem un līkni. Zināms, ka trešās pakāpes Bezjē līkni iespējams izvilkt caur jebkādiem četriem mezglu punktiem, tātad mazāko kvadrātu metode atradīs līkni, kurai distanču kvadrātu summa ir nulle. Bezjē līkne ir parametriska un, lai aprakstītu punktu uz tās, tiek specificēta vērtība, savukārt mezglu punkti tiek uzdoti ar koordinātām. Lai mērītu un minimizētu distanci starp mezgla punktu un līkni, nepieciešams saistīt dotos punktus ar punktiem uz līknes, tas ir, katram mezgla punktam nepieciešams piesaistīt vērtību. Šo procesu sauc par parametrizāciju (skatīt nodaļu 3.1.7) un tās rezultātā atrasto vērtību katram mezgla punktam apzīmēsim ar .

No šīm vērtībām tiek izveidota matrica , kas interpolācijas aprēķinos aizstāj matricu ar simboliskajām vērtībām [12]:

Lai aprēķinātu kontrolpunktus (sakārtotus matricā ), kuru veidotā līkne interpolē mezglu punktus (sakārtotus matricā ), jāveic sekojoši aprēķini:

Aprēķinu izvedumu un mazāko kvadrātu metodes pierādījumu skatīt 2. pielikumā.

### Tuvākās līknes metodes izmantošanas pamatojums un teorija

Ja lietotājs vēlas novilkt līkni caur vairāk nekā četriem punktiem, viens no iespējamajiem risinājumiem ir atrast tuvāko līkni dotajiem mezglu punktiem – līkni, kura ir pēc iespējas tuvu visiem punktiem. Tas izdarāms dažādos veidos, bet viens no ērtākajiem un plašāk izmantotajiem veidiem ir mazāko kvadrātu metode. Metodes apraksts sakrīt ar četru punktu interpolāciju (skatīt 3.1.3 nodaļu), tikai patvaļīgu punktu gadījumā (), distanču kvadrātu summa nebūs nulle un matrica būs formā:

Tuvākā līkne analītiski neizies cauri visiem dotajiem punktiem, tomēr, ņemot vērā apģērba piegrieztņu modelēšanas specifiku – nepieciešamo precizitāti, iespējamās nobīdes un salīdzinoši mazās virziena izmaiņas līknēm, tā ir noderīga, jo prasa maz resursu un ir lietotājam vienkārši saprotama.

### Saliktas līknes izmantošanas pamatojums un teorija

Ja lietotājs vēlas novilkt līkni caur vairāk nekā četriem punktiem, un ir svarīgi, ka konstruētā līkne analītiski iziet caur šiem punktiem, patvaļīgiem mezgliem tas nav izdarāms ar vienu kubisku Bezjē līkni. Tā vietā var izmantot vairākas Bezjē līknes, kuras savā starpā ir savienotas. Katru no šīm savienotajām līknēm sauc par segmentu. Lai gan iespējams katriem četriem punktiem uzdot vienu Bezjē līkni, tādējādi minimizējot segmentu skaitu, šāds sadalījums rada daudz speciālgadījumu (ja mezglu skaits nedalās ar četri), būtiski sarežģī aprēķinus un ierobežo pielāgošanas iespējas. Ērtāks veids, kā ieviest saliktas Bezjē līknes, ir uzdot jaunu līkni katriem diviem secīgiem mezglu punktiem, kur mezglu punkti ir Bezjē līknes galapunkti. Lietojot šo metodi, nav jāuztraucas par mezglu punktu piederību līknēm.

Lai savienotās Bezjē līknes būtu gludas, tām jāpiemīt nepārtrauktībai. Lai nodrošnātu šo nepārtrauktību, katram mezglu punktam (izņemot pirmo un pēdējo) un tā blakus rokturiem jābūt uz vienas taisnes. Šāda veida saliktas Bezjē līknes sauc par B-līnijām, un tās ir iespējams konstruēt dažādos veidos. Kādā leņķī un cik tālu no mezgla punkta ir blakus rokturi ir maināmi lielumi un raksturo B-līniju veidu.

### B-līniju konstruēšanas un modificēšanas metode

B-līniju veids, kas tiek konstruēts izstrādātajā rīkā, aprakstāms šādi:

* Katrs mezgla punkts (izņemot pirmais un pēdējais) un tā blakus rokturi ir uz vienas taisnes (skatīt 3.2. att.);
* Mezgla punkta blakus rokturu veidotā taisne ir paralēla blakus mezglu veidotajai taisnei (skatīt 3.2. att.);
* Attālums starp blakus rokturiem vienāds ar pusi no attāluma starp blakus mezgliem (skatīt 3.2. att.);
* Attālums starp mezglu un blakus rokturiem proporcionāls attālumam starp mezglu un blakus mezgliem (skatīt 3.2. att.);
* Pirmais/pēdējais B-līnijas rokturis tiek atlikts simetriski otrajam/priekšpēdējam B-līknes rokturim (skatīt 3.2. att.);
* Ja B-līnija sastāv no 2 mezglu punktiem, attālums no roktura līdz tam tuvākajam mezgla punktam vienāds ar pusi no attāluma starp mezglu punktiem; leņķis starp mezglu punktu veidoto taisni un taisni, ko veido rokturis un tā tuvākais mezglu punkts, vienāds ar (skatīt 3.3. att.).

A close up of a device

Description automatically generated

3.2 att. **B-līniju konstruēšanas metodes piemērs**

A screenshot of a cell phone

Description automatically generated

3.3. att. **B-līniju konstruēšanas piemērs divu mezglu gadījumā**

Aprakstītais B-līniju veids ir cilvēkam intuitīvs, tā implementācija ir vienkārša un neprasa daudz resursu. Rīka konstruētās saliktās Bezjē līknes ir gludas un, izņemot retus gadījumus (piemēram, asus pagriezienus), atbilstošas vizuālajām prasībām.

### Parametrizācijas metodes

Bezjē līkņu parametrizācija ir mezglu punktu koordinātu sasaiste ar Bezjē līkni . Parametrizācijas rezultātā katram mezglam tiek atrasta atbilstoša vērtība, apzīmēta ar . Parametrizācija nepieciešama, lai veiktu Bezjē līkņu interpolāciju. Caur četriem punktiem iespējams izvilkt bezgalīgi daudz kubisku bezjē līkņu un tieši parametrizācijas metode nosaka, kāda līkne tiks izvilkta. Trīs parametrizācijas metodes, kas tiek izmantotas programmatūrā, ir:

1. Vienmērīga sadalījuma parametrizācijas metode (*uniform method*).

Tiek pieņemts, ka dotie punkti būs vienmērīgi izkārtoti, un atbilstošā vērtība punktam aprēķināma kā , kur – mezglu skaits. Piemēram, ja tiek doti 5 punkti, attiecīgās vērtības ir 0, 0.25, 0.5, 0.75, 1. Šī ir viena no vienkāršākajām un skaitļošanas resursu taupīgākajām parametrizācijas metodēm, bet tā var dot nevēlamus rezultātus – piemēram, ja mezgli nav vienmērīgos attālumos, līknē var rasties lieki izliekumi, cilpas un asas smailes [8], skatīt 3.4. att. Lai gan šādi nevēlami rezultāti nav unikāli vienmērīgā sadalījuma metodei, tie atgadās biežāk nekā citām metodēm [8].

1. Hordu garuma parametrizācijas metode (*chord length method*).

Tiek pieņemts, ka konstruētās līknes punkti būs netālu no lauztās līnijas, kas veidojas savienojot mezglu punktus. Tādā gadījumā līknes garums starp diviem mezgliem būtu tuvs hordas garumam starp tiem un visas interpolētās līknes garums būtu tuvs minētajai lauztajai līnijai. Atbilstošo vērtību punktam atrod kā lauztās līnijas garumu no pirmā līdz -tajam mezglam, mērogotu intervālā. Aprēķināms kā , kur , ar apzīmē garumu starp dotajiem punktiem, – lauztās līnijas garums, – mezglu skaits. Šī ir vispopulārākā Bezjē līkņu parametrizācijas metode []. Tai nav apjomīgas skaitļošanas operācijas un visbiežāk līnijas, kas konstruētas izmantojot hordu garuma parametrizāciju, dod vēlamus rezultātus. Bieža nevēlama parādība šai metodei ir lieli izliekumi [9], skatīt 3.4. att.

1. Centrtieces spēka parametrizācijas metode (*centripetal method*).

Šī parametrizācijas metode atsaucas uz centrtieces spēku. Ja mēs iedomājamies mašīnu, kas brauc pa Bezjē līkni, pagriezienos tai būtu jāsamazina ātrums, lai centrtieces spēks nav pārlieku liels un tā neizslīd no trajektorijas – spēkam vajadzētu būt proporcionālam izmaiņām leņķī. Centrtieces spēka parametrizācija ir aproksimācija šim modelim. Atbilstošo vērtību punktam aprēķina kā , kur , ar apzīmē garumu starp dotajiem punktiem, , – mezglu skaits. Var ievērot, ka šī metode ir ļoti līdzīga hordu garuma metodei, bet kvadrātsaknes dēļ garākām hordām tiek samazināta ietekme, savukārt īsākām – palielināta. Šī iemesla dēļ centrtieces spēka parametrizācija labāk tiek galā ar asiem pagriezieniem [10], skatīt 3.4. att.

A picture containing text

Description automatically generated

3.4. att. **Parametrizācijas metožu raksturiezīmes**

Visām parametrizācijas metodēm ir savas priekšrocības un trūkumi un nav viena labākā metode, it īpaši ņemot vērā, ka līknes labums ir daļēji subjektīvs. Kā redzams 3.5. attēlā, veicot interpolāciju caur dažādiem mezglu punktu izvietojumiem, piemērotākā parametrizācijas metode (tā, kuras veidotā Bezjē līkne ir vistuvākā mezglu punktu veidotajai lauztajai līnijai) var būt gan vienmērīga sadalījuma (I), gan hordu garuma (II), gan centrtieces spēka (III). Šo iemeslu dēļ, izstrādātais rīks ļauj mainīt interpolēto līkņu parametrizācijas metodi.

Eksistē parametrizācijas metodes, kuru konstruētās līknes retāk ir nevēlamas, piemēram, afīnā invarianta leņķa metode (“affine invariant angle method”), taču tās ir sarežģīti implementējamas un prasa daudz skaitļošanas resursu.

A close up of a map

Description automatically generated

3.5. att. **Kvalifikācijas darbā izmantoto parametrizācijas metožu salīdzinājums**

### Līkņu modificēšanas metodes

Lai nodrošinātu pilnvērtīgu rīka darbību, konstruētās līknes jāspēj modificēt. Modificēšanas veidi un iespējas atkarīgas no līknes konstruēšanas un modificēšanas veida:

* Līknes <4 cPoints> iespējams modificēt, patvaļīgi pārvietojot līknes kontrolpunktus.
* Līknes <4 pPoints> un <Least Squares> iespējams modificēt, patvaļīgi pārvietojot līknes mezglu punktus.
* Līkņu <Composite> iespējams modificēšana iespējama trīs veidos:
  + Patvaļīgi pārvietojot līknes rokturi. Roktura, kurš tiek modificēts, pretējais rokturis arī maina atrašanās vietu, lai abi rokturi un mezgla punkts starp tiem vienmēr atrastos uz vienas taisnes. Šis nosacījums nepieciešams, lai saglabātu līknes nepārtrauktību. Attālums starp mezglu un pretējo rokturi paliek nemainīgs. Skatīt 3.6. att.
  + Ierobežoti pārvietojot līknes rokturi. Šajā modificēšanas veidā rokturi iespējams pārvietot tikai pa staru, kas atrodas uz taisnes, ko veido roktura tuvākais mezgla punkts un pretējais rokturis. Stara sākumpunkts ir minētais mezgls un pretējais rokturis staram nepieder. Šāds modificēšanas veids nodrošina, ka atrašanās vietu maina tikai izvēlētais rokturis, bet līkne nezaudē nepārtrauktību. Skatīt 3.7. att.
  + Patvaļīgi pārvietojot mezgla punktu. Lai līknei arī pēc mezgla modificēšanas piemistu nepārtrauktība, bet pārējie līknes segmenti modifikācijas rezultātā nemainītos, ērtākais veids ir nemainīt mezgla blakus rokturu relatīvo atrašanās vietu pret minēto mezglu. Skatīt 3.8. att.

A close up of a map

Description automatically generated

3.6. att. **Saliktas līknes kontrolpunkta patvaļīga modificēšana**

**A screenshot of a cell phone

Description automatically generated**

3.7. att. **Saliktas līknes kontrolpunkta ierobežota modificēšana**

**A close up of a map

Description automatically generated**

3.8. att. **Saliktas līknes mezgla punkta modificēšana**

## Lietotāja saskarņu projektējums

Programmatūrai ir divi izvades logi – galvenais logs (skat. 3.9. att.) un koordinātu logs (skat. 3.10. att.). Lielākā daļa funkcionalitāšu noris galvenajā logā un koordinātu logs tiek no tā izsaukts tikai nepieciešamības gadījumā – lai ievadītu vai modificētu punktu koordinātas ar datora tastarūru un punktu koordināšu izvadei uz ekrānu.

### Galvenais logs

Galvenais logs paredzēts, visu rīkā izveidoto grafisko objektu attēlošanai un visu funkcionalitāšu darbības sākšanai (uzspiežot uz atbilstošās pogas). Vairākas funkcionalitātes realizējamas izmantojot tikai galveno logu – jaunu līniju konstruēšana un esošu līniju modificēšana ar peli (tai skaitā parametrizācijas izvēle un maiņa), esošu punktu izvēle, līkņu dzēšana, loga inicializācija no jauna, fona attēla redzamības maiņa. Jaunu līkņu konstruēšana no .txt faila, esošu punktu koordinātu izvade uz .txt failu un fona attēla augšuplādēšana arī neizmanto koordinātu logu. Galvenais logs ir uz tā izmēra maiņu reaģējošs, tam piesaistītas funkcijas, kas aprakstītas 3.3.1.-3.3.4. nodaļā.

**A screenshot of a cell phone

Description automatically generated**

3.9. att. **Galvenā loga saskarne**

Galvenajā logā ietvertas vadīklas:

1. – Lauks, kas paredzēts fona attēla un visu grafisko objektu attēlošanai. Reaģējoši maina izmēru, kad tiek veikta galvenā loga izmēra maiņa.

2. – Virsraksts “New Bezier of type”, pie kura atrodas piecas pogas. Pirmo četru pogu “4 cPoints”, “4 pPoints”, “Composite” un “Least Squares” nospiešana izsauc attiecīgi funkcijas NC02\_b4C, NC02\_b4P, NC02\_bLS un NC02\_bC. Pēc šo pogu nospiešanas iespējams iespējams pievienot jaunu līkni. Visu līkņu punktu ievades veids atkarīgs no 8. vadīklu grupas radio pogu statusiem, interpolēto līkņu parametrizācijas metode atkarīga no 9. vadīklu grupas radio pogu statusa. Piektā poga “Done” izsauc funkciju NC06\_bDC un apzīmē, ka pēdējā konstruētā saliktā līkne ir pabeigta.

3. – Virsraksts “Modify existing line”, pie kura atrodas divas pogas – “Modify cPoints” un “Modify pPoints”. Pogu nospiešana izsauc attiecīgi funkcijas MC05\_bMC un MC06\_bMP un pēc šīs darbības iespējams modificēt attiecīgi esošu līkņu kontrolpunktus vai mezglu punktus. Modificēšanas veids atkarīgs no 10. vadīklu grupas radio pogu statusiem. Detalizētāk par modificēšanas iespējām aprakstīts 3.1.8. nodaļā.

4. – Virsraksts “Get coordinates”, pie kura atrodas divas pogas – “cPoints” un “pPoints”. Pogu nospiešana izsauc attiecīgi funkcijas EC02\_bOC un EC03\_bOP un ļauj izvēlēties līkni, kuras kontrolpunktus vai mezglu punktus izvadīt. Izvades veids atkarīgs no 11. vadīklu grupas radio pogu statusiem.

5. – Poga “Choose a Line to Delete”. Pogas nospiešana izsauc funkciju EC06\_bDL un ļauj izvēlēties līkni, ko izdzēst.

6. – Poga “Upload Background” un izvēles rūtiņa “Show Background”. Nospiežot minēto pogu, tiek izsaukta funkcija FM06\_bUB, kas ļauj augšuplādēt attēlu. Augšuplādētais attēls tiek attēlots 1. vadīklā un tā redzamību iespējams kontrolēt ar minēto izvēles rūtiņu – mainot tās statusu, tiek izsaukta funkcija FM07\_cSB. Veicot loga inicializāciju, izvēles rūtiņa ir neaktīva.

7. – Poga “Reset All”. Nospiežot šo pogu tiek izsaukta funkcija FM08\_bRA, kas inicalizē galveno logu no jauna.

8. – Virsraksts “Choose points”, pie kura atrodas grupa ar trīs radio pogām – “w/ Mouse”, “w/ Keyboard” un “From .txt file”. Radio pogu statusi nosaka veidu, kā līknei tiks pievienoti punkti – tos atzīmējot ar datorpeli 1. vadīklā, ievadot punktu koordinātas ar datora tastatūru (tiek izsaukts koordinātu logs) vai ielasot punktu koordinātas no .txt faila. Veicot loga inicializāciju, aktīva ir pirmā no radiopogām.

9. – Virsraksts “”Parametrization”, pie kura atrodas grupa ar trīs radio pogām – “Uniform”, “Chord Length”, “Centripetal” un 10. vadīkla. Minētās radio pogas nosaka interpolētu līkņu parametrizacijas metodi, attiecīgi vienmērīga sadalījuma, hordu garuma vai centrtieces spēka. Mainot pirmo divu radio pogu statusu, tiek izsauktas funkcijas MC03\_rUC vai MC04\_rCC, kas maina izvēlētas līknes parametrizācijas metodi. Detalizētāk par parametrizācijas metodēm aprakstīts 3.1.7. nodaļā. Veicot loga inicializāciju, aktīva ir pirmā no radiopogām.

10. – Poga “Choose Line”. Nospiežot pogu tiek izsaukta funkcija MC01\_bCP, kas ļauj mainīt interpolētu līkņu parametrizācijas metodi, kā arī attēlo izvēlētās līknes parametrizācijas metodi, aktivizējot attiecīgo radio pogu no 9. vadīklu grupas. Veicot loga inicializāciju, aktīva ir pirmā no radiopogām.

11. – Virsraksts “Modify points”, pie kura atrodas grupa ar divām radio pogām – “w/ Mouse” un “w/ Keyboard”. Radio pogu statusi nosaka, kā tiks modificēti līknes punkti – tos pārvietojot ar peli vai ievadot punktu koordinātas ar datora tastatūru (tiek izsaukts koordinātu logs).

12. – Virsraksts “Output to”, pie kura atrodas grupa ar divām radio pogām – “Screen” un “.txt file”. Radio pogu statusi nosaka, vai līknes punkti tiks izvadīti uz ekrāna (izsaucot koordinātu logu) vai saglabāti .txt failā. Veicot loga inicializāciju, aktīva ir pirmā no radiopogām.

### Koordinātu logs

Koordinātu logs paredzēts punktu koordinātu ievadīšanai vai izvadīšanai. Šis logs tiek izsaukts no galvenā loga, tam ir piesaistītas 3.3.5. nodaļā aprakstītās funkcijas un tā izsaukšanai iespējami trīs mērķi:

1. Ja, veicot jaunas līknes pievienošanu (nospožiet kādu no pirmajām četrām 2. vadīklu grupas pogām), radio poga “w/ Keyboard” no 8. vadīklu grupas ir aktīva, koordinātu logs paradzēts punktu ievadei ar datora tastatūru, un, tam atveroties, tiks izsaukta funkcija FC01\_IA.
2. Ja, veicot esošas līknes modificēšanu (nospiežot kādu no 3. vadīklu grupas pogām), radio poga “w/ Keyboard” no 11. vadīklu grupas ir aktīva, koordinātu logs paredzēts punktu koordinātu modifikācijai ar datora tastatūru, un, tam atveroties, tiks izsaukta funkcija FC02\_IM.
3. Ja, veicot esošas līknes punktu izvadi (nospiežot kādu no 4. vadīklu grupas pogām), radio poga “Screen” no 12. vadīklu grupas ir aktīva, koordinātu logs paredzēts punktu koordinātu izvadei uz ekrāna, un, tam atveroties, tiks izsaukta funkcija FC03\_IO.

A screenshot of a cell phone

Description automatically generated

3.10. att. **Koordinātu loga saskarne**

13. – Koordinātu loga virsraksts. Tā saturs atkarīgs no loga izsaukšanas mērķa – “New <lineType> line”, “Modify <lineType> line” vai “Output <lineType> line”; <lineType> vietā ir izvēlētās līknes konstruēšanas veida nosaukums (skatīt 3.1.2. nodaļu).

Bez tā, koordinātu logā vēl ietvertas vadīklas:

14. – Virsraksts. Tā saturs atkarīgs no loga izsaukšanas mērķa – “Input <lineType> <pointType> point coordinates:”, “Modify <lineType> <pointType> point coordinates:” vai “List of <lineType> <pointType> point coordinates:”; <lineType> vietā ir izvēlētās līknes konstruēšanas veida nosaukums (skatīt 3.1.2. nodaļu), <pointType> vietā ir “control” kontrolpunktu gadījumā un “knot” mezglu punktu gadījumā.

15. – Tabula ar vadīkļiem un ritjosla. Tabulas augšpusē ir virsraksti “X” un “Y”, savukārt pirmā kollona ir aizpildīta ar virsrakstiem formā “<pointType><i>”; <pointType> vietā ir “C” kontrolpunktu gadījumā un “P” mezglu punktu gadījumā, <i> vietā ir skaitlis, kas apzīmē rindiņu. Blakus šiem virsrakstiem ir teksta lauki, kuros pēc nepieciešamības tiek ievadītas vai izvadītas punktu koordinātas. Inicializējot koordinātu formu līkņu pievienošanai, tiek izveidotas četras teksta lauku rindiņas, veicot punktu izvadi vai modifikāciju, teksta lauku rindiņu skaits sakrīt ar līknes apskatīto punktu skaitu, izņemot saliktu līkņu mezglu punktu modificēšanas gadījumā, kas tiek veikta pa vienam punktam.

16. – Pogas “Add New Row” un “Delete Row”. Nospiežot pogas tiek attiecīgi izsauktas funkcijas FC05\_bAR un FC06\_bDR. Šīs pogas ir redzamas un izmantojamas tikai pievienojot <Least Squares> vai <Composite> līknes un tās 15. vadīkļu grupas tabulai pievieno jaunas rindas, kas tiek aizpildītas ar nepieciešamajām vadīklām.Tiek ievēroti punktu skaita ierobežojumi.

17. – Pogas “Reset” un “OK”. Nospiežot pogas tiek izsauktas attiecīgi funkcijas FC08\_bRI un FC07\_bSI. Pogas nav redzamas, ja koordinātu logs izsaukts ar mērķi izvadīt esošas līknes punktu koordinātas. Poga “Reset” nodzēš visus teksta laukus 15. vadīkļu grupā, savukārt poga “OK” pievieno jaunu līkni vai modificē izvēlēto.

## Funkciju projektējums

Esošie pasūtītāja uzņēmuma informāciju tehnoloģiju projekti tiek izstrādāti C# valodā, tāpēc arī kvalifikācijas darbs tiks implementēts C# valodā. Izpētot C# iespējas, par piemērotāko programmēšanas rīku tika atzīts Microsoft .NET satvars un tajā iekļautā Windows Forms bibliotēka. Šie rīki piedāvā plašu vadīklu izvēli, tai skaitā vadīklas grafisku objektu attēlošanai, un saskarņu elementus – piemēram, tekstus, krāsas, darbības ar datorpeli.

### Galvenā loga pamatfunkcijas

Galvenā loga pamatfunkcijas iekļauj visas tās funkcijas, kas saistītas ar vadīklu, kurā tiek attēloti visi grafiskie objekti (pbCanva), kā arī funkcijas, kas saistītas tikai ar galveno logu – tā inicializāciju no jauna un reaģēšanu uz izmēra maiņu. Visu funkciju uzskaite un īsi apraksti atrodami 3.1. tabulā, sarežģītākajām funkcijām pieejami arī detalizētāki apraksti.

*3.1. tabula –* **Galvenā loga pamatfunkcijas**

|  |  |  |
| --- | --- | --- |
| **Nosaukums** | **Funkcija** | **Apraksts** |
| FM02\_pMM | void **pbCanva\_MouseMove**(object sender, MouseEventArgs e) | Izpildās, kad vadīklā pbCanva tiek kustināts peles kursors. Ļauj veikt interaktīvu līkņu modificēšanu ar peli un nosaka raustītās līnijas galapunktu, atliekot līknes <4 cPoints> punktus. |
| FM03\_pMU | void **pbCanva\_MouseUp**(object sender, MouseEventArgs e) | Izpildās, kad vadīklā pbCanva tiek atlaists datorpeles taustiņš. Pārtrauc līknes modificēšanu ar peli. |
| FM04\_pP | void **pbCanva\_Paint**(object sender, PaintEventArgs e) | Attēlo visus grafiskos objektus vadīklā pbCanva – Bezjē līknes, kontrolpunktus, mezglu punktus, kontrolpunktu veidotos nogriežņus un raustīto līniju atliekot līknes <4 cPoints> punktus. |
| FM05\_fRS | void **FormMain\_Resize**(object sender, EventArgs e) | Izpildās, kad tiek veikta galvenā loga izmēru maiņa. Nodrošina, ka minētais logs ir reaģējošs (uz izmēra izmaiņām). |
| FM06\_bUB | void **btnUploadBackground\_Click**(object sender, EventArgs e) | Augšuplādē un iestata lietotāja izvēlētu fona attēlu vadīklā pbCanva. |
| FM07\_cSB | void **cbShowBackground\_ CheckStateChanged**(object sender, EventArgs e) | Maina augšuplādētā fona attēla redzamību uz pretējo. |
| FM08\_bRA | void **btnResetAll\_Click**(object sender, EventArgs e) | Inicializē galveno logu no jauna. |

void **pbCanva\_Paint**(object sender, PaintEventArgs e):

Funkcija attēlo grafisku objektus:

* Visas Bezjē līknes melnā krāsā ar līnijas biezumu 1px.
* Visus kontrolpunktus izņemot <4 pPoints> līkņu galapunktus un <Composite> līkņu kontrolpunktus, kas nav rokturi. Kontrolpunkti tiek attēloti kā sarkanas riņķa līnijas ar centru kontrolpunktā, rādiusu 2px un līnijas biezumu 1px.
* Visus mezglu punktus kā melnus riņķus ar centru kontrolpunktā un rādiusu 2px.
* Visus nogriežņus starp katriem diviem secīgiem kontrolpunktiem līknēs <4 cPoints>, <4 pPoints> un <Least Squares> un starp katriem diviem secīgiem rokturiem <Composite> līknēs gaiši pelēkā krāsā ar līnijas biezumu 1px.
* Nogriezni starp pēdējo kontrolpunktu nepabeigtā <4 cPoints> līknē un datorpeles kursora atrašanās vietu ar raustītu līniju biezumā 1px, gaiši pelēkā krāsā.

### Jaunu līkņu pievienošana

Jaunu līkņu pievienošana iekļauj visas tās funkcijas, kas saistītas ar Bezjē līkņu pievienošanu – līknes kontrolpunktu vai mezglu pievienošanu un līknes raksturojošo parametru saglabāšanu, bet neiekļauj funkcijas, kas nepieciešamas konstruēšanas aprēķiniem. Visu funkciju uzskaite un īsi apraksti atrodami 3.2. tabulā, sarežģītākajām funkcijām pieejami arī detalizētāki apraksti.

*3.2. tabula –* **Jaunu līkņu pievienošanas funkcijas**

|  |  |  |
| --- | --- | --- |
| **Nosaukums** | **Funkcija** | **Apraksts** |
| NC01\_NL | void **NewLine**(BezierType lineType) | Izveido jaunu līkni. |
| NC02\_b4C | void **btnNew4cPoints\_Click**(object sender, EventArgs e) | Izpildās, kad tiek nospiesta vadīkla btnNew4cPoints. Atļauj veikt līknes <4 cPoints> kontrolpunktu ievadi ar datorpeli, klaviatūru vai no .txt faila. |
| NC03\_b4P | void **btnNew4pPoints\_Click**(object sender, EventArgs e) | Izpildās, kad tiek nospiesta vadīkla btnNew4pPoints. Atļauj veikt līknes <4 pPoints> mezglu punktu ievadi ar datorpeli, klaviatūru vai no .txt faila. |
| NC04\_bLS | void **btnNewLeastSquares\_Click**(object sender, EventArgs e) | Izpildās, kad tiek nospiesta vadīkla btnNewLeastSquares. Atļauj veikt līknes <Least Squares> mezglu punktu ievadi ar datorpeli, klaviatūru vai no .txt faila. |
| NC05\_bC | void **btnNewComposite\_Click**(object sender, EventArgs e) | Izpildās, kad tiek nospiesta vadīkla btnNewComposite. Atļauj veikt līknes <Composite> mezglu punktu ievadi ar datorpeli, klaviatūru vai no .txt faila. |
| NC06\_bDC | void **btnDoneComposite\_Click**(object sender, EventArgs e) | Izpildās, kad tiek nospiesta vadīkla btnDoneComposite. Atzīmē, ka pēdējai <Composite> līknei nepieciešams pievienot pēdējos kontrolpunktus. |
| NC07\_AC | void **AddcPoint**(Point mouseLocation) | Pievieno ar peli ievadīta kontrolpunkta koordinātas izvēlētai līknei. |
| NC08\_AP | void **AddpPoint**(Point mouseLocation) | Pievieno ar peli ievadīta mezgla punkta koordinātas izvēlētai līknei. |
| NC09\_PF | List<Point> **GetPointsfromFile**() | Ielasa .txt failu, pārvērš tā tekstu par punktu koordinātām un atgriež sarakstu ar šiem punktiem. |

### Līkņu konstruēšana

Līkņu konstruēšana iekļauj visas tās funkcijas, kas saistītas ar Bezjē līkņu konstruēšanu – līknes kontrolpunktu koordinātu aprēķināšanu, tai skaitā visas interpolācijai nepieciešamās funkcijas, parametrizācijas metodes, saliktu līkņu konstruēšanu. Visu funkciju uzskaite un īsi apraksti atrodami 3.3. tabulā, sarežģītākajām funkcijām pieejami arī detalizētāki apraksti.

*3.3. tabula –* **Līkņu konstruēšanas funkcijas**

|  |  |  |
| --- | --- | --- |
| **Nosaukums** | **Funkcija** | **Apraksts** |
| CC01\_CI | void **AddcPointsInterpolation**(int i) | Veic līkņu <4 pPoints> un <Least Squares> interpolāciju izmantojot izvēlētās līknes mezglu punktus. Aprēķina un saglabā interpolētās līknes kontrolpunkus. |
| CC02\_VU | List<double> **GetsValuesUniform**(List<Point> pList) | Aprēķina un atgriež sarakstu ar vērtībām mezglu punktiem izvēlētai līknei. Vērtības tiek iegūtas izmantojot vienmērīgā sadalījuma parametrizācijas metodi. Skatīt 3.1.7. nodaļu. |
| CC03\_VCh | List<double> **GetsValuesChord**(List<Point> pList) | Aprēķina un atgriež sarakstu ar vērtībām mezglu punktiem izvēlētai līknei. Vērtības tiek iegūtas izmantojot hordu garuma parametrizācijas metodi. Skatīt 3.1.7. nodaļu. |
| CC04\_VCe | List<double> **GetsValuesCentripetal**(List<Point> pList) | Aprēķina un atgriež sarakstu ar vērtībām mezglu punktiem izvēlētai līknei. Vērtības tiek iegūtas izmantojot centrtieces spēka parametrizācijas metodi. Skatīt 3.1.7. nodaļu. |
| CC05\_AS | double[,] **GetArrayS**(List<double> sValues) | Aprēķina un atgriež divdimensiju masīvu, kas reprezentē matricu . Masīvs tiek aizpildīts, izmantojot parametrizācijas rezultātā iegūtas vērtības. |
| CC06\_CC | void **AddcPointsComposite**(int i) | Aprēķina un saglabā kontrolpunktu koordinātas <Composite> līknēm ar vismaz trīs mezglu punktiem. |
| CC07\_OC | void **AddOnlycPointsComposite**(int i) | Aprēķina un saglabā kontrolpunktu koordinātas <Composite> līknēm ar diviem mezglu punktiem. |
| CC08\_VF | Point **GetVeryFirstHandle**(Point firstpPoint, Point oppositeHandle, Point secondpPoint) | Aprēķina un atgriež paša pirmā roktura koordinātas mezgla punktam, kas ir sākumpunkts un pieder <Composite> līknei ar vismaz trīs mezglu punktiem. |
| CC09\_FH | Point **GetFirstHandle**(Point prevpPoint, Point thispPoint, Point nextpPoint) | Aprēķina un atgriež pirmā roktura koordinātas mezgla punktam, kas nav galapunkts un pieder <Composite> līknei ar vismaz trīs mezglu punktiem. |
| CC10\_LH | Point **GetSecondHandle**(Point prevpPoint, Point thispPoint, Point nextpPoint) | Aprēķina un atgriež otrā roktura koordinātas mezgla punktam, kas nav galapunkts un pieder <Composite> līknei ar vismaz trīs mezglu punktiem. |
| CC11\_VL | Point **GetVeryLastHandle**(Point prevpPoint, Point prevHandle, Point lastpPoint) | Aprēķina un atgriež pirmā roktura koordinātas mezgla punktam, kas ir beigu punkts, un pieder <Composite> līknei ar vismaz trīs mezglu punktiem. |
| CC12\_L | double **GetLength**(Point firstPoint, Point secondPoint) | Aprēķina un atgriež attālumu starp diviem punktiem. |

void **AddcPointsComposite**(int i):

Funkcija aprēķina rokturus un saglabā visu kontrolpunktu koordinātas <Composite> līknēm ar trīs vai vairāk mezglu punktiem. Apzīmējam saliktās līknes mezglu punktu skaitu ar . Varam ievērot, ka:

* Katrs trešais kontrolpunkts sakrīt ar līknes -to mezgla punktu;
* Katrs kontrolpunkts formā ir pirmais rokturis, aprēķināms ar funkciju **GetFirstHandleComposite()**;
* Katrs kontrolpunkts formā ir otrais rokturis, aprēķināms ar funkciju **GetSecondHandleComposite()**;
* Kontrolpunkts ir saliktās līknes pats pirmais rokturi, aprēķināms ar funkciju **GetVeryFirstHandleComposite()**;
* Kontrolpunkts ir saliktās līknes pats pēdējais rokturis, aprēķināms ar funkciju **GetVeryLastHandleComposite()**;

void **AddOnlycPointsComposite**(int i):

Funkcija aprēķina rokturus un saglabā visu kontrolpunktu koordinātas <Composite> līknēm ar diviem mezglu punktiem.

Līknes pirmais kontrolpunkts un pēdējais kontrolpunkts sakrīt ar attiecīgi līknes pirmo mezglu un otro mezgla punktu . Rokturu veidotais leņķis ar taisni, ko veido mezglu punkti, ir , savukārt attālumi no rokturiem līdz tiem tuvākajiem mezglu punktiem vienādi ar pusi no attāluma starp mezglu punktiem. Tātad rokturu koordinātas aprēķināmas ar formulu:

Pilnu metodes skaidrojumu un pamatojumu skatīt 3.1.6. nodaļā.

Point **GetVeryFirstHandle**(Point firstpPoint, Point oppositeHandle, Point secondpPoint):

Funkcija aprēķina un atgriež pirmā roktura koordinātas mezgla punktam, kas ir sākumpunkts un pieder saliktai Bezjē līknei ar vismaz 3 mezgliem.

Funkcijai tiek padoti divi mezglu punkti sākumpunkta mezgls (firstpPoint) un līknes nākamais mezgls (secondpPoint), kā arī nākamā mezgla pirmais rokturis (nextHandle). Punkti tiek skaitīti no saliktās līknes sākumpunkta līdz galapunktam.

Aprēķinus var izteikt kā vektoru darbības. Ieviešam sekojošus apzīmējumus:

* – vektors no secondpPoint uz firstpPoint;
* – vektors no secondpPoint uz nextHandle;
* – vektors no nextHandle uz rokturi .

Roktura koordinātas aprēķināmas kā. Vektora virziens ir tāds pats kā vektoram , tā garums izsakāms kā garums mīnus divi garumi projekcijai uz :

Pilnu metodes skaidrojumu un pamatojumu skatīt 3.1.6. nodaļā.

Point **GetFirstHandle**(Point prevpPoint, Point thispPoint, Point nextpPoint):

Funkcija aprēķina un atgriež pirmā roktura koordinātas mezgla punktam, kas nav galapunkts un pieder saliktai Bezjē līknei.

Funkcijai tiek padoti trīs mezglu punkti – tas, kuram tiek rēķināts rokturis (thispPoint) un tā blakus mezgli (prevpPoint un nextpPoint). Punkti tiek skaitīti no saliktās līknes sākumpunkta līdz galapunktam.

Aprēķinus var izteikt kā vektoru darbības. Vektoram no thispPoint uz pirmo rokturi būs tāds pats virziens, kā vektoram no nextpPoint uz prevpPoint. Apzīmējam attālumu no prevpPoint līdz thispPoint ar un attālumu no thispPoint līdz nextpPoint ar . Tad vektoru var aprēķināt kā pusi no vektora reiz . Esam ieguvuši, ka pirmā roktura koordinātas izsakāmas ar formulām:

Pilnu metodes skaidrojumu un pamatojumu skatīt 3.1.6. nodaļā.

Point **GetSecondHandle**(Point prevpPoint, Point thispPoint, Point nextpPoint)

Aprēķini ir ļoti līdzīgi aprēķiniem funkcijā **GetFirstHandleComposite()**. Vektoram no thispPoint uz otro rokturi būs tāds pats virziens, kā vektoram no prevpPoint uz nextpPoint. Apzīmējam attālumu no prevpPoint līdz thispPoint ar un attālumu no thispPoint līdz nextpPoint ar . Tad vektoru var aprēķināt kā pusi no vektora reiz . Esam ieguvuši, ka pirmā roktura koordinātas izsakāmas ar formulām:

Pilnu metodes skaidrojumu un pamatojumu skatīt 3.1.6. nodaļā.

Point **GetVeryLastHandle**(Point prevpPoint, Point prevHandle, Point lastpPoint):

Aprēķini ir ļoti līdzīgi aprēķiniem funkcijā **GetVeryFirstHandleComposite()**. Ieviešam sekojošus apzīmējumus:

* – vektors no prevpPoint uz lastpPoint;
* – vektors no prevpPoint uz prevHandle;
* – vektors no prevHandle uz pēdējo rokturi .

Roktura koordinātas aprēķināmas kā. Vektora virziens ir tāds pats kā vektoram , turklāt tā garums izsakāms kā garums mīnus divi garumi projekcijai uz :

Pilnu metodes skaidrojumu un pamatojumu skatīt 3.1.6. nodaļā.

### Līkņu modificēšana

Līkņu modificēšana iekļauj visas tās funkcijas, kas saistītas ar jau konstruētu Bezjē līkņu modificēšanu, tai skaitā parametrizācijas metodes maiņu, kontrolpunktu un mezglu punktu atrašanās vietu maiņu. Visu funkciju uzskaite un īsi apraksti atrodami 3.4. tabulā, sarežģītākajām funkcijām pieejami arī detalizētāki apraksti.

*3.4. tabula –* **Līkņu modificēšanas funkcijas**

|  |  |  |
| --- | --- | --- |
| **Nosaukums** | **Funkcija** | **Apraksts** |
| MC01\_bCP | void **btnChangeParam\_Click**(object sender, EventArgs e) | Atļauj izvēlēties līkni, kurai veikt līkņu parametrizācijas metodes maiņu. Izpildās, kad tiek nospiesta atbilstošā vadīkla. |
| MC02\_CP | void **ChangeParametrization**() | Norāda izvēlētās līknes parametrizācijas metodi un atļauj veikt līkņu <4 pPoints>, <Least Squares> parametrizācijas metodes maiņu. |
| MC03\_rUC | void **rbUniform\_CheckedChanged**(object sender, EventArgs e) | Veic līkņu <4 pPoints> un <Least Squares> parametrizācijas metodes maiņu uz/no vienmērīga sadalījuma metodes. Izpildās, kad tiek mainīts atbilstošās radio pogas statuss. |
| MC04\_rCC | void **rbChord\_CheckedChanged**(object sender, EventArgs e) | Veic līkņu <4 pPoints> un <Least Squares> parametrizācijas metodes maiņu uz/no hordu garuma metodes. Izpildās, kad tiek mainīts atbilstošās radio pogas statuss. |
| MC05\_bMC | void **btnModifycPoints\_Click**(object sender, EventArgs e) | Atļauj veikt kontrolpunktu koordinātu modificēšanu. Izpildās, kad tiek nospiesta atbilstošā vadīkla. |
| MC06\_bMP | void **btnModifypPoints\_Click**(object sender, EventArgs e) | Atļauj veikt mezglu punktu koordinātu modificēšanu. Izpildās, kad tiek nospiesta atbilstošā vadīkla. |
| MC07\_MC | void **ModifycPoint**(MouseEventArgs e) | Pārbauda, vai izvēlētais kontrolpunkts var tikt modificēts, nosaka tā modificēšanas veidu. |
| MC08\_MP | void **ModifypPoint**() | Pārbauda, vai izvēlētais mezgla punkts var tikt modificēts, nosaka tā modificēšanas veidu. |
| MC09\_MH | void **ModifyHandleComposite**(Point modifyHandle, Point middlepPoint, Point oppositeHandle, int opposite) | Saliktas līknes roktura patvaļīgas modificēšanas gadījumā aprēķina un saglabā pretējā roktura koordinātas. |
| MC10\_MHS | void **ModifyHandleCompositeStraight**(Point modifyHandle, Point middlepPoint, Point oppositeHandle) | Saliktas līknes roktura ierobežotas modificēšanas gadījumā aprēķina un saglabā roktura koordinātas. |
| MC11\_MPC | void **ModifypPointComposite**(Point mouseLocation) | Saliktas līknes mezgla punkta modificēšanas gadījumā aprēķina blakus rokturu koordinātas un saglabā visu modificēto punktu koordinātas. |

void **ModifyHandleComposite**(Point modifyHandle, Point middlepPoint, Point oppositeHandle, int opposite):

Patvaļīgi modificējot (izmantojot datorpeles kreiso taustiņu) saliktas līknes roktura modifyHandle koordinātas, nepieciešams mainīt arī pretējā roktura oppositeHandle koordinātas, lai katrs mezgls un tā blakus rokturi būtu uz vienas taisnes un tādējādi tiktu nodrošināts, ka līknei piemīt nepārtrauktība. Tās tiek aprēķinātas tā, lai starp rokturiem esošais mezgla punkts middlepPoint pieder taisnei, ko veido minētie rokturi, un attālums no middlepPoint līdz oppositeHandle ir nemainīgs.

Aprēķinus var izteikt kā vektoru darbības. Ieviešam apzīmējumus:

* – vektors no modifyHandle uz middlepPoint;
* – vektors no middlepPoint uz oppositeHandle;
* – vektors no middlepPoint uz jaunajām roktura koordinātām .

Vektoram jābūt tādam pašam virzienam kā un garumam jābūt vienādam ar . Varam izmantot formulu:

Pilnu metodes skaidrojumu un pamatojumu skatīt 3.1.8. nodaļā.

void **ModifyHandleCompositeStraight**(Point modifyHandle, Point middlepPoint, Point oppositeHandle):

Ierobežoti modificējot (izmantojot datorpeles labo taustiņu) saliktas līknes roktura modifyHandle koordinātas, minēto rokturi iespējams pārvietot tikai pa staru, kas pieder taisnei, ko veido pretējais rokturis oppositeHandle un mezgla punkts middlepPoint starp minētajiem rokturiem. Stara sākumpunkts ir middlepPoint.

Ērts veids, kā to implementēt, ir izmantojot vektoru darbības. Vektoram no middlepPoint uz modifyHandle jābūt tādam pašam virzienam, kā vektoram no oppositeHandle uz middlepPoint, savukārt attālumam no middlepPoint līdz jaunajām roktura koordinātām jāsakrīt ar . Varam izmantot formulu:

Implementācijā jāpievērš uzmanība maksimālajam attālumam starp datorpeles kursora atrašānās vietu un mezgla koordinātām.

Pilnu metodes skaidrojumu un pamatojumu skatīt 3.1.8. nodaļā.

void **ModifypPointComposite**(Point mouseLocation):

Funkcija aprēķina un saglabā saliktas līknes mezgla punkta un tā blakus rokturu jaunās koordinātas, kad mezgla punkts tiek modificēts. Lai līknei arī pēc modificēšanas piemistu nepārtrauktība un pārējie līknes segmenti modifikācijas rezultātā nemainītos, ērtākais veids ir nemainīt un relatīvo atrašanās vietu pret .

Aprēķinus var interpretēt kā vektoru darbības. Jaunās rokturu koordinātas var aprēķināt izmantojot formulu:

Pilnu metodes skaidrojumu un pamatojumu skatīt 3.1.8. nodaļā.

### Esošu līkņu funkcijas

Esošu līkņu funkcijas iekļauj visas tās funkcijas, kas saistītas ar jau konstruētām Bezjē līknēm – līkņu kontrolpunktu vai mezglu punktu izvadi un līkņu dzēšanu, bet neiekļauj funkcijas, kas saistītas ar līkņu modificēšanu. Visu funkciju uzskaite un īsi apraksti atrodami 3.5. tabulā.

*3.5. tabula –* **Esošu līkņu funkcijas**

|  |  |  |
| --- | --- | --- |
| **Nosaukums** | **Funkcija** | **Apraksts** |
| EC01\_LP | void **FindLocalPoint**(List<List<Point>> PointsAll, Point MouseLocation) | Atrod, vai peles kursora apkārtnē ir kāds kontrolpunkts vai mezgla punkts. |
| EC02\_bOC | void **btnOutputcPoints\_Click**(object sender, EventArgs e) | Atļauj izvēlēties līkni, kurai izvadīt līknes kontrolpunktu koordinātas. |
| EC03\_bOP | void **btnOutputpPoints\_Click**(object sender, EventArgs e) | Atļauj izvēlēties līkni, kurai izvadīt līknes mezglu punktu koordinātas. |
| EC04\_OCF | void **OutputcPointsToFile**() | Izvada izvēlētas līknes kontrolpunktu koordinātas .txt failā. |
| EC05\_OPF | void **OutputpPointsToFile**() | Izvada izvēlētas līknes mezglu punktu koordinātas .txt failā. |
| EC06\_bDL | void **btnDeleteLine\_Click**(object sender, EventArgs e) | Izpildās, kad tiek nospiesta vadīkla btnDeleteLine. Atļauj izvēlēties līkni, kuru izdzēst. |
| EC07\_DL | void **DeleteLine**(int i) | Izdzēš izvēlētu līkni. |

### Koordinātu logs

Koordinātu logs iekļauj visas tās funkcijas, kas izpildās un ir piesaistītas koordinātu logam – tā inicializācijas veidus, punktu koordinātu ievadi, izvadi un pievienošanu. Visu funkciju uzskaite un īsi apraksti atrodami 3.5. tabulā.

*3.5. tabula –* **Koordinātu loga funkcijas**

|  |  |  |
| --- | --- | --- |
| **Nosaukums** | **Funkcija** | **Apraksts** |
| FC01\_IA | void **InitializeAdd**() | Inicializē formu, ja tā izsaukta jaunas līknes pievienošanai izmantojot datora tastatūru. |
| FC02\_IM | void **InitializeModify**() | Inicializē formu, ja tā izsaukta esošas līknes modificēšanai izmantojot datora tastatūru. |
| FC03\_IO | void **InitializeOutput**() | Inicializē formu, ja tā izsaukta esošas līknes punktu izvadīšanai uz ekrāna. |
| FC04\_AR | void **AddRow**() | Pievieno jaunu rindu un aizpilda to ar nepieciešamajām vadīklām, lai ievadītu vai izvadītu punktu koordinātas. |
| FC05\_bAR | void **btnAddRow\_Click**(object sender, EventArgs e) | Izpildās, kad tiek nospiesta vadīkla btnAddRow. Pievieno jaunu rindu pēc lietotāja vaicājuma. |
| FC06\_bDR | void **btnDeleteRow\_Click**(object sender, EventArgs e) | Izpildās, kad tiek nospiesta vadīkla btnDeleteRow. Pievieno jaunu rindu pēc lietotāja vaicājuma |
| FC07\_bSI | void **btnSubmitInput\_Click**(object sender, EventArgs e) | Izpildās, kad tiek nospiesta vadīkla btnSubmitInput. Pārbauda, vai visas vadīklas koordinātu ievadei ir aizpildītas, pārvērš ievadīto tekstu punktos un tos saglabā. |
| FC08\_bRI | void **btnResetInput\_Click**(object sender, EventArgs e) | Izpildās, kad tiek nospiesta vadīkla btnResetInput. Nodzēš tekstu visās vadīklās, kas paradzētas punktu koordinātu ievadei. |

# TESTĒŠANAS DOKUMENTĀCIJA

## Testpiemēru projektējums

Kkads ievads i guess

*4.1. tabula –* **Testpiemēri**

|  |  |  |
| --- | --- | --- |
| **Identifikators** | **Darbības** | **Sagaidāmais rezultāts** |
|  | Līknes <4 cPoints> kontrolpunktu izvēle un modificēšana ar datorpeli, kontrolpunktu izvade uz ekrāna. |  |
|  | Līknes <4 cPoints> kontrolpunktu izvēle un modificēšana ar datora tastatūru, kontrolpunktu izvade .txt failā, līknes dzēšana. |  |
|  | Līknes <4 pPoints> mezglu punktu ielasīšana no pareiza .txt faila, izmantojot hordu garuma parametrizācijas metodi, mezglu punktu modificēšana ar datorpeli, mezglu punktu izvade uz ekrāna. |  |
|  | Līknes <4 pPoints> mezglu punktu ielasīšana no nepareiza .txt faila. |  |
|  | Līknes <4 pPoints> mezglu punktu izvēle ar peli, izmantojot centrtieces spēka parametrizācijas metodi, kontrolpunktu modificēšana ar datorpeli, parametrizācijas metodes maiņa. |  |
|  | Līknes <Least Squares> sešu mezglu punktu izvēle ar datora tastatūru, pirms tam punktu skaitu palielinot līdz 20 un samazinot līdz 2. Līknes kontrolpunktu un mezglu punktu modificēšana izmantojot peli. |  |
|  | Līknes <Composite> sešu mezglu punktu izvēle ar peli, pirms tam punktu skaitu samazinot līdz 1. Līknes kontrolpunktu modificēšana izmantojot peles labo un kreiso taustiņu un mezglu punktu modificēšana izmantojot peli. |  |
|  | Līknes <Composite> divu mezglu punktu ievade no .txt faila. |  |
|  | Visu četru līkņu veidu mezglu punktu modificēšana izmantojot datora tastatūru. |  |
|  | Visu četru līkņu veidu kontrolpunktu izvade uz ekrāna. |  |
|  | Visu četru līkņu veidu mezglu izvade uz ekrāna. |  |

## Testēšanas žurnāls

Kkads ievads i guess

*4.2. tabula –* **Testēšanas žurnāls**

|  |  |  |
| --- | --- | --- |
| **Identifikators** | **Rezultāts** | **Datums** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |

# PROJEKTA ORGANIZĀCIJA

Projekta izstrādi veica viens cilvēks un tika pielietots pakāpeniskā dzīves cikla modeļa paveids. Projekta izstrādi var iedalīt trīs daļās:

1. prasību specifikācija un matemātiskais projektējums,
2. atsevišķu programmatūru izstrāde dažādām funkcionalitātēm,
3. gala produkta izstrāde, testēšana un dokumentēšana.

Pirmajā izstrādes daļā tika veikta visu prasību specifikācija un lielākā daļa programmatūras matemātiskā projektējuma (3.1. nodaļa). Otrajai izstrādes daļai bija seši soļi, kuros tika izstrādātas un daļēji testētas sešas atsevišķas, vienkāršas programmatūras (4points, caur\_4, type\_coord, leastSquares, compositeBezier un 2bezje), lai pārbaudītu, vai iecerētās metodes var apmierināt funkcionālās prasības. Atšķirībā no klasiskā pakāpeniskā dzīves cikla modeļa, katra soļa izstrādātie modeļi uzreiz netika pievienoti kopējai sistēmai. Izstrādes trešajā daļā tika izveidots vispārējs arhitektūras projektējums, kopējā sistēma un tai pievienoti iepriekš izstrādātie modeļi, kā arī pievienotas jaunas iepriekš neizstrādātas funkcionalitātes. Bez tā, trešajā izstrādes daļā notika gan vienībtestēšana, gan kopēja sistēmas testēšana, kā arī dokumentācijas izveide. Programmatūras projektēšana mazākos apmēros notika arī otrajā un trešajā izstrādes daļā.

## Darbietilpības novērojums

### Prognozētā darbietilpība

Lai prognozētu projekta darbietilpību, ērti sadalīt visus izstrādes procesus vairākās daļās un katras daļas iecerēto ilgumu dienās novērtēt ar trīs punktu metodi (pesimistisko, reālistisko un optimistisko), skatīt 5.1. tabulu. Prognozētais ilgums tiek aprēķināts ar formulu .

*5.1. tabula –* **Prognozētā darbietilpība**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Izstrādes process** | **Pesimistiskais novērtējums (dienās)** | **Reālistiskais novērtējums (dienās)** | **Optimistiskais**  **novērtējums (dienās)** | **Prognozētais ilgums**  **(dienās)** |
| Prasību specificēšana | 4 | 3 | 2 | 3 |
| Matemātiskais projektējums | 60 | 45 | 40 | 46.7 |
| Metodes 4points izstrāde | 4 | 3 | 2 | 3 |
| Modeļa caur\_4 izstrāde | 3 | 2 | 1 | 2 |
| Modeļa type\_coord izstrāde | 3 | 2 | 1 | 2 |
| Modeļa leastSquares izstrāde | 2 | 1 | 1 | 1.2 |
| Modeļa compositeBezier izstrāde | 4 | 2 | 1 | 2.2 |
| Modeļa 2bezje izstrāde | 2 | 1 | 1 | 1.2 |
| Sistēmas arhitektūras un funkciju projektējums | 6 | 4 | 2 | 4 |
| Izveidoto modeļu integrācija | 7 | 4 | 2 | 4.2 |
| Sistēmas  izstrāde | 20 | 17 | 12 | 16.7 |
| Sistēmas testēšana | 5 | 3 | 2 | 3.2 |
| Dokumentācijas izstrāde | 10 | 7 | 5 | 7.2 |
| Kopā | 130 | 94 | 72 | 96.3 |

Kā redzams 5.1. tabulā, sagaidāmais izstrādes ilgums ir 96.3 dienas jeb aptuveni 4.4 personmēneši, pieņemot, ka vienā mēnesī vidēji ir 21.74 darba dienas.

### Reālā darbietilpība

Pēc programmatūras izstrādes bija iespējams novērtēt reālo projekta darbietilpību. Izveidotais rīks sastāv no divām saskarnēm un ļauj konstruēt līknes četros veidos, papildus izvēloties no trīs parametrizācijas metodēm un trīs punktu ievades veidiem. Jau konstruētas līknes iespējams modificēt, mainot to parametrizācijas metodi, kā arī mainot to kontrolpunktu un/vai mezglu punktu koordinātas ar datorpeli vai ievadot ar datora tastatūru. Konstruētās līknes iespējams arī dzēst, izvadīt to kontrolpunktu un/vai mezglu punktu koordinātas – uz ekrāna vai .txt failā. Tiek piedāvāta iespēja arī pievienot fona attēlu un mainīt tā redzamību. Rīkam kopumā saskaitāmi 23 funkcijpunkti jeb 3.2 personmēneši [7].

Pirms rīka programmēšanas, bija nepieciešams veikt padziļinātu izpēti par parametriskām, tai skaitā Bezjē, līknēm, kas prasīja vairāk nekā divus personmēnešus. Pēc rīka izstrādes bija nepieciešams izveidot programmatūras dokumentāciju.

Programmatūras funkcionālo rindiņu skaits ir 1391, un, ņemot vērā, ka izstrādātais rīks ir biznesa sistēma ar zinātnisku ievirzi, tā darbietilpība ir 3.2-7.2 personmēneši [7].

Kopumā projekta izstrāde prasīja 5.2 personmēnešus.

## Kvalitātes nodrošināšana

Lai nodrošinātu kvalifikācijas darba kvalitāti, tā izstrādē tika ievērotas vairākas darbības.

Darba dokumentācija tika izstrādāta, balstoties uz standartiem LVS 68:1996 «Programmatūras prasību specifikācijas ceļvedis», LVS 72:1996 «Ieteicamā prakse programmatūras projektējuma aprakstīšanai», LVS 66:1996 «Programmatūras lietotāja dokumentācija» un LVS 70:1996 «Programmatūras testēšanas dokumentācija».

Sistēmas pirmkoda izstrādē ir ievērots Microsoft .NET programmēšanas stila standarts. Pirmkods rakstīts veidā, kas iespējami palielina tā lasāmību, pirmkodā pēc vajadzības ievietoti komentāri, kas apraksta tā darbību. Identifikatori veidoti saturīgi, funkcijas sakārtotas secībā, kas iespējami sakrīt ar to izmantošanas secību.

Visām kvalifikācijas darba funkcijām veikta vienībtestēšana, kā arī veikta sistēmas kopējās darbības testēšana un atkļūdošana.

## Konfigurāciju pārvaldība

Kvalifikācijas darba izstrādei tika izmantotas versiju kontroles sistēmas. Pirmkoda izstrādes

versiju kontrole pie lielākām izmaiņām un darba dienas beigās tika veikta Git sistēmā. Sistēmas dokumentācijas versiju kontrolei tika izmantots Microsoft Office 365 tiešsaistes dokumentu veidošanas rīks [13], pie apjomīgākām izmaiņām dokumentācijas versiju kontrole tika uzturēta arī Git sistēmā. Izstrādes versiju kontroles sistēmas tika izmantotas, lai glabātu un nepieciešamības gadījumā spētu atgriezt veiktās izmaiņas, piemēram, ja pašreizējā sistēmas versijā radusies kļūda.

# REZULTĀTI UN SECINĀJUMI

Kvalifikācijas darba izstrādes rezultātā tika iegūts funkcionējošs rīks, kas ļauj attēlot četros dažādos veidos konstruētas Bezjē līknes, papildus izmantojot trīs dažādas parametrizācijas metodes. Konstruētās līknes iespējams modificēt, dzēst un izvadīt to kontrolpunktus un/vai mezglu punktus. Līkņu punktu ievade, izvade un modificēšana veicama izmantojot datorpeli, datora tastatūru un teksta failus. Tika veikta arī rīka testēšana un izstrādāta tā dokumentācija.

Izstrādes laikā autore ieguva vērtīgu pieredzi visos programmatūras izstrādes procesos un veiksmīgi apguva jaunus programmēšanas rīkus – Microsoft Visual Studio vidi, C# valodu, Microsoft .NET satvaru, bibliotēku Windows Forms un Git versiju kontroles sistēmu. Bez tā autore apguva parametrisku līkņu teoriju, pastiprināti Bezjē līknes, to interpolāciju un parametrizāciju.

Pašlaik pasūtītājs sistēmu lieto ražošanā, turklāt ir plānots turpināt tās izstrādi, pievienojot tai papildu funkcionalitātes, piemēram, līknes garuma noteikšanu, grafisko objektu pietuvināšanu un mēroga iestatīšanu u. c.

Izvērtējot padarīto, autore secina, ka rīka implementācijā būtu ērtāk izmantot objektorientētās programmēšanas metodes, it īpaši Bezjē līkņu un citu grafisko objektu implementāciju kā klases.

Pabeidzot kvalifikācijas darbu, autore un pasūtītājs atkārtoti pārliecinājās, ka Bezjē līknes ir piemērotas izvēlētajai sistēmai. Veicot matemātisko projektējumu, tika izvēlēti četri Bezjē līkņu konstruēšanas veidi, trīs parametrizācijas metodes un dažādi līkņu modificēšanas veidi. Autore secina, ka apvienojot šīs metodes, tiek nosegtas visas līkņu konstruēšanas prasības, turklāt tas tiek izdarīts intuitīvi un patērējot salīdzinoši maz skaitļošanas un atmiņas resursu.

# PATEICĪBAS

Autore izsaka pateicību profesorei Dr. mat. Inesei Bulai par ieteikumiem kvalifikācijas darba matemātiskajā daļā un aprēķinu pārskatīšanu.

Sirsnīgs paldies arī draudzenei un kolēģei Ievai Pakalnai par atbalstu, ieteikumiem un uzklausīšanu visos darba izstrādes procesos.

# IZMANTOTĀ LITERATŪRA UN AVOTI

lietas no seienes: <https://pages.mtu.edu/~shene/COURSES/cs3621/NOTES/>

<https://pages.mtu.edu/~shene/COURSES/cs3621/NOTES/spline/Bezier/intro.html>

[1] 68:1996 pps

[2] 72:1996 ppa

[3] <https://en.wikipedia.org/wiki/B%C3%A9zier_curve>

[4] <https://www.gimp.org/tutorials/Bezier_Selections/>

[5] <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4712932/>

[6] <https://en.wikipedia.org/wiki/Non-uniform_rational_B-spline>

[7] tas word borzova funkcijpunktu raksts

[8] <https://pages.mtu.edu/~shene/COURSES/cs3621/NOTES/INT-APP/PARA-uniform.html>

[9] <https://pages.mtu.edu/~shene/COURSES/cs3621/NOTES/INT-APP/PARA-chord-length.html>

[10] <https://pages.mtu.edu/~shene/COURSES/cs3621/NOTES/INT-APP/PARA-centripetal.html>

[11] <https://apps.dtic.mil/dtic/tr/fulltext/u2/a350611.pdf>

[12] <https://pomax.github.io/bezierinfo/>

myb tā viena grāmata? !!!!!!!

<https://en.wikipedia.org/wiki/B-spline>

# PIELIKUMI

# Programmatūras koda fragments

# Mazāko kvadrātu metodes pierādījums Bezjē līknes interpolācijai

**A screenshot of a cell phone

Description automatically generated**

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

## Skalārā reizinājuma atvasinājums

**A close up of a logo

Description automatically generated**

Kvalifikācijas darbs „***Rīks Bezjē līkņu konstruēšanai un modificēšanai***” izstrādāts Latvijas Universitātes Datorikas fakultātē.

Ar savu parakstu apliecinu, ka darbs izstrādāts patstāvīgi, izmantoti tikai tajā norādītie informācijas avoti un iesniegtā darba elektroniskā kopija atbilst izdrukai.

Autors: ***Elīza Gaile*** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ **\_\_\_.05.2019**.

Rekomendēju darbu aizstāvēšanai

Darba vadītājs: ***doc., Dr. dat. Vineta Arnicāne*** \_\_\_\_\_\_\_\_\_\_\_\_\_ **\_\_\_.05.2019.**

Recenzents: ***Dr. sc. ing. Edžus Žeiris***

Darbs iesniegts 27.05.2019.

Kvalifikācijas darbu pārbaudījumu komisijas sekretāre: *Darja Solodovņikova*\_\_\_\_\_\_\_

Darbs aizstāvēts kvalifikācijas darbu pārbaudījuma komisijas sēdē

\_\_\_.06.2019. prot. Nr. \_\_\_\_\_\_\_\_\_\_

Komisijas sekretārs(-e): \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_