TARTU ÜLIKOOL
Arvutiteaduse instituut
informaatika õppekava

Argo Neumann

Miinipildujapatarei arvutusprogramm

Bakalaureusetöö (9 EAP)

Juhendaja: professor Eero Vainikko

Tartu 2017

Miinipildujapatarei arvutusprogramm

Lühikokkuvõte:

Antud töö eesmärgiks on luua miinipildujapatarei arvutuskeskusele universaalne arvutusprogramm, mis oleks kasutatav erinevate kaliibrite ja moonadega ning uuendatavate lasketabelitega. Töö loomiseks kasutatakse Java 8 koos JavaFX teegiga. Programm arvutab etteantud positsioonide ja sihtmärkide vahelised laskeandmed, kasutades selleks sisseloetud lasketabeleid. Programmi arvutused toetuvad militaarkeskkonnas kasutatavale MGRS koordinaatsüsteemile ning nurkade leidmiseks kasutatavale tuhandiksüsteemile. Laskeandmeid saab mõjutada nii temperatuuri muutmisega, sihtmärgile korrektuure tehes või positsioonide ja asukohtade koordinaate muutes. Positsioonide ning sihtmärkide omavaheline asetus on realiseeritud liigutatava graafikaga.

Võtmesõnad:

Geomeetria, andmetöötlus, JavaFX, miinipildujapatarei

CERCS: P150 Geomeetria, algebraline topoloogia; P170 Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine (automaatjuhtimisteooria)

Calculator program for mortar battery

Abstract:

The goal of this thesis is to develop universal calculator program for mortar battery´s fire direction center which would be usable with different types and calibers of ammos and has updateable shooting tables. Following technologies were used for development: Java 8 and JavaFX. Program calculates firing data between multiple positions and targets by using pre-loaded mortar ammunition firing tables. Programs calculations are based on MGRS grid coordinates and NATO mils. Firing data can be affected by temperature, corrections or changes in coordinates of targets and positions. Disposition between targets and positions is implemented by draggable and zoomable graphical grid.

Keywords:

Geometry, data processing, JavaFX, mortar battery

CERCS: P150 Geometry, algebraic topology; P170 Computer science, numerical analysis, systems, control

Sisukord

1. Sissejuhatus

Käesolev teema sai valitud seoses ajateenistuse läbimisega 2016/2017. aastal, mil töö autor teenis aega Kuperjanovi jalaväepataljonis miinipildujapatareis ja nägi, et senine arvutuslaudadel laskeandmete arvutamine on käsimeetodil aeglane ja mitte nii täpne kui oleks samade meetodite rakendamine arvutusprogrammis. Mõne millimeetri arvutuslaual vale näidu lugemist arvutaja poolt võib tähendada mõnekümne meetriga eksimist maastikul. Selle probleemi lahendamiseks ning laskeandmete kontrollimiseks loodi aastaid tagasi arvutusprogramm, mis kasutas vanasid lasketabeleid. Sellel programmil aga puudusid mõned vajalikud funktsioonid ning sisaldas mõningaid programmeerimisvigasid. Samuti tuli alates käesolevast aastast kasutusele uus laskemoon, mille lasketabelid ei vastanud vanas programmis sees olnud andmetele ja seega tekkis vajadus uue universaalse programmi järgi, milles saaks ilma lähtekoodi muutmata uuendada lasketabeleid ja kasutada erineva kaliibriga moona. Seega oleks vaja programmi, milles saab lihtsalt ja lähtekoodi muutmata uuendada lasketabeleid ning kasutada erineva kaliibriga moona.

Selle töö eesmärgiks on luua kaudtulerelvade laskeandmete arvutamiseks vajalik arvutusprogramm, mis oleks kiirem ja täpsem kui käsitsi arvutamine. Antud tarkvara annab võimaluse ajateenijate väljaõppe kontrollimiseks, suurendades seeläbi ajateenijatest arvutajate kindlust, et nende poolt tehtav töö on korrektne ning arvutused õiged. Samuti annab programm graafilise visualiseeringuga kasutajale tagasisidet, et kas arvutuslaual olev joonestus on programmi graafikaga vastavuses. Tööga on kaasas töötav Java rakendus, lähtekood ning dokumentatsioon koos selgitavate kommentaaridega programmi arhitektuurist.

Järgnevas peatükis selgitatakse mõisteid ja termineid, mida antud lõputöös kasutatakse. Töö kolmandas peatükis antakse ülevaade kaudtulesüsteemi ajaloost ning sellest, mida kaudtulesüsteem endast kujutab ja millistest komponentidest koosneb. Samuti kirjeldatakse käsitsi laskeandmete arvutamise protseduure. Neljas peatükk on pühendatud loodud tarkvarale. Kirjeldatakse tarkvara arhitektuuri, kasutatud tehnoloogiad, tähtsamaid funktsioone ja programmi graafikat. Viiendas peatükis analüüsitakse loodud lahendust ja tuuakse välja milliseid tulemusi loodud tarkvara on saavutanud. Samuti on viiendas peatükis toodud välja installeerimis-ja kasutusjuhend.

2. Mõisted ja terminid

Tabel 1. Mõisted

Mõõdikupunkt	Positsiooni keskpunkt 10 meetrilise täpsusega
Topograafiline sihtmärk	Sihtmärgi asukoht 10 meetrilise täpsusega
Ballistiline sihtmärk	Peale korrektuure saadav sihtmärk erinevatele moonatüüpidele
Kaliiber	Miinipildujamiini läbimõõt

Tabel 2. Lühendid

MGRS	Military Grid Reference System
NATO	North Atlantic Treaty Organisation

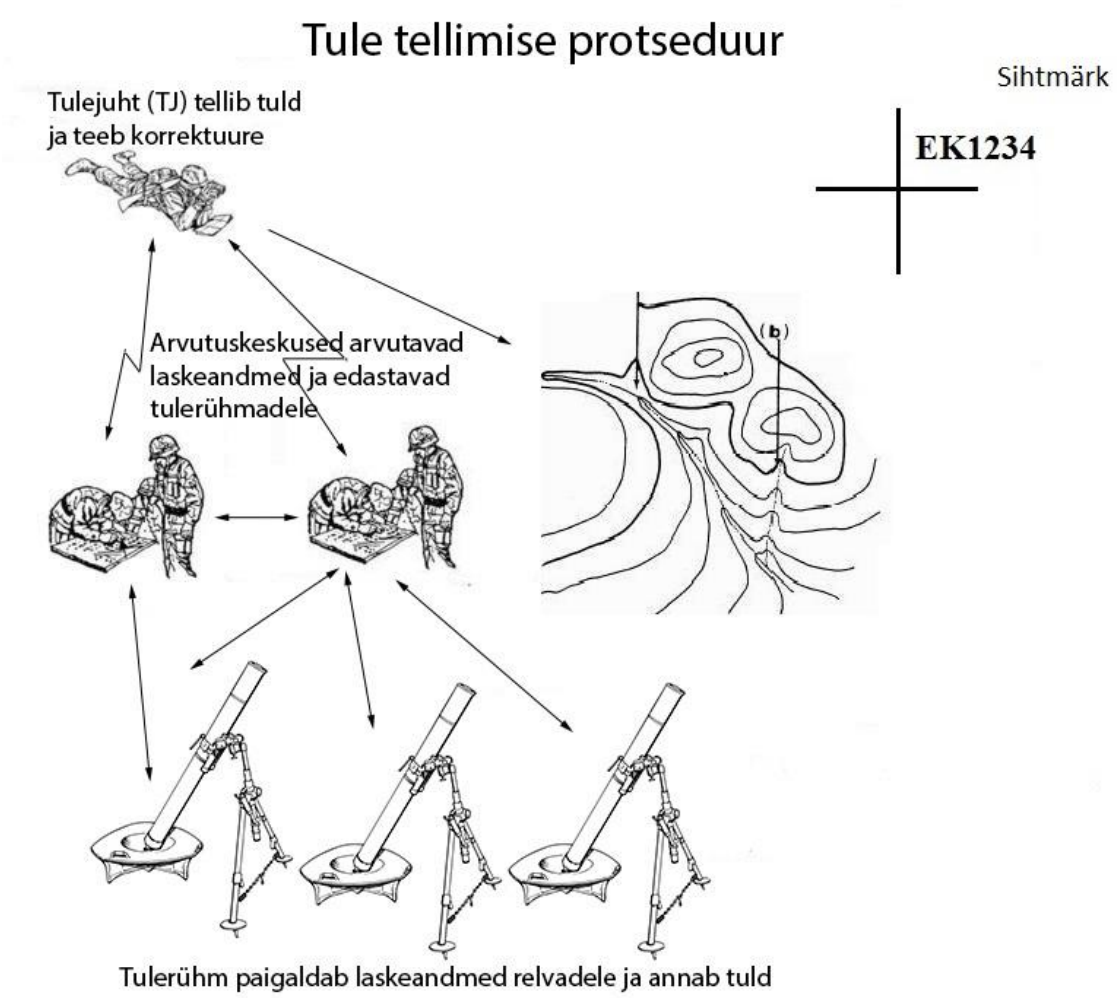
3. Kaudtulesüsteemi ajalugu ja arvutuskeskuse töö miinipildujapatereis

3.1 Kaudtulesüsteemi ajalugu

Tänapäevase kaudtulesüsteemi alguseks võib ilmselt pidada 19.sajandi teist poolt, mil Venemaal kirjeldati mitte-otsest meetodit vaenlase pihta laskmiseks, kasutades geomeetriat ning erinevaid tõstenurki kaugemale laskmiseks [1]. Puudu oli vaid seade, millega oleks saanud relvasid täpselt suunata. Puuduva sihiku leiutasid sakslased 1890-ndatel. Järgneva sajandi jooksul arendati välja erinevaid sihikuid alates klinomeetrist kuni täpsete güroskoopideni, mis tegid sihtimise aina täpsemaks ja vähem aeganõudvamaks.

Kaudtulerelvasid on erinevate kaliibrite ning moonadega. Alates nii 81mm miinipildujast, 155mm haubitsast kuni 240mm liikuvsuurtükkideni. Moon võib olla nii tavaline kildmiin, valgusmiin, suitsumiin, kassettmiin jne.

Kaudtulesüsteem (vt Joonis 1) koosneb üldjuhul kolmest komponendist: „silmadest“ ehk tulejuhist(TJ, ing. k *Forward Observer*, lüh. FO), „ajust“ ehk arvutuskeskusest(FDC, ing. k *Fire Direction Center*) ja „musklist“ ehk relvadest.



Joonis 1. Tuletellimuse protsess ja kaudtulesüsteemi üldised komponendid

Relvade suunamiseks kasutatakse peamiselt kahte meetodit. Mõlema meetodi edastab arvutajatele tulejuht. Esimene võimalus on polaarmeetod, kus tulejuht edastab enda asukoha, suuna ja kauguse sihtmärki. Teine võimalus on koordinaatmeetod, kus edastatakse ainult sihtmärgi koordinaadid MGRS-süsteemis. Olemas on ka kolmas võimalus ehk tuleülekanne. Selle meetodi puhul edastatakse mingi koordinaat, suund ning korrektuur sellest koordinaadist. Nende andmete põhjal arvutavad arvutajad välja suunamiseks vajalikud laskeandmed – suund, kaugus, tõstenurk, lennuaeg ning laeng.

3.2 Miinipildujapatarei arvutuskeskus

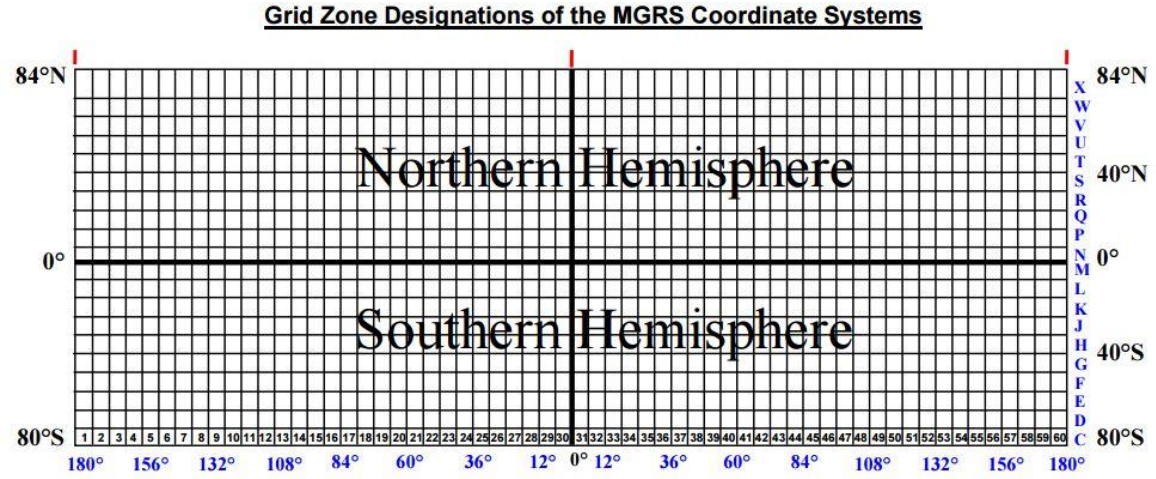
Arvutamist arvutuskeskuses tehakse alati kahe arvutuslauaga (vt joonis 5), eesmärgiga välistada tekkida võivad vead. Samal põhimõttel peaks teostama arvutamist ka arvutusprogrammiga. Et kas tehakse paralleelselt arvutused kahe arvutiga või ühe arvuti ja ühe arvutuslauaga. Järgnevalt kirjeldatakse arvutamisel kasutatavat MGRS-süsteemi ja tuhandiksüsteemi. Samuti antakse ülevaade erinevatest tuletellimuste formaatidest ning arvutuslaua komponentidest ja tingmärkidest.

MGRS koordinaatsüsteem

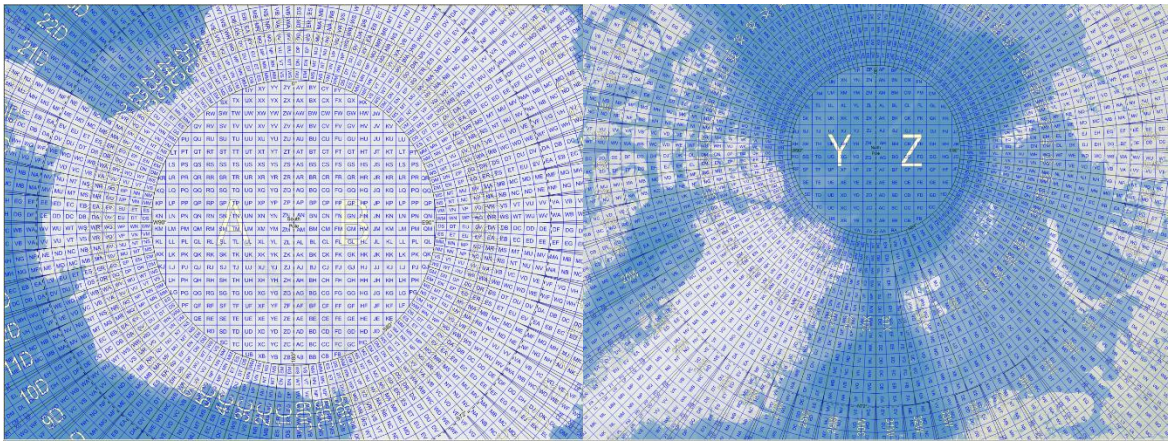
MGRS [2] (ing. k Military Grid Reference System) süsteemi alguseks peetakse 1940-ndaid, mil Ameerika sõjaväel tekkis vajadus lihtsa ja üheselt mõistetava asukohasüsteemi järgi, seejuures kõige suurem vajadus selle süsteemi järgi oli just suurtükiväelastel [3]. 1948. aasta 27. veebruaril kirjeldas uue süsteemi eest vastutava komitee esimees kolonel W.H.Mills´i, milline peaks uus süsteem olema ja mis nõudeid see peab täitma [4].

Uus süsteem pidi sobima nii mereväele, jalaväele kui ka õhuväele ja olema sobilik nii väikestele ülesannetele kui suurt maa-ala hõlmavatele operatsioonidele ning seda ka polaaraladel. See pidi olema lühike ja lihtsalt kasutatav kauguste ning suundade arvutamiseks ning välistama, et kaks osapoolt saaksid antud koordinaadist kaheti aru. Samuti pidi uus süsteem välistama märgimuutused seoses algmeridiaani ja 180° meridiaani ületamisel ning sama ka ekvatoriaaljoone ületusel [4].

Loodud süsteem [5] jagab maailma pikkuselt 60-ks tsooniks, kus iga tsooni pikkus on 6° ja laiuselt 20-ks tsooniks, kus iga tsooni laius on 8°, v.a. kõige põhjapoolsem tsoon mille laius on 12°. Kusjuures iga tekkiv ruut 6x8 ristkülikust omab suurust 100 000 km^2. Loodud võrgustik nummerdatakse läänest alates 0-st kuni 60-ni ning alates 80. lõunalaiusest tähega „C“ kuni 84. põhjalaiuseni tähega „X“, kusjuures vahele jäävad tähed „I“ ja „O“, sest neid võib segi ajada numbritega 1 ja 0. Lõunapooluse ning põhjapooluse jaoks on loodud eraldi jaotus (vt Joonis 3). Loodud võrgustikku kujutab joonis 2.



Joonis 2. MGRS võrgustik



Joonis 3. Lõunapooluse (vasakul) ja põhjapooluse jaotus [6].

Nii näiteks võtab enamuse Eestist enda alla ristkülik 35V, ning seda nimetatakse tsooni identifikaatoriks. Edasi tähistatakse saadud ristkülikus olevad ruudud kahetäheliselt, esimene täht alates läänest tähega „A“ kuni täheni „X“ ja teine täht alates lõunast „C“ laiuselt samamoodi. Ka siin jäävad vahele tähed „I“ ja „O“. Saadud kahetähelist kombinatsiooni nimetatakse 100km võrgutunnuseks. Nii näiteks asub Tartu ruudus 35VME. Edasi jagunevad ruudud omakorda võrgustikeks kuni 1m täpsuseni, kus nummerdamine algab läänest 0-9ni ning lõunast 0-9ni. Koordinaati loetakse algul paremale, ning seejärel üles. Saadud paarisarvulise koordinaadi vasakpoolset osa nimetatakse idapikkuseks (ing k. *easting*) ning parempoolset osa põhjalaiuseks (ing k. *northing*). Nii näiteks asub Tartu Ülikooli arvutiteaduse instituudi peauks koordinaadil 35VME 83319 70859 [7].

Täpsused koordinaatide edastamisel olenevad sellest, mitmekohaline koordinaat anda.

0-kohaline MGRS koordinaat – 35VME (100000m täpsus)

2-kohaline MGRS koordinaat – 35VME 8 7 (10000m täpsus)

4- kohaline MGRS koordinaat – 35VME 83 71 (1000m täpsus)

6-kohaline MGRS koordinaat – 35VME 833 708 (100m täpsus)

8- kohaline MGRS koordinaat – 35VME 8332 7086 (10m täpsus)

10-kohaline MGRS koordinaat – 35VME 83319 70859 (1m täpsus)

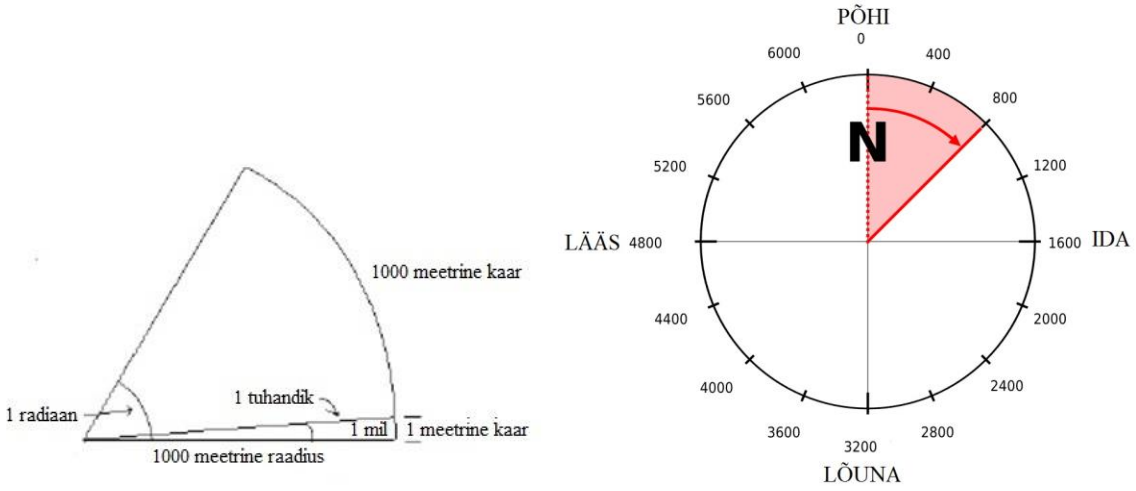
Tähtis on, et nii idapikkus kui põhjalaius antakse sama täpsusega, ehk mõlemad koordinaadi pooled peavad olema sama pikad. Miinipildujapatarei arvutuskeskuses kasutatakse 8-kohalisi koordinaate, sest see võtab vähem eetriaega, on kiirem edastada ning miinide kildude 100-meetrine mõjuala on piisavalt suur, et 1 meetrine täpsus pole lihtsalt vajalik.

NATO MIL tuhandiksüsteem

Seoses kaudtulesüsteemi levikuga 19. sajandil tekkis vajadus täpsema ühiku järgi, mille järgi määrata laskesuund. Olemasolev kraadide süsteem oli kaugemate sihtmärkide puhul ebatäpne ning minutite ja sekundite kasutamine oli raske. Esimest korda hakkas uut süsteemi, milliradiaane ehk tuhandiksüsteemi, kasutama Charles-Marc Dapples 19. sajandil [8]. Ta jagas ühe täisringi 6283.125 osaks ehk milliradiaanideks. I maailmasõja ajal hakkas Prantsuse armee kasutama kaudtulesüsteemide sihikutes 6400 tuhandiksüsteemi, mida tänapäeval teatakse kui NATO tuhandiksüsteemi [9].

Ühes täisringis on umbes 6,2832 radiaani, kus iga milliradiaan on tuhandik radiaanist ehk ring koosneb umbes 6283st tuhandikust, mis teeb ühe tuhandiku laiuseks 1000m kaugusel 1 meetri [10]. Lihtsustatuse ja mugavama kasutamise tõttu standardiseeriti see NATO poolt 6400ks tuhandikuks (mil). Tänapäeval kasutavad enamus NATO riike just 6400 tuhandiksüsteemi. Kuni 2007. aastani kasutas Rootsi 6300 tuhandiksüsteemi ning Vene Föderatsioon kasutab siiamaani 6000 tuhandiksüsteemi [11].

Antud süsteem tähendab, et 1 tuhandik võrdsustatakse 1000 meetri kaugusel 1 meetriga (vt joonis 4), kuigi tegelikkuses on see suurus 6283/6400 =~0,981718m.



Joonis 4. 6400 tuhandiksüsteemi jaotus.

Antud tuhandiksüsteemi kasutuselevõtt suurendas kaudtulesüsteemides täpsust, sest näiteks miinipildujate puhul, mille laskekaugus on kuni 8 km, on võimalik eksimus tuhandiku puhul seega ~8m.

Tuletellimuse formaadid

Vastase määratlemiseks kasutatakse nelja meetodit: koordinaatmeetod, polaarmeetod, tuleülekanne ja sihtmärgi number. Iga meetodi puhul on võimalik anda sihtmärk ka maa-ala sihtmärgina.

Koordinaatmeetodiga (lühend RUUT) edastatakse ainult 8-kohaline MGRS koordinaat ning vajadusel ka tulejuhi vaatlussuund.

Polaarmeetodiga (lühend POL) edastatakse tulejuhi 8-kohaline MGRS koordinaat, vaatlussuund sihtmärgile ning kaugus sellest meetrites.

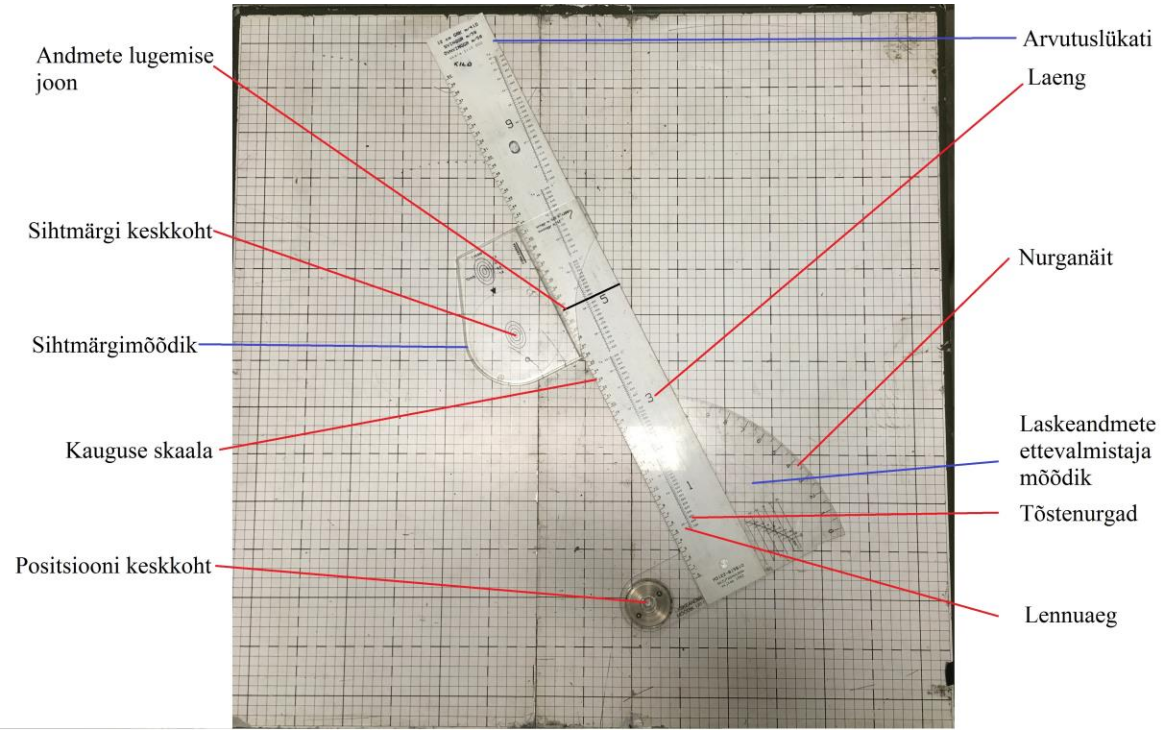
Tuleülekande (lühend TÜK) puhul edastatakse 8-kohaline ülekandepunkt, vaatlussuund ning korrektuur sellest punktist, ehk kas vasakule (lühend VAS), paremale (lühend PAR), suurendada (kaugemale, lühend SUR) või vähendada (lähemale, lühend VÄH).

Kõiki sihtmärke saab edastada ka maa-ala sihtmärgina (lühend MAA-ALA). Edastada tuleb maa-ala sihtmärgi suund. Antud suunas määratakse kolm erinevat sihtmärki, kus sihtmärkide vahe on 50 meetrit antud suunas. Iga sihtmärk määratakse erinevale positsioonile.

Sihtmärgi numbri puhul edastatakse ainult eelnevalt antud kinnitatud sihtmärgi number.

Arvutuslaua kasutamine arvutuskeskuses

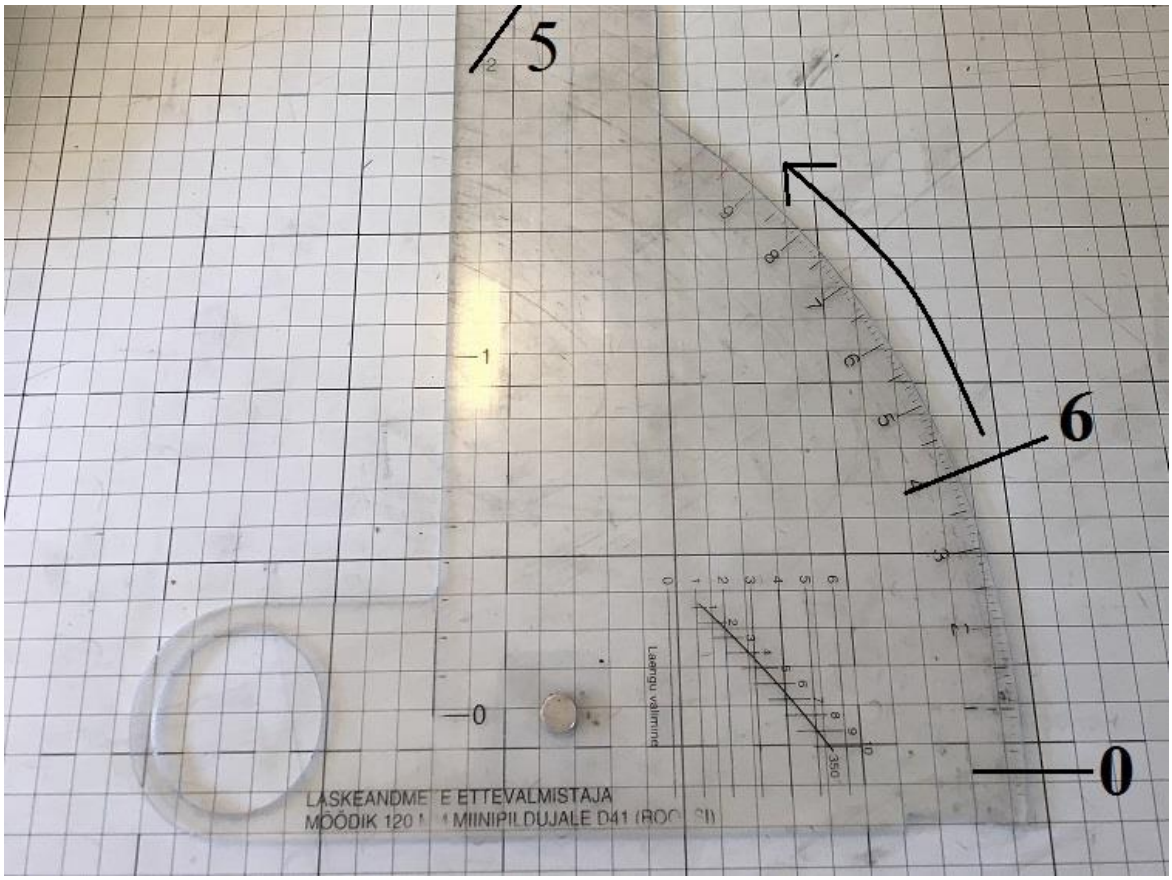
Arvutuslaud kujutab endast 1x1m suurust sentimeetervõrgustikuga lauda, mis on mõõtkavas 1:10000 ehk 1 cm laual võrdub 100 meetriga maastikul. Lisaks on komplektis laskeandmete ettevalmistaja mõõdik, liuglev sihtmärgimõõdik ning laskeandmetega erinevad arvutuslükatid. Kõigepealt on vajalik teha laua töökorda seadmine mis tähendab alljärgnevate andmete märkimist lauale: algandmed ehk põhisuund ja tulepositsiooni mõõdikupunkt, nurganäidu skaala, idapikkuste ja põhjalaiuste märkimine. Peale laua töökorda seadmist tuleb viia läbi laua kontroll, ehk valitakse suvaline sihtmärk positsiooni laskekauguses ja kontrollitakse, kas mõlema arvutuslauaga leitud andmed on samad, kusjuures vea erinevus kahel arvutamisel kauguse puhul võib olla +-30m ning suunas +-10 tuhandikku.



Joonis 5. Arvutuslaua komponendid ja kasutatavad andmed.

Nurganäidu skaala saamiseks pannakse ettevalmistaja mõõdik olenevalt põhisuunast kas suunda 0000, 1600, 3200 või 4800 ja märgitakse peale näidud (vt joonis 6). Ehk suunda

0000 paigutades märgitakse peale näidud 0, 6 ja 5. See tähendab, et on olemas näidud mõlemale poole vähemalt 1000 tuhandikku. Joonisel 6 saadakse suund 6400 või 0000. Nurganäit kasvab paremalt vasakule ning skaala on 10 tuhandikku täpsusega. 1 tuhandiku täpsus oleneb arvutaja hinnangust kahe kümnendiku vahelisest kaugusest.



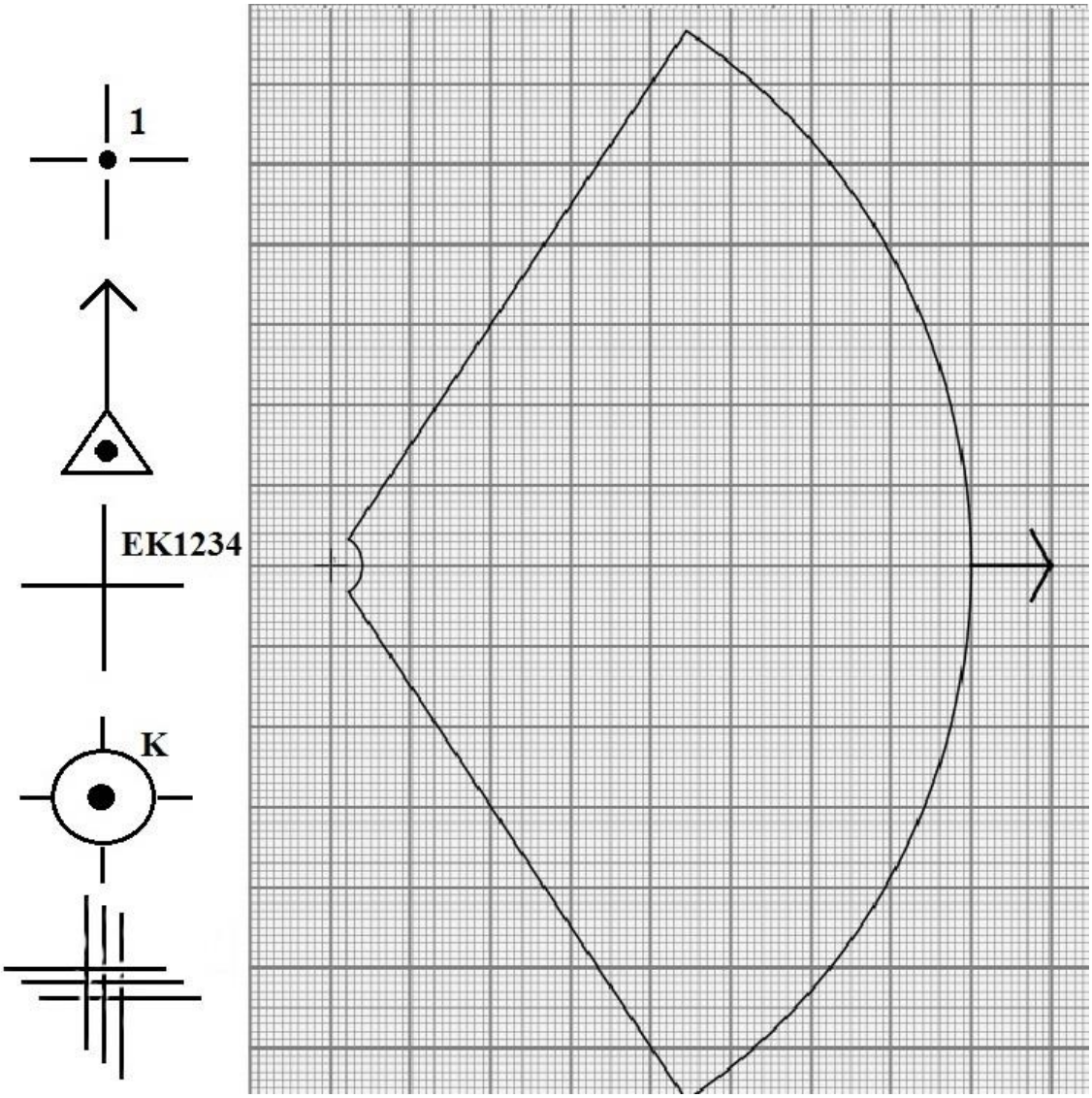
Joonis 6. Arvutuslaua nurganäidu skaala.

Arvutuslaual laskeandmete leidmiseks pannakse kõigepealt magnetiline rullik positsiooni keskkohta ehk mõõdikupunkti. Seejärel kas joonistatakse lauale õigesse ruudustikku sihtmärk ehk topograafiline punkt, või liigutatakse sihtmärgi mõõdik juba olemasoleva sihtmärgi peale nii, et sihtmärk jääks täpselt mõõdiku keskkohta. Nurganäidu skaalalt loetakse olemasoleva kriipsu juurest suund. Andmete lugemise joonelt saadakse nii kaugus, laeng, lennuaeg kui tõstenurk (vt joonis 5). Arvutuslükati on kahepoolne - ühel pool paarituarvulised laengud, teisel pool paarisarvulised. Kui saadud kaugusel on andmete lugemise joon kahe laengu vahelisel tühjal alal, peab lükati ümber pöörama ja lugema andmed teiselt poolt uue laenguga. Samuti on arvutuslükatile märgitud vastava moona ajaparandustegurid erinevate laengutega ning temperatuuriparandi graafik.

Arvutuslaua tingmärgid ja graafilised omapärad

Arvutuslaual peab olema kindlasti positsioonist põhisuunda suunatud sektor, mille vasak äär on 1000 tuhandikku põhisuunast vasakule ja parem äär 1000 tuhandikku paremale, ning mille lähim kaar lubatud minimaalsel ohutul laskekaugusel ehk 400m kaugusel ning kaugem kaar maksimaalse laskeulatuse ehk 8000m kaugusel. Seega on sektori kaare pikkuseks 2000 tuhandikku ehk sektori nurk on 112,5 kraadi. Lisaks peab olema sektoril nool põhisuunda. Joonisel 7 on näha alates ülevalt kasutatavad tingmärgid: mõõdikupunkt

koos positsiooni numbriga, tulejuhi asukoht, nool suunaga põhja, topograafiline sihtmärk koos sihtmärgi numbriga, ballistiline punkt koos miinitüübiga (K-kild, S-suits, V-valgus) ning maa-ala sihtmärk kolme topograafilise sihtmärgiga.



Joonis 7. Arvutuslaua tingmärgid ning põhisuunaga 1600 joonestatud laskesektor.

Ballistilisi punkte kasutatakse alates sellest hetkest, kui kasutaja on sisse viinud korrektuure ning esitatakse ainult selle miinitüübi ballistiline punkt, millisele miinitüübile korrektuur sisse viidi. Tulejuhi asukoht joonistatakse lauale ainult siis, kui sihtmärk antakse polaarmeetodiga. Tulejuhi nool näitab kokkuleppeliselt suunda 0000 ning antud nool ei oma seost antud polaarmeetodiga saadud suunaga. Veel joonistatakse lauale piirangute ala. See on vähemalt kolmest punktist koosnev hulknurk, kuhu võib arvutaja tuld tellida, kuid pole kohustatud seda jälgima.

4. Arvutusprogrammi nõuded ja struktuur

Arvutusprogramm kujutab endast virtuaalset arvutuslauda, mida on võimalik hiirega lohistada ning suurendada ja vähendada. Lisaks arvutuslauale on programmis võimalik sisestada erinevaid positsioone, sihtmärke, piiranguid ja muid olulisi asju. Programmi eesmärk on kiirelt ja täpselt arvutada laskmiseks vajalikke andmeid ning kuvada need ka arvutuslaual, et arvutaja saaks veenduda, et tema laud oleks vastavuses programmi poolt joonistatud olukorraga. Programm võib korraga kuvada kuni kolme erineva positsiooni laskeandmeid. Antud peatükis antakse ülevaade, milliseid nõudeid programm peab täitma, milliseid tehnoloogiad kasutatakse ning milline on programmi struktuur koos selle mõningate tähtsamate funktsioonidega.

4.1 Programmi nõuded

Järgnevas kahes alapeatükis kirjeldatakse milliseid funktsionaalseid ja mittefunktsionaalseid nõudeid peab loodud tarkvara täitma.

Funktsionaalsed nõuded

- Peab saama sisestada mitut positsiooni ning vajadusel peavad need olema ka muudetavad.
- Programm peab kuvama arvutuslauda, mida oleks võimalik suurendada ja vähendada ning ringi lohistada. Arvutuslaud peab olema suunaga põhja ning kuvama erinevaid sihtmärke ning positsioone.
- Peab saama sisestada kolmel erineval viisil sihtmärke. Lisaks peab saama neid muuta ning kustutada ning kiirelt nende vahel saama valida.
- Programm peab oskama vastavalt sisestatud temperatuurile ümber arvutada laskeandmeid.
- Programm peab oskama lasketabeleid sisse lugema ning laskeandmete tabeli uuenduste puhul ei tohi tekkida komplikatsioone.
- Olemasolevaid sihtmärke ning positsioone peab saama salvestada ning ka sisse lugeda.
- Programmis peab saama teha korrektuure sihtmärkidele ning programm oskama laskeandmeid ümber arvutada.
- Programm peab korraga kuvama kõiki võimalikke variante erinevate laengutega tule tellimiseks, nii kild-, suitsu- ja valgusmiini korral.
- Programm peab olema eestikeelne ning võimalikult lihtne ja üheselt arusaadav, et mis andmetega on tegu ja kuidas neid sisestada.

Mittefunktsionaalsed nõuded

- Arusaadav ja piisav dokumentatsioon.
- Kiire ning vähese jõudluse vajadusega.
- Programm peab suutma talletada vähemalt 50 erinevat sihtmärki.
- Võimalikult loogiline nuppude ja väljade asetus.
- Programm peab olema skaleeruv väiksemate ekraanide jaoks.
- Programm peab toetama ka vanemaid Windows süsteeme (XP, Vista, 7, 8).

4.2 Kasutatavad tehnoloogiad

Programmi arenduseks kasutatakse Eclipse IDE Neon 2 (4.6.2) arenduskeskkonda [12]. Programmi arenduskeeleks kasutatakse Javat, sest töö autor omab antud keeles kõige rohkem kogemusi ning samuti leidub Java keelele palju teeke ning abimaterjale veebis. Lisaks hilisema võimaliku edasiarenduse puhul Android operatsioonisüsteemile ei pea tervet lähtekoodi uuesti teise keelde kirjutama. Graafilise kasutajakeskkonna loomiseks kasutatakse JavaFX [13] lisateeki e(fx)clipse [14]. JavaFX graafika-ja meediapakettide teek lubab arendada erinevaid rakendusi nii *backend´*is Java keeles kirjutades kui *frontend´*i FXML-is, kasutades selleks eraldi SceneBuilder [15] nimelist tarkvara, millega saab kujundada ja siduda Java koodiga FXML-faile. JavaFX valiti selle uudsuse, internetis leiduvate õpetusmaterjalide ja lihtsa *drag and drop* disainimisprogrammi SceneBuilder-i pärast. Samuti toetab JavaFX teeki ka Android ja iOS operatsioonisüsteemid ning JavaFX teegil on tugi erinevatele puutetundlikele liigutustele nagu *scroll, zuum, swipe, rotate* jne. Lisaks on JavaFX-s teegis kasutusel liidesed ObservableList [16] ja Property [17], mis lubavad jälgida listis või mingis väärtuses toimuvaid muudatusi, mis on vajalik sihtmärkide ja positsioonide listi väärtuste muutuste jälgimiseks.

4.3 Sisendandmetena kasutatavad lasketabelid

Arvutusprogramm kasutab laskeandmete arvutamiseks kahte erinevat lasketabelite komplekti: 120mm ja 81mm. Mõlemas komplektis on kolm erinevat tabelit: kild-, valgus-, ja suitsumiin. Tabelid on TXT-formaadis ning kujul „suits_120.txt“ või „suits_81.txt“. Igas failis on loodud tabelid, mis on tabulaatoritega eraldatud (vt joonis 8). Esimene veerg igas tabelis on alati kaugus meetrites, mis algab võimalikust vähimast lastavast kaugusest kuni maksimaalse laskekauguseni, kusjuures sammuks on 10 meetrit. Iga laeng, mida antud moon toetab, on järgnevas kahes veerus, kus esimesena on tõstenurk ja teisena lennuaeg. Kui antud kaugusele on võimalik lasta mitme erineva laenguga, on laskeandmed olemas kõigil võimalikel laengutel. Kui sellele kaugusele pole võimalik lasta, on tabelis antud kohal arv 0. Erinevus siinkohal on ainsana kildmiinil, millel on 5 võimalikku laengut, alates nulllaengust kuni neljanda laenguni. Valgusmiinil ja suitsumiinil on 9 laengut, aga neil puudub nn nulllaen. Seega on lasketabelites nulllaengu kohal neil kirjas 0.

Kaugus	0 laeng		1. laeng		2. laeng		3. laeng	
1000	0	0	0	0	623	26.4	693	31.9
1010	0	0	0	0	620	26.4	691	31.9
1020	0	0	0	0	617	26.4	689	31.9
1030	0	0	0	0	614	26.4	687	31.9
1040	0	0	0	0	611	26.4	685	31.9
1050	0	0	0	0	608	26.2	683	31.8
1060	0	0	0	0	605	26.2	681	31.8
1070	0	0	0	0	602	26.2	679	31.8
1080	0	0	0	0	598	26.2	676	31.8

Tõstenurk Lennuaeg

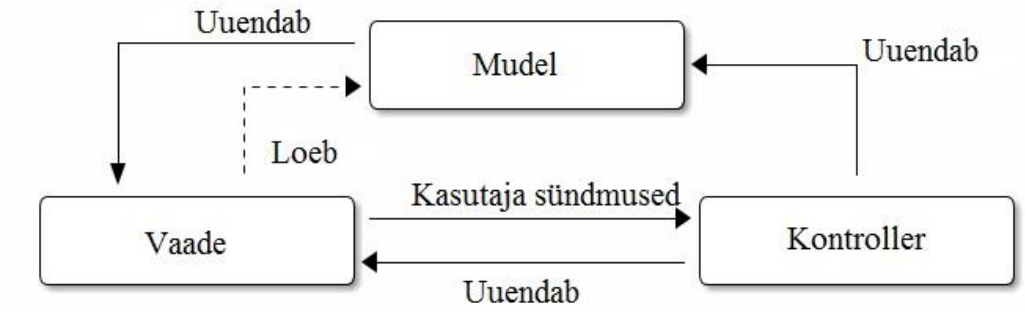
Joonis 8. Lasketabeli ülesehitus 120mm suitsumiini puhul.

Loodud kujul tabel annab võimaluse hilisemalt laskemoonade tabeleid uuendada programmi lähtekoodi muutmata, ning ei pane piiranguid minimaalsele ega maksimaalsele kaugusele, samuti kasutatavate laengute arvule.

4.4 Programmi struktuur

Programm on üles ehitatud Mudel-Vaade-Kontroller (lüh. MVC) disainimustrit kasutades. MVC disaini (vt Joonis 9) puhul on programm jagatud justkui kolmeks komponendiks [18] :

- Mudel - esindab vastava süsteemi andmeid ja loogikat sisaldavaid objekte.
- Vaade – esindab mudeli sisu kasutajale sobival kujul. Käsitleb, kuidas andmeid peaks olema graafiliselt esindatud ja paigutatud.
- Kontroller – reageerib kasutaja toimingutele vaates ning käsitleb neid toiminguid mudelis või vaates.



Joonis 9. MVC disain.

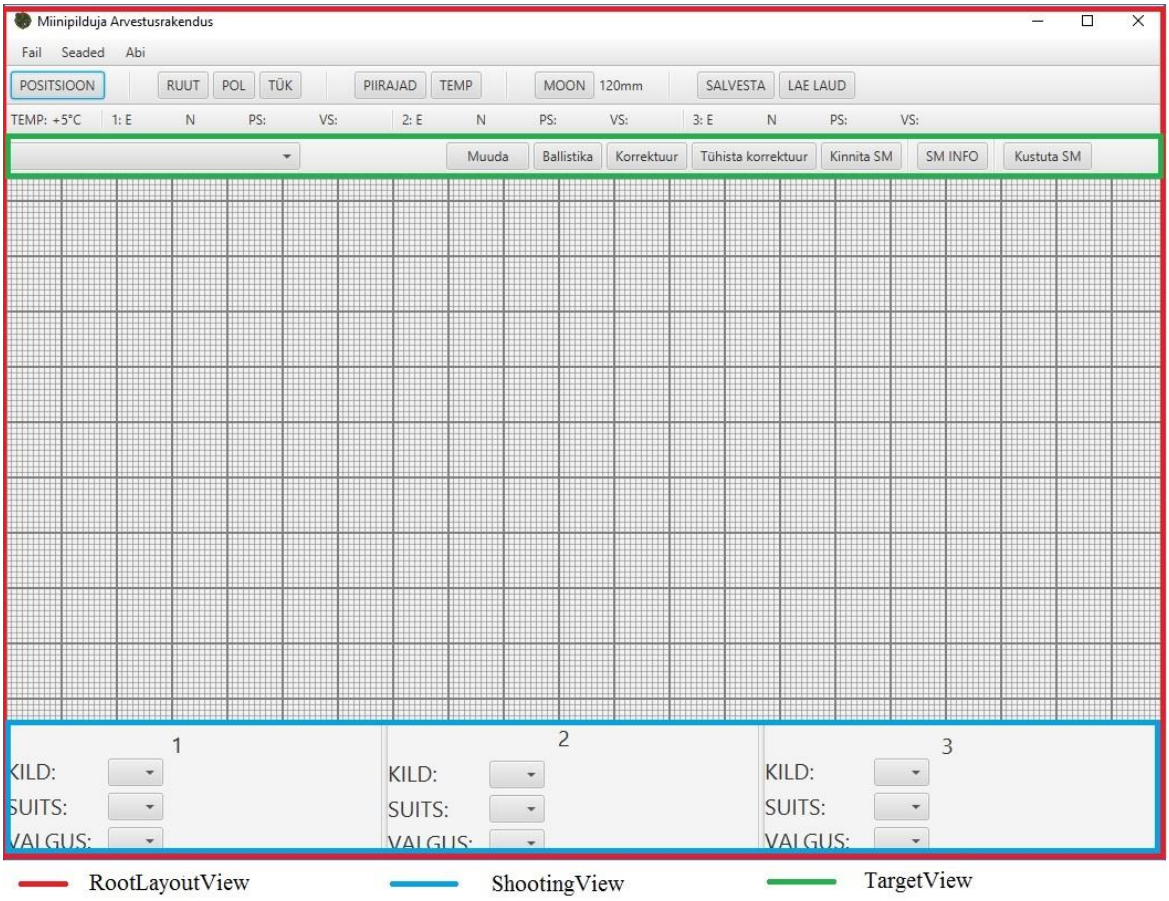
Järgnevalt kirjeldatakse, kuidas on arvutusprogramm üles ehitatud. Programm on jagatud kolme paketti:

- mppa – põhipakett kahe peamise kontrolleriga: MainApp.java ja ViewCanvas.java, kus MainApp on peamine loogika kontroller ning ViewCanvas on arvutuslaua graafika joonistamise klass, kus lisaks on ka klass SceneGestures reageerimiseks kasutajapoolsetele hiireliigutustele nagu *scroll, zoom* ja *drag*.
- mppa.model – sisaldab kahte programmis kasutatavat mudelit Position.java ja Target.java.
- mppa.view – sisaldab kõiki erinevaid vaateid ning nende vaadete kontrollereid.

Lisaks pakettidele sisaldab programm kausta *data*, kus on kõik lasketabelid, ja kausta *doc*, kus on programmi ingliskeelne dokumentatsioon.

MainApp peaklass

MainApp.java on arvutusprogrammi peaklass, mis laiendab *javafx.application.Application* klassi ja mis käivitab *start()* meetodiga programmi pealava [19] ehk nn peaakna, mis on programmi kõige kõrgem konteiner. Esimesena luuakse *RootLayoutView* (vt Joonis 10), mida kontrollib *PositionController* ja kuhu luuakse ka *ViewCanvas* klassi *canvas*, mille peale luuakse kõik graafika, seejärel *TargetView*, mida kontrollib klass *TargetController* ja siis *ShootingView*, mida kontrollib klass *ShootingController*.



Joonis 10. Arvutusprogrammi graafiline paigutus.

Pea iga vaate all oleva komponendi (nupp, *drop-down* nimekiri jne) jaoks on oma kontroller. Näiteks *RootLayoutView* all oleva „TEMP“ nupu jaoks on *PositionController* klassis meetod *handleTempEdit()*, mis kutsub välja *TempEdit* vaate, mida kontrollib *TempEditController*.

Lisaks programmi põhikontrollerite välja kutsumisele on peaklassis *MainApp.java* ka mitmed tähtsad andmekogumid.

```
private TreeMap<Double, ArrayList<ArrayList<Double>>> kild;
private TreeMap<Double, ArrayList<ArrayList<Double>>> valgus;
private TreeMap<Double, ArrayList<ArrayList<Double>>> suits;
```

Kolm esimest andmekogumit on vajalikud lasketabelite hoidmiseks. Igal moonatüübil on oma TreeMap [20] kujutis, kus kirjed on sorteeritud võtme järgi kasvavas järjekorras. See on vajalik kiireks väärtuse otsimiseks. Võtmeks on kaugus sihtmärgini ning väärtuseks on kahemõõtmeline järjend erinevate võimalike laengutega laskmiseks. Võtame näiteks kildmoona kujutise väärtuse, võtmega 2520.0 ehk 2520 meetrit.

```
System.out.println(kild.get(2520.0));
[[1.0, 365.0, 31.4], [2.0, 612.0, 45.8], [3.0, 687.0, 54.8], [4.0,
721.0, 60.6]]
```

Saadud väärtuseks on 4 erinevat järjendit, kus igaühes on laengu suurus, tõstenurk ning lennuaeg.

Järgnevad kaks andmekogumit on klassi ObservableList objektid, milles hoitakse kõiki sihtmärke ning positsioone.

```
private ObservableList<Target> targetData;
private ObservableList<Position> positionData;
```
Mõlemat järjendit peab iga võimaliku muudatuse korral uuendama. Näiteks peab *targetData* järjendis tehtud sihtmärgi andmete muudatuse puhul uuesti arvutama kõik laskeandmed igal positsioonil. Laskeandmeid hoitakse Position klassi objektides mitmemõõtmelises järjendis nimega *shootingData* (edaspidi laskeandmed).

```
private ObservableList<ArrayList<ArrayList<ArrayList<Double>>>> shootingData;
```
Andmete arvutamise ja uuendamise protsess käib alljärgnevalt:

1. Lisatakse uus sihtmärk, muudetakse aktiivse sihtmärgi andmeid, vahetatakse sihtmärki või muudetakse positsioonide andmeid .
2. Iga positsiooni laskeandmed tühjendatakse.
3. Iga positsiooni laskeandmetele arvutatakse aktiivse sihtmärgi iga moonatüübiga laskmiseks kaugus ning suund.
4. Otsitakse lasketabelitest vastavate moonadele ning kaugustele vajalikud väärtused ja lisatakse ka need laskeandmetesse.
5. Uuendatakse vastavad väljad ning graafika.

ViewCanvas klass

ViewCanvas klass on mõeldud graafika joonistamiseks ning sellel navigeerimiseks nii sisse ja välja suurendades kui ringi liikudes kasutades lohistamist. Antud klass laiendab javafx.scene.layout.Pane [21] klassi ja pärib ka paani (ing. k *pane*) meetodi getChildren(), millega saab lisada paani erinevaid objekte. Kõigepealt luuakse 3200x3200 suurune ruudustik klassina Canvas [22], mis kujutab endast 32x32 kilomeetervõrgustikku ja kasutades klassi GraphicsContext [23], joonistatakse ruudustik. Meetodiga *setCenterLines(String easting, String northing)* määratakse ära ruudustiku keskel oleva koha koordinaat, kuhu alati joonistatakse 2. positsioon. Juhul kui kasutatkse ainult ühe positsiooniga laskmist, joonistatakse sinna vastav positsioon. Seega tekib positsioonist põhja, lõunasse, idasse ja läände 16km võrgustikku. Ruudustiku suuruse määratlesin seetõttu 3200x3200, sest suuremaks minnes hakkas programm rohkem arvuti mälu kasutama ning reaalsuses korraga lihtsalt laual sihtmärke ei vaadelda kui neid , mis on laskeulatuses. Positsiooni uuendades liigutatakse vastavalt ka ruudustikku..

Probleem, mis tekib, on see, et paanil jooksevad koordinaadid x-teljel vasakult paremale, nii nagu MGRS koordinaatsüsteemis, aga y-teljel ülevalt alla. MGRS-süsteemis aga y-teljel jooksevad koordinaadid alt üles. Et teisendada MGRS-süsteemist lõuendi jaoks õigesse süsteemi, kasutan meetodit *fromCordToCanv(String easting, String northing)*. Antud meetod leiab idapikkuse ja põhjalaiuse järgi uued koordinaadid paanile kujundite lisamiseks, arvestades graafika keskel oleva positsiooni koordinaate.

Antud klassis on olemas meetodid erinevate kujundite joonistamiseks ning kustutamiseks. Kõik kujundid koosnevad erinevatest joontest, ringidest ja tekstidest, mis lisatakse järglastena ühte gruppi [24] (ing. k Group) ja seejärel lisatakse grupp paani järglaste hulka, mille tulemusena antud objektid joonistatakse graafikale. Objektide kustutamiseks eemaldatakse paani järglaste hulgast kogu grupp. Objektideks on nii sihtmärgid, positsioonid, piirangud, sektorid kui ballistilised punktid.

Tähtsamad meetodid

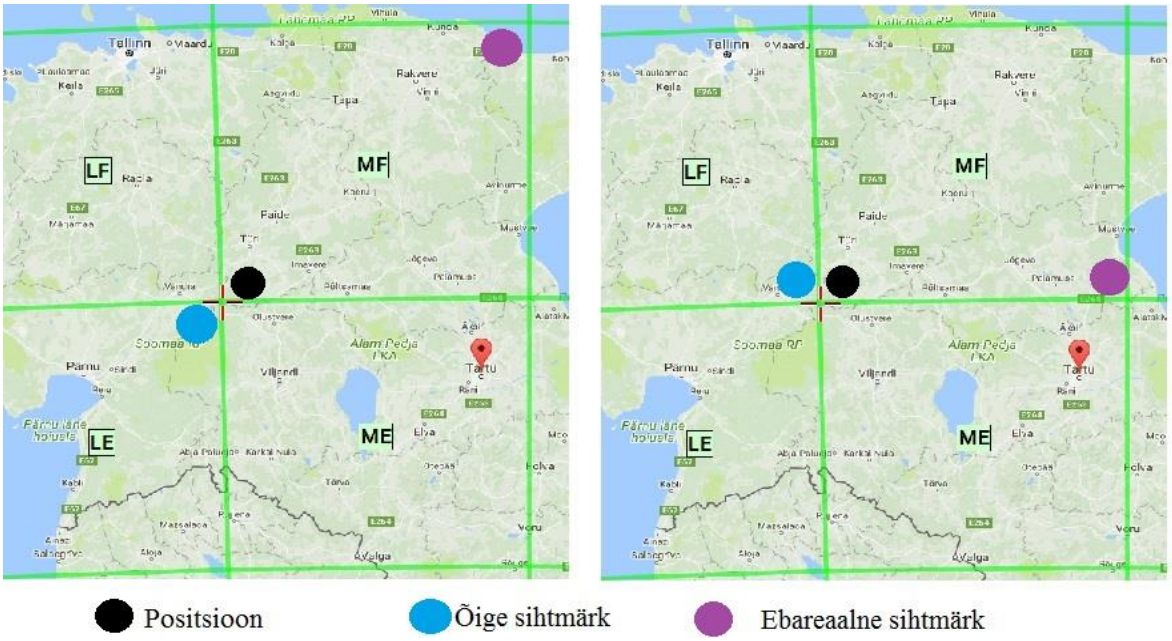
Siinkohal tuuakse ära mõned tähtsamad meetodid, mis on vajalikud erinevate andmete arvutamiseks.

Esimesena meetod *distanceCalculator(Position position, Target target)* klassis MainApp, mis arvutab sisse antud kahe parameetri, sihtmärgi ja positsiooni, vahelise kauguse ja suuna ning tagastab järjendina need iga moonatüübi jaoks. Võtame positsiooniks punkti $(x_1,\ y_1)$ ning sihtmärgiks punkti (x_2, y_2). Nende vahelise kauguse d leiame valemiga 1. Samuti on vaja leida nendevaheline suund . Leiame kõigepealt nendevahelise nurga α. Kusjuures, nurk α peab olema alguspunktki $(x_1,\ y_1)$ y-telje ja lõpppunkti (x_2, y_2) vaheline nurk. Javas saab selle jaoks kasutada funktsiooni *atan2(double x, double y)* [25], kus x=$x_2 - x_1$ ja y=$y_2 - y_1$. Saadud arv tuleb korrutada 180ga ja jagada π-ga, et saada nurk kraadides. Leitud nurk tuleb viia positiivsele kujule ning teisendada NATO tuhandikeks.

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2} \tag{1}$$

```
    double distance = Math.sqrt(Math.pow(positionEasting - targetEasting, 2)
        + Math.pow(positionNorthing - targetNorthing, 2))* 10.0;
    double angle = Math.atan2(targetEasting - positionEasting, targetNorthing -
positionNorthing)* 180/ Math.PI;
    if (angle < 0) {
        angle += 360.0; }
    angle = Math.round(angle * 17.777777777778); //Degrees to NATO MIL
```

Probleem, mis võib tekkida, on see, et programm peab kuidagi aru saama, millises 100 000km^2 ruutvõrgustikus on sihtmärk ja millises on positsioon. Võimalik on, et kui positsioon asub mingi 100 000 km^2 ruutvõrgustiku servas, et siis on arvutatav sihtmärk ebareaalne, ning õige sihtmärk asub hoopis kõrval olevas teises võrgustikus. Võtame näiteks võrgustikus 35VME 1000 1000 positsiooni ning andes talle sihtmärgiks asukoha 9000 9000, on nende kahe vaheline distants liiga suur, ning tegelikult peaks vastav sihtmärk olema hoopis võrgustikus 35VLE (vt joonis 11, vasakpoolne joonis). Arvutajad arvutuskeskuses saavad sellest kohe aru, sest ilmselgelt ei tellida tuld sihtmärki, mis asub juba üle 50 km kaugusel.



Joonis 11. Ebareaalsete sihtmärkide illustratsioon.

Et vältida ebareaalse sihtmärgi arvutamist, hakkab programm juhul kui arvutatud kaugus on suurem kui 50km, läbi käima erinevaid tingimusdirektiive, leidmaks millises võrgustikus reaalselt sihtmärk asuda võiks, ning hakkab arvutama uusi väärtuseid. Näiteks joonisel 10 esitatud vasakpoolse olukorra puhul kasutatakse järgmist tingimusdirektiivi:

```
double tooFarDist=50000.0;
if (distance > tooFarDist) {
if ((Math.abs(positionEasting - targetEasting) > 5000)
&& Math.abs(positionNorthing - targetNorthing) > 5000) {
    if (positionEasting > targetEasting) {
        targetEasting = 10000 + targetEasting;
    } else {
        targetEasting = -(10000 - targetEasting);
    }
    if (positionNorthing > targetNorthing) {
        targetNorthing = 10000 + targetNorthing;
    } else {
        targetNorthing = -(10000 - targetNorthing);
    }
    double distance1 = Math.sqrt(Math.pow(positionEasting - targetEasting, 2)
        + Math.pow(positionNorthing - targetNorthing, 2)) * 10.0;
    double angle1 = Math.atan2(targetEasting - positionEasting, targetNorthing -
positionNorthing)* 180 / Math.PI;
    if (angle1 < 0) {
        angle1 += 360.0;}
    angle1 = Math.round(angle1 * 17.777777777778);
}
```

Antud olukorra puhul on saadud sihtmärk diagonaalselt asetsevas kõrvalolevas ruudustikus, sest nii nende idapikkuste kui põhjalaiuste omavaheline erinevus üle 50 kilomeetri.

Teisena tuuakse ära meetodid *shootingInfo(ArrayList<ArrayList<Double>> shootingPoints)* ja *tempDiffCalc(doubledistance)*.

Meetod *shootingInfo(ArrayList<ArrayList<Double>> shootingPoints)* kutsutakse välja siis, kui igale positsioonile tahetakse välja arvutada laskeandmeid aktiivse sihtmärgi kohta. Sisendina saadav kahemõõtmeline järjend *shootingPoints* sisaldab kolme järjendit, kus esimeses järjendis on kildmiini laskmiseks vajalik kaugus ja suund, teise suitsumiini ja

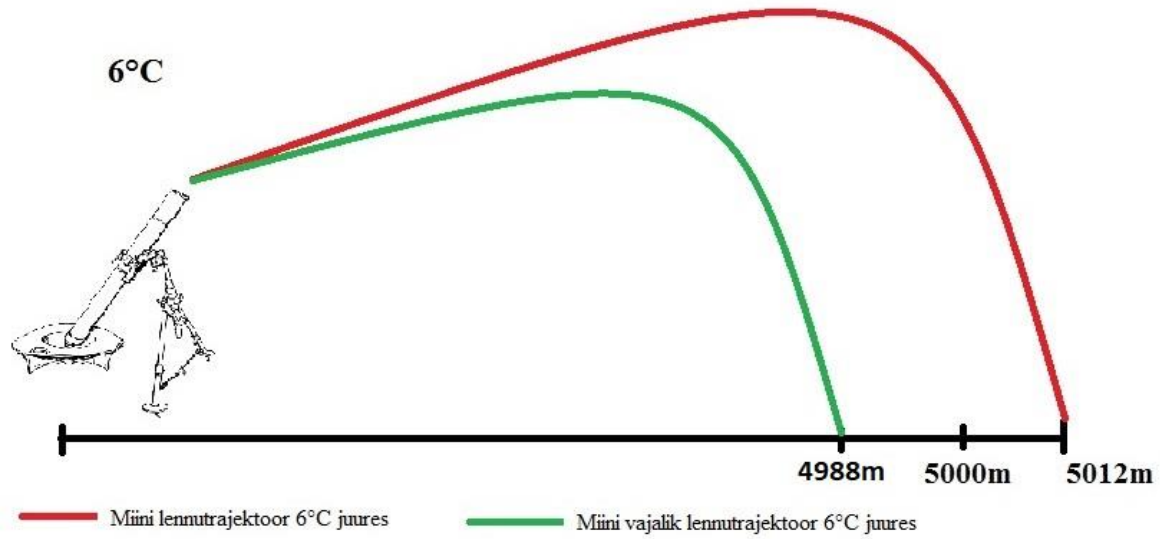
kolmandas valgusmiini vastavad andmed. Näitena tuuakse suitsumiini laskeandmete otsimise protseduur:

```
ArrayList<Double> suitsPoint = shootingPoints.get(1);
boolean tooFar2 = true; //Kui ei leita võtit, siis järelikult on kaugus liiga suur
for (double i : suits.keySet()) { // Läbitakse kõik lasketabeli võtmed
    if (tempDiffCalc(suitsPoint.get(0)) <= i) { //Arvutatakse kaugus temperatuuri järgi
        tooFar2 = false;
        ArrayList<ArrayList<Double>> suitsData = suits.get(i);
        data.add(suitsData);
        break;
        }
    }
if (tooFar2) {
    data.add(new ArrayList<>()); //Lisatakse tühi järjend, kui liiga kaugel
}
```

Kõigepealt läbitakse kasvavas järjekorras laskeandmete kogumi *suits* võtmete hulk. Kasutades funktsiooni *tempDiffCalc(double distance)* arvutatakse kõigepealt uus kauguse vastavalt temperatuurile. Kui leitud võti (kaugus positsioonist sihtmärgini) on väiksem kui arvutatud kaugus, kasutatakse leitud võtmele vastavaid laskeandmeid iga miinitüübi kohta eraldi. Seoses puuduva informatsiooniga, kuidas käitub uus kildmoon erineva temperatuuriga, ei teha temperatuuriparandust kildmiinile. Temperatuuril 5 °C ei rakendata temperatuurimuutu, sest see on suitsumiini ja valgusmiini tootjapoolne normaaltemperatuur. Temperatuuriparand arvutati välja käsitsi arvutuslauale erinevate vahemaade peal testides. Katse käigus tehti neljale erinevale kaugusele laskeandmete arvutamine. Iga arvutus viidi läbi 5 °C vahega, alates -30 °C kuni 30 °C. Saadud laskeandmete tõstenurk pandi kirja ja võrreldi selle kaugusega, kuhu antud moon lendaks normaaltemperatuuril. Võrreldes kauguseid erinevatel temperatuuridel, leiti, et kauguse kasv temperatuuri muutumisel on võrdlemisi lineaarne ning uus kaugus arvutatakse järgnevalt:

```
if (getTemp() < 5) {//getTemp() tagastab programmi temperatuuri
    double deltaTemp = Math.abs(getTemp() - 6.0);//Temperatuuri vahe
    double deltaDist = deltaTemp * 0.0016 * (distance);//Kauguse muut
    return distance + deltaDist;//Väiksema temperatuuri korral liidetakse
} else {
    double deltaTemp = Math.abs(getTemp() - 4.5);
    double deltaDist = deltaTemp * 0.0016 * (distance);
    return distance - deltaDist;//Suurema temperatuuri korral lahutatakse
```

Ehk siis näiteks 6°C suitsumiini lastes suureneb iga meetri peale kaugus 2,4mm võrra. Ehk näiteks 5000m peale lastes on see kauguse juba 12 meetrit. Põhjuseks lihtsalt miinides kasutatava lõhkelaengu efektiivsem süttimine. Mida kuumem on, seda kaugemale miin lendab. See aga tähendab seda, et õige kauguse arvutamiseks peab selle parandi maha lahutama esialgsest kaugusest.



Joonis 12. Miini lennutrajektoori muutus temperatuuri muutumisel.

Antud olukorda visualiseerib joonis 12. Seega tuleb temperatuuri suurenemisel alates 5°C leida väiksem kaugus, kui on tegelik vahemaa sihtmärgini, sest miin lendab temperatuuri tõttu juba kaugemale. Temperatuuri külmenedes tuleb teha vastupidist. See tähendab, et tuleb lasta kaugemale, et miin kukuks õigele kaugusele. Temperatuuriparandit ei kasutata kildmiini puhul, sest pole veel täpselt teada, kuidas temperatuur uut kildmiini mõjutab.

Kolmandana kirjeldan klassis PolarTargetController olevat meetodit *handleOk()*, mis kutsutakse välja kui kasutaja on sisestanud polaarmeetodiga sihtmärgi. Seetõttu on vaja arvutada sihtmärgi koordinaadid, teades vaid tulejuhi koordinaate, suunda ja kaugust sihtmärgini. Olgu siin algpunkt (x_1 ja y_1), kauguseks d ning nurgaks y-teljega α, siis leiame uue punkti (x_2, y_2) valemitega 2 ja 3.

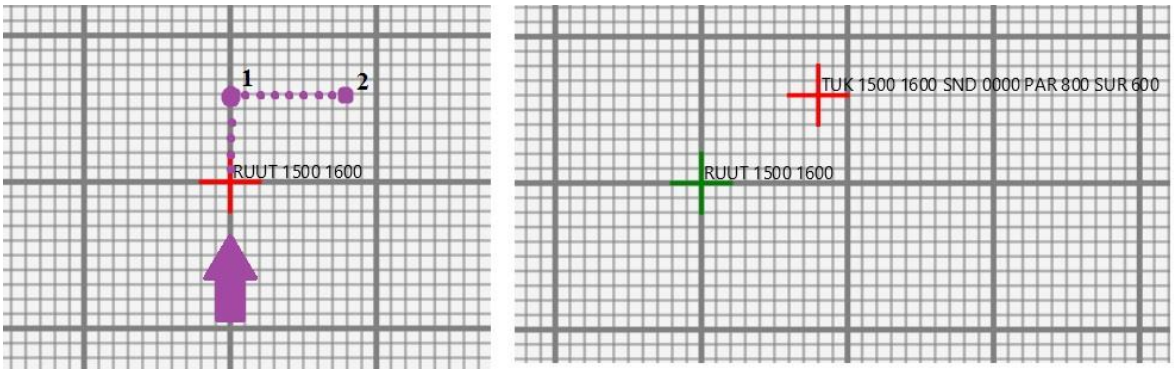
$$x_2 = x_2 + (d \times \sin(\alpha)) \tag{2}$$

$$y_2 = y_1 + (d \times cos(\alpha)) \tag{3}$$

Koodinäites on idapikkus ja põhjalaius viidud viiekohaliseks ehk 1m täpsuseks, sest antud kaugus on ka 1m täpsusega. Peale uue punkti leidmist viiakse leitud idapikkus ja põhjalaius uuesti neljakohaliseks.

```
int FOEasting = Integer.parseInt(easting.getText());//Tulejuhi easting
int FONorthing = Integer.parseInt(northing.getText());//Tulejuhi northing
int FODistance = Integer.parseInt(distance.getText()); //Kaugus sihtmärgini meetrites
double FODirectionAngle = Double.valueOf(FOdirectionAngle.getText());//Suund tuhandikes
double FOangle = Math.round(FODirectionAngle / 17.777777777778);//Suund kraadides
int targetEasting = (int) Math
    .round((FOEasting * 10 + FODistance * Math.sin(FOangle * Math.PI / 180)) / 10);
int targetNorthing = (int) Math
    .round((FONorthing * 10 + FODistance * Math.cos(FOangle * Math.PI / 180)) / 10);
```

Sama loogikat kasutades arvutatakse ka tuleülekandega sihtmärki, kuid seal peab arvestama seda, milline on tulejuhi vaatlussuund ülekandepunktile ning arvutama uue sihtmärgi kaks korda. Esimest korda arvutatakse uus sihtmärk ülekandepunkti suurendamise või vähendamisega ning teist korda saadud punkti liigutamisega kas paremale või vasakule.



Joonis 13. Tuleülekandega arvutatud sihtmärk.

Joonisel 13 on näidatud tuleülekandega punkti arvutamine. Sisendiks on ülekandepunkt 1500 1600, suunaks 0000, korrektuur paremale 800 ja suurenda 600. Esmalt arvutab klassis TukTargetController olev meetod *handleOk()* punkti 1 ja siis punkti 2.

5. Programmi nõuded ja juhendid

Antud peatükis tuuakse välja loodud tarkvara nõuded riistvarale ja tarkvarale ning programmi installeerimisjuhend koos kasutamisjuhendiga.

5.1 Arvutusprogrammi nõuded riistvarale ja tarkvarale

Programm vajab Java versiooni 1.8.x SE Runtime Environment [26] installeerimist, vähemalt 200 MB vaba muutmälu ning 1 MB vaba kettaruumi.

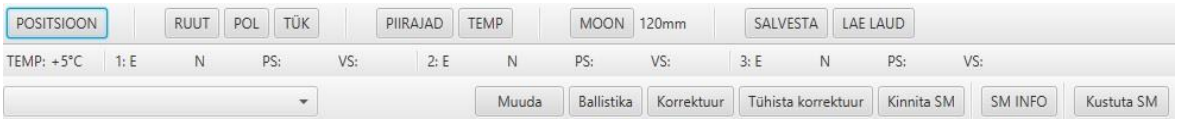
5.2 Installeerimis- ja käivitamisjuhend

Kogu arhiivi sisu eraldada ühte kausta nii, et kausta jääb ainult 2 faili nimega „logo.png“ ja „arvutus.jar“ ning 1 kaust nimega *data*.

Programmi käivitamiseks topeltklõps failil „arvutus.jar“ või paremklõps failil „arvutus.jar“, valida avanenud menüüst „Open with“ ja klõps „Java(TM) Platform SE binary“

5.3 Kasutamisjuhend

Arvutusprogrammi kasutamine peaks üldjuhul alati algama positsiooni(de) sisestamisega. Selle jaoks on programmi päises kaks tööriistariba (vt joonis 14).



Joonis 14. Päises asuvad kaks tööriistariba.

Vajutades nupule „POSITSIOON“ ilmub programmi uus aken, kus on võimalik sisestada kuni 3 positsiooni (vt joonis 15). Põhisund ja varusuund sisestatakse tavaliselt ümardatuna sajakohaliseni, ehk näiteks kujul 0300 või 4600. Peale „Salvesta“ vajutamist joonistatakse graafikale positsioon ning laskesektor. Juhul kui põhisuund ei sisestatud, laskesektorit ei joonistata. Teistkordselt nupule „POSITSIOON“ vajutades on võimalik positsiooni infot muuta.

Joonis 15. Positsiooni sisestamine.

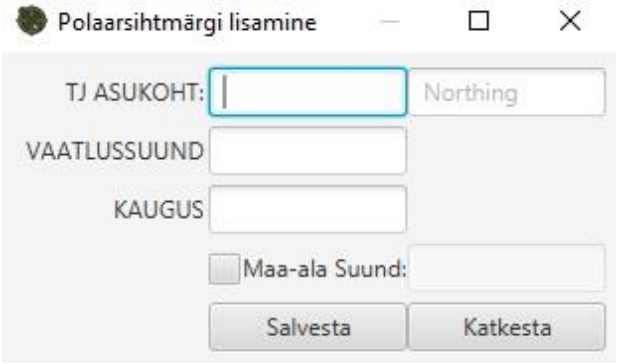
Järgnevalt võib lisada sihtmärgid, kasutades selleks kolme nuppu: „RUUT“, „POL“ või „TÜK“.

Iga nupp vastab erinevale sihtmärgi lisamise meetodile:

- „RUUT“ – koordinaatmeetod.
- „POL“ – polaarmeetod.
- „TÜK“ – tuleülekanne.

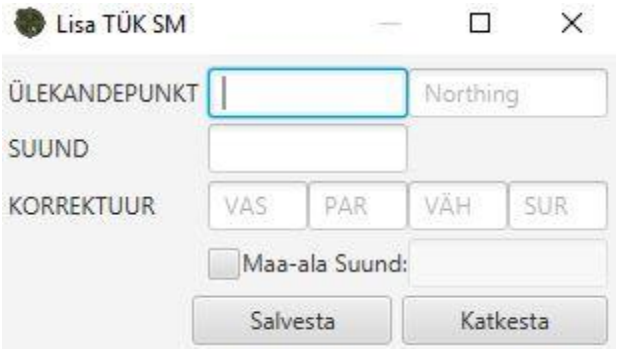
Joonis 16. Koordinaatmeetodil sihtmärgi lisamine.

Koordinaatmeetodile sihtmärgi lisamiseks piisab ainult idapikkuse ja põhjalaiuse sisesetamisest. Võib sisestada ka kohe sihtmärgi nime ning vaatlussuuna. Samuti on võimalik antud sihtmärk sisestada maa-ala sihtmärgina. Selle jaoks tuleb teha linnuke „Maa-ala“ kasti ning sisestada suund.



Joonis 17. Polaarmeetodile sihtmärgi lisamine.

Polaarmeetodil sihtmärgi lisamiseks tuleb täita kõik etteantud väljad, sisestades nii tulejuhi idapikkuse kui põhjalaiuse, tema vaatlussuuna ning kauguse sihtmärgini meetrites. Tehes linnuke „Maa-ala“ kasti ning sisestades suund, on võimalik sisestada antud sihtmärk ka maa-ala sihtmärgina.

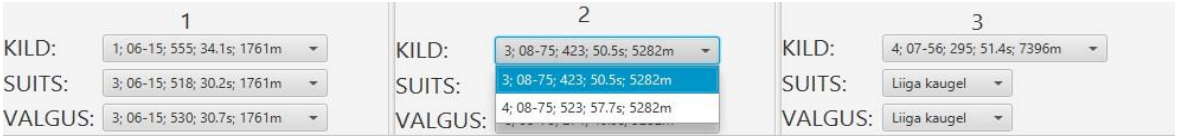


Joonis 18. Tuleülekandega sihtmärgi lisamine.

Tuleülekandega sihtmärgi lisamiseks tuleb sisestada ülekandepunkti idapikkus ja põhjalaius, vaatlussuund ning tehtav korrektuur meetrites. Maa-ala sihtmärgina lisamiseks tuleb teha linnuke „Maa-ala“ kasti ja sisestada korrektne suund.

Sihtmärke on võimalik hiljem muuta, kasutades nuppu „Muuda“ või kustutada kasutades nuppu „Kustuta SM“. Peale sihtmärgi lisamist ilmub sihtmärk graafikale ning jaluses olevatesse positsioonide menüüdesse ilmuvad laskeandmed (vt joonis 19). Klikkides

vabalt valitud rippmenüüle, kuvatakse kõikvõimalikud laskeandmete variandid. Juhul kui rippmenüüs on kirjas „Liiga kaugel“, siis tähendab see seda, et antud moonaga ei ole võimalik sellelt positsioonilt lasta, sest sihtmärk on liiga kaugel. Järjestus laskeandmetel on semikoolonitega eraldatud ning on järgnev : laeng, suund, tõstenurk, lennuaeg ning kaugus.



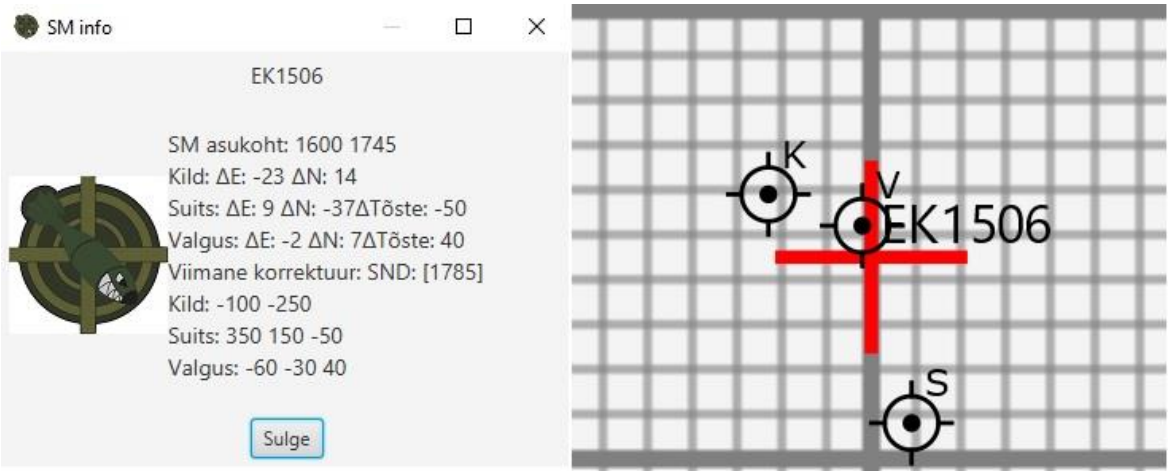
Joonis 19. Laskeandmete rippmenüüd.

Laskeandmeid on võimalik mõjutada ilma sihtmärgi või positsiooni koordinaate muutmata mitmel erineval moel. Esimene võimalus selleks on muuta temperatuuri. Selle jaoks on programmi päises ülemises tööriistaribas nupp „TEMP“ (vt joonis 14). Temperatuuri muutudes arvutab programm kõik laskeandmed ümber, sest temperatuuri muutmine toob endaga kaasa ka moona ballistiliste omaduste muutuse. Samuti on võimalik sisse viia korrektuure, kasutades päises olevas alumises tööriistaribas nuppu „Korrektuur“ (vt joonis 14).

Joonis 20. Sihtmärgile korrektuuri tegemine.

Korrektuuri tegemise aknas (vt joonis 20) tuleb sisestada korrektuurid meetrites. Vaatlussuund määrab ära mis suunas korrektuurid. Peale korrektuuride sisestamist ning vajutades nuppu „SALVESTA“ ilmuvad graafikale uued ballistilised punktid, kus iga ballistilise punkti kõrval on kirjas, millist miinitüüpi see punkt esindab. Nüüd arvutatakse kõikide miinitüüpide laskeandmed ümber uutesse vastavatesse ballistilistesse punktidesse.

Viimast tehtud korrektuuri on võimalik vaadata vajutades nuppu „SM INFO“ (vt joonis 21) ning viimast korrektuuri kustutada vajutades nuppu „Tühista korrektuur“. Korrektuure on võimalik järjest juurde lisada ning hiljem ühekaupa tühistada.

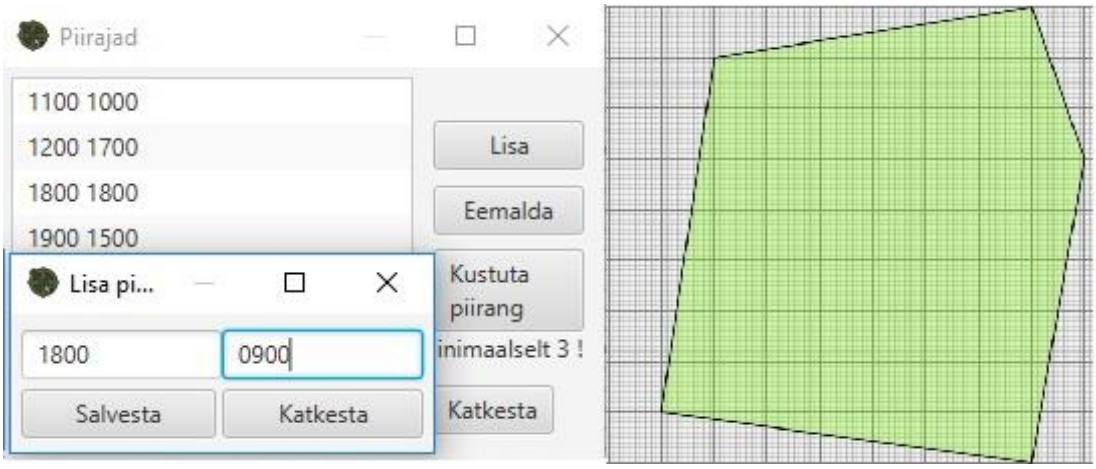


Joonis 21. Sihtmärgi info ja ballistilised punktid peale korrektuuri.

Sihtmärgi infot kuvav aken näitab ära ka sihtmärgi topograafilise punkti ning ballistiliste punktide vahelise erinevuse nii idapikkusel kui põhjalaiusel. Samuti on kuvatud ka tõste erinevused. Miinusmärgiga korrektuurid on vasakule, vähenda ja langeta.

Laskeandmed muutuvad ka siis kui kasutaja otsustab vahetada moona kaliibrit, kasutades selleks programmi päises olevas esimese tööriistaribas nuppu „MOON“. Kaliibreid on kaks: 81mm ning 120mm. Peale kaliibri vahetamist arvutatakse laskeandmed ümber uue moonaga.

Kasutajal on võimalik joonistada graafikale ka piirangute ala kasutades päises oleval tööriistaribal nuppu „PIIRAJAD“. Ilmuvas aknas (vt joonis 22, vasakpoolne joonis) tuleb järjest sisestada hulknurga tipud. Nupuga „Lisa“ avaneb uus aken, kuhu tuleb sisestada tipu idapikkus ja põhjalaius. Nupuga „Salvesta“ lisatakse tipp järjendisse. Tippude lisamisel on järjekord tähtis. Ehk siis tipud tuleb sisestada kas päripäeva või vastupäeva. Minimaalselt peab hulgnurga joonistamiseks sisestama vähemalt 3 tippu. Peale „Piirajad“ aknas nupule „Salvesta“ vajutamist joonistatakse graafikale piirangute ala (vt joonis 22, parempoolne joonis). Tervet ala on võimalik kustutada vajutades uuesti nupule „PIIRAJAD“ ning seal omakorda nupule „Kustuta piirang“.



Joonis 22. Piiranguala joonistamiseks vajalike koordinaatide sisestamine ja joonistamine

Arvutusprogrammi seisu on võimalik salvestada ning hiljem peale programmi taaskäivitamist ka sisse lugeda programmi päises oleva tööriistariba nuppudega „SALVESTA“ ja „LAE LAUD“.

6. Töö analüüs ja võimalikud edasiarendused

Selles peatükis analüüsitakse loodud tarkvara ning tuuakse välja võimalikud edasiarendused.

6.1 Valminud töö analüüs

Valminud programmi võib analüüsida kahest erinevast aspektist. Esimene neist on valminud programmi võrdlemine käsitsi arvutamisega ning teiseks valminud programmi võrdlemine varemloodud arvutusprogrammiga.

Käsitsi arvutamisega võrreldes saab välja tuua mõningaid näitajaid, mille tõttu on loodud programm efektiivsem:

- Aeg – käsitsi arvutamisel on ajaline tegur erinev sõltuvalt arvutajast, kuid nullseisust, ehk täiesti tühja laua puhul, laskandmete leidmiseks ühele koordinaatmeetodil antud sihtmärgile ühe positsiooni puhul loetakse 90 sekundit. Loodud programmi puhul on see aeg ~11-12 sekundit, kasutades väljade vahel liikumiseks tabulaatorit ning numbrite sisestamiseks numbristikku. Ajalisest faktorist on ka oluline see, et arvutusprogrammil ei pea tegema arvutuslaua töökorda seadmist, mille normaalajaks loetakse 3 minutit.
- Täpsus – käsitsi arvutamisel tekivad inimliku faktori tõttu sisse mõõtetäpsuse vead. Näiteks nurganäidu lugemisel on vähimaks skaala kriipsuks 10 tuhandikku, mistõttu on nurganäidu ühendike lugemisel väga lihtne eksida. Programm aga arvutab tuhandiku täpsusega nurganäidu (suuna). Samuti on kauguse mõõtmiseks lükatil kahe väikseima kriipsu vahe 50 meetrit. Programm aga on suuteline kauguse leidma 1 meetri täpsusega.
- Lasketabelite uundatavus ja loetavus – käsitsi arvutamisel puudub võimalus laskeandmeid lükatil mõistlikul viisil uuendada või muuta. Samuti peab erinevate laengute ja moonade jaoks kasutama erinevaid lükateid, mis kõik kulutab aega. Programmis aga on kõik laskeandmed loetavad samal ajahetkel, ning kõik erinevate laengutega laskmiseks olevad variandid kuvatavad.
- Laua seisu salvestamine ja uuesti lugemine – käsitsi arvutamisel peab asukohta muutes üle 3-4 km suure tõenäosusega terve laua kustutama ning uuesti joonistama positsioonid ja sihtmärgid. Ainus võimalus mingit kindlat seisu salvestada on kõikide laual olevate sihtmärkide üles joonistamine eraldi paberile ning hiljem kopeerimine. Programmis on lihtne salvestada nii palju võimalikke laua seise kui tahes vajalik.

Vana programmiga võrreldes on loodud programmis mitmeid varem mitteesinenud uuendusi:

- Lasketabelite uuendatavus – lasketabelid on nüüd lihtsalt uuendatavad .txt kujul olevaid lasketabaleid muutes.
- Mitme arvutuslauaseisu hoidmine – võimalus salvestada nimeliselt erinevaid arvutuslaua seisusid ning neid hiljem sisse lugeda.
- Erinevate kaliibrite kasutamine – võimalus kasutada nii 120mm kui 81mm kaliibriga laskmiseks vajalikke lasketabeleid.
- Hulknurga loomine piiranguala joonistamiseks – võrreldes vana programmiga, kus sai luua ainult nelinurkse piiranguala, on looduda programmis võimalik luua hulknurkne nii paljude nurkadega piiranguala kui kasutaja soovib.

- Väljade asetus – paljude väljade asetus on selgem, ajaliselt kokkhoidvam ning eksimusi vähendav. Näiteks pidi varem korrektuuride sisestamisel teadma, et vasakule korrektuur tuleb sisestada negatiivselt ning paremale korrektuur positiivselt. Samuti pidi vanas programmis sisestama enne sihtmärgi, ning siis sellele nime andmiseks kinnitama. Sihtmärgi lisamisel pidi kõigepealt hiirega klikkima väljal ja siis sai kasutaja alles trükkida, loodud programmil aga on automaatselt võimalik esimesse välja trükkida. Puudus ka võimalus sihtmärgi andmeid muuta.
- Sihtmärgi korrektuuride kuvamine – Loodud programm kuvab nupu „SM INFO“ all viimase loodud korrektuuri info.
- Aktiivse sihtmärgi kuvamine – Sihtmärk, mille kohta hetkel laskeandmed kuvatakse, on kasutajale näha punaselt. Kõik teised sihtmärgid aga rohelised. Välistab selle, et kasutaja valed laskeandmed loeb mingi sihtmärgi puhul.

Võib väita, et loodud arvutusprogramm on kindlasti parem, kui seni kasutusel olnud programm. Programm rahuldas vajalike korrektsete laskeandmete lugemise juba varasemas arenguetapis, kui puudus graafika, kuid oli võimalik sisestada erinevaid sihtmärke ja teha korrektuure. Esmane testimine kestis terve õppuse Kevadtorm 2017, mille jooksul kasutati antud tarkvara paralleelselt arvutuslaudadega, et koguda andmeid kasutajaliidese mugavdamiseks ning arvutuskäikude õigsuse kontrollimiseks.

Arvutid, mille peal programmi on testitud, on olnud väga erinevate tehniliste näitajatega. Programmil ei tekkinud käivitamisprobleeme ka Windows XP Service Pack 3 operatsioonisüsteemi kasutanud sülearvutitega.

6.2 Võimalikud edasiarendused

Võimalusi, kuidas programmi teha efektiivsemaks ning paremaks on mitmeid, millest toon ära mõned võimalused:

- Veebipõhise topograafilise kaarti või ortofotode kasutamine graafika põhjana. Eeliseks oleks nähtava maastiku parem hindamine ning arusaam, kuhu tegelikult tuld tellitakse. Samuti oleks näha nii looduslikud (orud, mäed, järved jne) kui tehislikud (majad, karjäärid) takistused, mis tähendaksid tegelikkuses suurema koguse moona laskmist. Miinuseks aga oleks veebiühenduse olemasolu, mis teeks programmi kasutatava arvuti lihtsamini leitavaks vastase poolt. Teine võimalus kasutada eellaetud mingisuguse ala topograafilist baaskaarti, kuid see oleks väga mahukas. Samuti ka suureneb jõudluse vajadus baaskaartide laadimisega.
- Lisada juurde ka suurtükiväe toetamiseks mõeldud laskeandmeid, muutes seeläbi programmi veel universaalsemaks. Eeliseks ühtse programmi võimalik kasutamine nii miinipilduja kui suurtüki laskeandmete arvutamiseks.
- Arvutusprogrammi arendamine nutiseadmele. Arvestades nutitelefonide ekraanide suurust tuleks ilmselt arendada mitmes vaates programm, kus oleksid laskeandmed ning graafika eraldi vaadetena. Eeliseks oleks programmi portatiivsus ning suure kasutajate arvu olemasolu. Miinuseks aga nutitelefonide poolt välja saadetavad signaalid ning kiirgused, mis on tuvastatavad ning positsioneeritavad. Ühe võimaliku lahendusena oleks siinkohal e-lugeri kasutamine, mis ei kiirga nii palju erinevaid signaale.

- Eelnevast tulenevalt on võimalik rakenduse portimine nii Android kui ka Linux operatsioonisüsteemi kasutavatele seadmetele. Õppuste käigus läbiviidavate laskeandmete arvutamiseks välitingimustes oleks kõige ideaalsem arendada rakendus just eelkõige mõelduna tahvelarvutile, arvestades seda, kui palju infot peab ekraan mahutama nii, et see oleks lihtsalt loetav. Välitingimusi arvestades võiks tahvelarvuti olla ka IP68 veekindlusega, nagu näiteks on seda Sony Xperia Z4 Tablet [27].
- Arvutusprogrammi välitestimine paralleelselt arvutuslaudadega. Vajalik oleks simuleerida võimalikult tihe tule tellimine ning laskeandmete arvutamine. Paralleelselt arvutuslaudadega testimine annab tagasisidet, kui palju kiirem ning täpsem on arvutusprogrammiga arvutamine. Samuti on simulatsiooni käigus võimalik, et tulevad välja mingisugused kitsaskohad kasutajaliideses või võimalikud programmeerimisvead.
- Lisafunktsioonide arendamine. Näiteks vajaks arendamist ballistiliste omaduste kasutamist võimaldav funktsioon. See tähendab seda, et kui mingile sihtmärgile on tehtud korrektuure ja sihtmärk kinnitatud, siis rakenduksid automaatselt sellele sihtmärgile tehtud korrektuurid kõikidele sihtmärkidele mis on sellest sihtmärgist vähem kui 1500 meetri kaugusel.
- Lõpmatu võrgustiku arendamine. Arendada oleks vaja praeguse võrgustiku asemele vähem arvutusjõudlust vajav lõpmatu võrgustik, mis kuvaks ka jooksvalt võrgustiku joonte numbreid, ning mida saaks igas suunas lõpmatult liigutada.

7. Kokkuvõte

Käesoleva töö eesmärk oli luua universaalne tarkvaralahendus kaudtulesüsteemide arvutuskeskuse tööks vajalike andmete arvutamiseks, kasutades algandmetena laskemoonade lasketabeleid. Tähtis oli loodava lahenduse puhul, et oleks olemas võimalus hilisemalt lasketabeleid uuendada ilma lähtekoodi muutmata.

Töö alguses kirjeldati, kuidas käib laskeandmete arvutamine käsitsi ning millised piirangud ja võimalused arvutuslaual käsitsi arvutades eksisteerivad. Samuti anti ülevaade arvutamiseks vajalikust MGRS-süsteemist ja suundade leidmiseks vajalikust tuhandiksüsteemist. Töös on kirjeldatud loodud tarkvara struktuuri, kasutatavaid tehnoloogiaid ning tähtsamaid meetodeid. Kirjeldatud on ka loodud tarkvara nõudeid riist- ja tarkvarale. Samuti on lisatud tarkvara kasutamiseks vajalik kasutamisjuhend.

Antud töö eesmärk oli luua universaalne ning uuendatavate lasketabelitega arvutusprogramm, mis oleks võimeline mitme positsiooniga arvutama laskeandmeid erinevatesse sihtmärkidesse. Programmi loomisel arvestati ajateenijatest arvutajate vajadusi ning programmi kasutajaliidese on disainitult selliselt, et laskeandmete arvutamine oleks võimalikult kiire ja kasutajasõbralik. Arvutatud laskeandmeid peab olema võimalik mõjutada temperatuuri muutmisega ja korrektuuride tegemisega. Samuti peab tarkvara kasutajale kuvama positsioonide ning sihtmärkide omavahelist asetust MGRS-koordinaatsüsteemis.

Töö tulemusena said täidetud kõik püstitatud eesmärgid ning arendati juurde mitmeid funktsioone, et muuta tarkvara veelgi efektiivsemaks ja kiirendada laskeandmete arvutamist.

Loodud tarkvara saab kindlasti edasi arendada veel mitmekülgsemaks ning vähem arvutusnõudlust vajavaks. Samuti on võimalik juurde arendada mitmeid lisavõimalusi nagu näiteks topograafiliste kaartide kuvamine.

Töö autor tänab militaarterminoloogiliste küsimuste lahendamise eest Kuperjanovi jalaväepataljoni miinipildujapatarei tegevväelasi ning ajateenijatest arvutajaid, kes aitasid kaasa esmasele testimisele ning kasutajaliidese mugavdamisele. Samuti tänab töö autor juhendamise ning nõustamise eest professor Eero Vainikkot.

8. Viidatud kirjandus

[1] [Võrgumaterjal]. Available: https://en.wikipedia.org/wiki/Indirect_fire. [Kasutatud Juuni 2017].

[2] NATIONAL GEOSPATIAL-INTELLIGENCE AGENCY, Veebruar 2014. [Võrgumaterjal]. Available: http://earth-info.nga.mil/GandG/publications/NGA_STND_0037_2_0_0_GRIDS/NGA.STND.0037_2.0.0_GRIDS.pdf.

[3] [Võrgumaterjal]. Available: https://www.maptools.com/mgrs_history.

[4] C. C. J. A.-h. C. o. U. M. G. R. S. W.H.Mills, 27 Veebruar 1948. [Võrgumaterjal]. Available: https://www.maptools.com/bundles/maptoolsstaticpages/pdfs/MGRSHistory/Letter_JMPC_Ad_Hoc.pdf. [Kasutatud 17 Juuni 2017].

[5] National Geospatial-Intelligence Agency, „Military Map Reading 201," [Võrgumaterjal]. Available: http://earth-info.nga.mil/GandG/coordsys/mmr201.pdf. [Kasutatud 17 Juuni 2017].

[6] M. Rittri, „Map of the Military Grid Reference System (MGRS) around the North Pole, with the AA lettering scheme for the 100 km squares south of 84°N," 2007.

[7] [Võrgumaterjal]. Available: https://mappingsupport.com/p/gmap4.php?mgrs=35VME83327086&tilt=off&z=8&t=t1&markers=||58.378328,%2026.714728^^%3Cspan%20style=%27font-weight:bold;font-size:1.2em;%27%3E35VME83327086%3C/span%3E..

[8] [Võrgumaterjal]. Available: http://www.metric-conversions.org/angle/milliradians-nato-conversion.htm.

[9] Juuli 2017. [Võrgumaterjal]. Available: https://en.wikipedia.org/wiki/Milliradian.

[10] R. J. Simeone, „MILS and MOA A Guide to understanding what they are and how to derive the Range Estimation Equations," [Võrgumaterjal]. Available: http://www.mil-dot.com/media/1027/the_derivation_of_the_range_estimation_equations.pdf. [Kasutatud 27 Juuni 2017].

[11] J. Huffman, 2000. [Võrgumaterjal]. Available: http://www.boomershoot.org/general/mils.htm.

[12] [Võrgumaterjal]. Available: https://eclipse.org/downloads/packages/release/Neon/2.

[13] [Võrgumaterjal]. Available: http://docs.oracle.com/javase/8/javafx/get-started-tutorial/jfx-overview.htm#JFXST784.

[14] [Võrgumaterjal]. Available: http://www.eclipse.org/efxclipse/index.html.

[15] [Võrgumaterjal]. Available: http://gluonhq.com/products/scene-builder/.

[16] [Võrgumaterjal]. Available: https://docs.oracle.com/javase/8/javafx/api/javafx/collections/ObservableList.html.

[17] [Võrgumaterjal]. Available: https://docs.oracle.com/javase/8/javafx/api/javafx/beans/property/Property.html.

[18] R. Eckstein, Märts 2007. [Võrgumaterjal]. Available: http://www.oracle.com/technetwork/articles/javase/index-142890.html.

[19] [Võrgumaterjal]. Available:

https://www.tutorialspoint.com/javafx/javafx_application.htm.

[20] [Võrgumaterjal]. Available: https://docs.oracle.com/javase/7/docs/api/java/util/TreeMap.html.

[21] [Võrgumaterjal]. Available: https://docs.oracle.com/javase/8/javafx/api/javafx/scene/layout/Pane.html.

[22] [Võrgumaterjal]. Available: https://docs.oracle.com/javase/8/javafx/api/javafx/scene/canvas/Canvas.html.

[23] [Võrgumaterjal]. Available: https://docs.oracle.com/javase/8/javafx/api/javafx/scene/canvas/GraphicsContext.html.

[24] [Võrgumaterjal]. Available: https://docs.oracle.com/javase/8/javafx/api/javafx/scene/Group.html.

[25] [Võrgumaterjal]. Available: https://docs.oracle.com/javase/7/docs/api/java/lang/Math.html#atan2(double,%20double).

[26] [Võrgumaterjal]. Available: http://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133155.html.

[27] „Xperia Z4 Tablet,“ Sony, [Võrgumaterjal]. Available: https://www.sonymobile.com/global-en/products/tablets/xperia-z4-tablet/. [Kasutatud 12 August 2017].

Lisad

I. Lähtekood ning dokumentatsioon

Tööga on kaasas ZIP-fail, mis sisaldab lähtekoodi ja lähtekoodi dokumentatsiooni. Dokumentatsioon asub kaustas *doc* ning käivitatav JAR-formaadis programm kaustas *runnable*.

II. Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina, **Argo Neumann**,
(*autori nimi*)

1. annan Tartu Ülikoolile tasuta loa (lihtlitsentsi) enda loodud teose **Miinipildujapatarei arvutusprogramm**,
(*lõputöö pealkiri*)

mille juhendaja on Eero Vainikko,
(*juhendaja nimi*)

1.1. reprodutseerimiseks säilitamise ja üldsusele kättesaadavaks tegemise eesmärgil, sealhulgas digitaalarhiivi DSpace-is lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;
1.2. üldsusele kättesaadavaks tegemiseks Tartu Ülikooli veebikeskkonna kaudu, sealhulgas digitaalarhiivi DSpace´i kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.

2. olen teadlik, et punktis 1 nimetatud õigused jäävad alles ka autorile.
3. kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest tulenevaid õigusi.

Tartus, **28.08.2017**