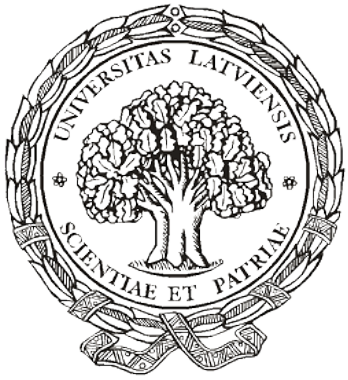


LATVIJAS UNIVERSITĀTE
DATORIKAS FAKULTĀTE

GALDA SPĒLES REALIZĀCIJA TIEŠSAISTĒ

KVALIFIKĀCIJAS DARBS

Autors: **Reinis Elksnis**
Studenta apliecības Nr.: RE11003
Darba vadītājs: Mg. Dat. Jānis Iļjins

Rīga, 2014

ANOTĀCIJA

Kvalifikācijas darbā "Galda spēles realizācija tiešsaistē" tiek veidota galda spēles "Chaos in the Old World" daļēja funkcionalitātes implementācija.
Projekta galvenais uzdevums ir izstrādāt platformu - spēles klientu un serveri, kuri nodrošina spēles spēlēšanu tiešsaistē un, kurus vēlāk būs iespējams papildināt ar pilnīgāku spēles loģikas implementāciju. Sistēma izstrādāta Java valodā, izmantojot XML-RPC datu apmaiņas protokolu.

Atslēgvārdi: Java, spēle, tiešsaiste.

ABSTRACT

Poject "Online board game" is a partial implementation of board game "Chaos in the Old World". It's main objective is to develop game client and server that can transfer player actions using network connection. Project has been developed so that it could be later enhanced with more complete game logic. Program has beeen developed using Java and XML-RPC data transfer protocol.

Keywords: Java, game, online

SATURS

Definīcijas un akronīmi

Instance - specifisks objekta gadījums.

Lokālais lietotājs - lietotājs, kas ir iedarbinājis esošo programmas instanci.

Tīkla lietotājs - ir lietotājs, ar kuru tiek komunicēts ar interneta starpniecību.

Spēles plūsma - gājienu izpildes secība.

"Chaos in the Old World" - četru spēlētāju galda spēle, kas veidota Warhammer fantāzijas pasaulē.

Aktīvais spēlētājs - spēlētājs, kuram šobrīd ir atļauts veikt gājienu.

DPD - datu plūsmu diagramma.

PPS - programmatūras prasību specifikācija.

PPA - programmas projektējuma apraksts

XML - Extensive Markup language, nodrošina datu kodēšanu cilvēkiem un datoriem saprotamā veidā.

XML-RPC protokols - datu pārraides protokols, kas izmanto XML datu kodēšanai un http datu transportam.

1. PROGRAMMATŪRAS PRASĪBU SPECIFIKĀCIJA

1.1. Ievads

1.1.1. Nolūks

Programmas prasību specifikācija ir izstrādāta ar nolūku programmas "Galda spēles realizācija tiešsaistē" prasību aprakstīšanai. Prasību specifikācijā tiek raksturota funkcionalitāte un noformulētas prasības sistēmai.

1.1.2. Darbības sfēra

Izstrādājamā programma ir galda spēles "Chaos in the Old World" daļēja implementācija. Izstrādājamās programmas mērķis ir nodrošināt iespēju spēlēt šo galda spēli neklātienē. Projekta galvenais uzdevums ir nodrošināt spēlētāju gājienu apmaiņu tiešsaistē.

1.1.3. Saistība ar citiem dokumentiem

Noformējot dokumentu, ievērotas standarta LVS 68:1996 „Programmatūras prasību specifikācijas ceļvedis" prasības.

1.1.4. Dokumenta pārskats

Dokuments sastāv no piecām daļām:

1. Ievads, kas sastāv no dokumenta nolūka, mērķa un saistības ar citiem dokumentiem.
2. Vispārējs apraksts par produkta perspektīvu, funkcijām, lietotāja raksturiezīmēm un vispārējiem ierobežojumiem.
3. Izstrādājamās programmas funkciju apraksts.
4. Prasības programmas saskarnei.
5. Nefunkcionālās prasības.

1.2. Vispārējs apraksts

1.2.1. Produkta perspektīvas

Spēlētāju komunikācija ir svarīga spēles sastāvdaļa, tādēļ papildus ieteicams lietot tādu programmu kā Skype. Izstrādātajā programmā nav kauliņu mešanas funkcionalitātes, tādēļ nepieciešams izmantot kādu rīku, kas nodrošina godīgu kauliņu mešanu caur internetu, piemēram [6].

1.2.2. Produkta funkcijas

Izstrādājamās programmas galvenais uzdevums ir nodrošināt, galda spēles spēlēšanu attālināti. Šī mērķa paveikšanai ir nepieciešams visiem spēlētājiem nodrošināt vienotu informāciju par spēles elementiem.

Spēles galvenie elementi ir figūras un kārtis.

Programmas funkcionālās prasības tiek iedalītas četros moduļos:

Spēles inicializācijas modulis - Iegūst no lietotāja programmas darbināšanas parametrus un sasaista pārējos moduļus atbilstoši iegūtajiem parametriem.

Spēles loģikas modulis - Nodrošina spēles ritumu, veic gājienu validāciju un uztur informāciju par spēles elementiem un procesiem.

Grafikās saskarnes modulis - Nodrošina spēles informācijas attēlošanu lokālajam spēlētajam, saņem gājienus no lokālā spēlētāja.

Tīkla modulis - Nodrošina gājienu informācijas plūsmu starp spēlētājiem.

1.2.3. Lietotāja raksturiezīmes

Lai pilnvērtīgi lietotu izstrādāto programmu lietotājiem jāpārzina "Chaos in the Old World" noteikumi.

1.2.4. Vispārējie ierobežojumi

Programmas lietošanai ir nepieciešams ekrāns ar vismaz 1366x768 pikseļu izšķirtspēju, Java 1.7 un interneta pieslēgums.

1.3.Funkcionālās prasības

1.3.1. Spēles inicializācijas modulis

1.3.1.1. Parametru izvēle

Ievads

Lietotājam tiek uzdoti jautājumi par vēlamajiem spēles parametriem

Ievade

1. Lietotājs izvēlas vienu no spēles rasēm.

2. Lietotājs izvēlas, vai izveidot jaunu komunikācijas kanālu serverī, vai pievienoties jau eksistējošam kanālam.

3. Ja lietotājs vēlas pievienoties kanālam, lietotājam prasa norādīt kanāla identifikācijas numuru.

4. Lietotājam lūdz ievadīt kanāla paroli .

Apstrāde

Ievaddati tiek validēti

Izvade

Tiek izsaukta spēles inicializācijas funkcija ar norādītajiem parametriem.

1.3.1.2. Spēles inicializācija

Ievads

Palaiž spēles loģikas, grafikas un tīkla moduļus.

Ievade

Tiek saņemti lietotāja ievadītie parametri.

Apstrāde

Tiek izveidota spēles loģikas instance, kurā lietotājs kontrolē vienu no rasēm. Spēles loģikas instancei tiek norādīts kanāls, ar kuru komunicēt par notikumiem spēlē un citu spēlētāju darbībām. Tiek izveidota grafiskā saskarne, kas atspoguļo spēles elementus.

Izvade

Lietotājam tiek atgriezta spēles grafiskā saskarne.

1.3.2. Spēles loģikas modulis

1.3.2.1. Spēles plūsmas kontrole

Ievads

Kontrolē komandu saņemšanu un nosūtīšanu.

Ievade

Saņem komandu no aktīvā spēlētāja.

Ja gājiens ir lokālajam spēlētajām, tad komandas tiek saņemtas no grafiskās saskarnes un nosūtītas uz serveri ar tīkla moduļa palīdzību. Ja gājiens nav lokālajam spēlētājam, tad komandas tiek saņemtas no citiem spēlētājiem un izpildītas grafiskajā saskarnē.

Apstrāde

1. Tiek saņemta komandu no aktīvā spēlētāja.
2. Tiek veikta komandas validācija

Izvade

Tiek izsaukta funkcija saņemtās komandas izpildei.

1.3.2.2. Figūru inicializācija

Ievads

Tiek inicializētas 45 spēles figūras.

Ievade

Tiek ielasīti figūru definētie parametri.

Apstrāde

Katrai figūrai tiek izveidotas divas instances: loģiskā un grafiskā.

Izvade

Figūras tiek attēlotas grafiskajā saskarnē.

1.3.2.3. Kāršu inicializācija

Ievads

Tiek inicializētas 96 spēles kārtis. Katram spēlētājam no viņam piederošajām 24 kārtīm ir jāizdala 5 kārtis.

Ievade

Tiek ielasīti definētie parametri.

Apstrāde

Katram spēlētājam tiek piešķirtas 5 nejaušas kārtis, tiek izveidotas to loģiskās un grafiskās instances.

Izvade

Piešķirtās kārtis tiek attēlotas grafiskajā saskarnē.

1.3.2.4. Aktīvā spēlētāja nomaiņa

Ievads

Maina spēlētāju, kuram ļauts veikt gājienus.

Ievade

Funkcija tiek izsaukta, kad tiek saņemta komanda no lokālā spēlētāja, vai no servera par aktīvā spēlētāja nomaiņu.

Apstrāde

Spēles loģikā tiek mainīts aktīvais spēlētājs.

Izvade

Visiem spēlētājiem, izņemot aktīvajam, ir liegta gājienu veikšana

1.3.2.5. Komandu nosūtīšana

Ievads

Funkcija tiek izsaukta no spēlēs plūsmas kontroles, lai izpildītu pēdējo aktīvā spēlētāja komandu.

Ievade

Pēdējā aktīvā spēlētāja izpildītā komanda.

Informācija par aktīvo spēlētāju.

Apstrāde

Ja aktīvais spēlētājs ir lokālais spēlētājs, tad komanda tiek nosūtīta citiem spēlētājiem ar tīkla moduļa palīdzību.

Ja lokālajam spēlētājam nav gājiens, tad komanda tiek nosūtīta uz grafisko saskarni un izpildīta.

Izvade

Komanda tiek nosūtīta uz atbilstošo saskarni.

1.3.2.6. Komandu saņemšana

Ievads

Funkcija tiek izsaukta no spēlēs plūsmas kontroles, lai iegūtu informāciju par pēdējo aktīvā spēlētāja komandu.

Ievade

Informācija par aktīvo spēlētāju.

Apstrāde

Ja aktīvais spēlētājs ir lokālais spēlētājs, tad no grafiskās saskarnes tiek iegūta pēdējā izpildītā komanda.

Ja lokālajam spēlētājam nav gājiens, tad komanda tiek iegūta no servera.

Izvade

Tiek atgriezta pēdējā izpildītā komanda.

1.3.2.7. Komandu validācija

Ievads

Funkcija pārbauda, saņemto komandu

Ievade

Saņemtā komanda

Apstrāde

1. Tiek pārbaudīts, vai komandas formāts ir pareizs
2. Tiek pārbaudīts, vai komanda ir jau izpildīta.
3. Tiek pārbaudīts, vai komanda ir elementu kustināšana, vai aktīvā spēlētāja nomaiņa.
4. Tiek pārbaudīta komandas izpildes loģika.

Izvade

True- ja komanda ir derīga, False- ja nē.

1.3.3. Grafiskās saskarnes modulis

1.3.3.1. Elementu attēlošana

Ievads

Attēlo spēles interaktīvos elementus.

Ievade

No spēlēs loģikas moduļa tiek saņemts saraksts ar spēles elementiem un to parametriem.

Apstrāde

1. Tiek ielādēti elementi atbilstoši to parametriem
2. Elementiem tiek ielādētas tiem atbilstošas bildes
3. Elementiem tiek piesaistīta pārvietošanas funkcija

Izvade

Elementi tiek attēloti grafiskajā saskarnē.

1.3.3.2. Elementu pārvietošana

Ievads

Nodrošina elementu kustināšanu spēles laukumā. Nospiežot kreiso pogu un velkot lietotājs, var kustināt spēles elementus. Atlaižot nospiesto pogu, spēles loģika mēģinās pārvietot elementu uz norādīto lauciņu.

Ievade

1. Lokācija, kurā lietotājs nospieda peles kreiso pogu.
2. Lokācija, kurā lietotājs atlaida peles kreiso pogu.

Apstrāde

1. Tiek pārbaudīts, vai lietotājs drīkst kustināt elementus.
2. Spēles loģikas modulis pārbauda gājienu.

Izvade

Izvēlētais elements tiek pārvietos grafiskajā saskarnē

1.3.3.3. Kontroles panelis

Ievads

Sniedz informāciju par spēles gaitu. Tiek attēlota informācija par to, kuram spēlētājam ir gājiens, sniedz iespēju beigt savu gājienu.

Ievade

Poga, kas ļauj lietotājam beigt savu gājienu.

Apstrāde

1. Tiek pārbadīts, vai lietotājs ir aktīvais spēlētājs
2. Komanda tiek nosūtīta spēles loģikas modulim

Izvade

1. Informācija par aktīvo spēlētāju.
2. Informācija par lietotāja izvēlēto rasi.

1.3.4. Tīkla modulis

1.3.4.1. Komunikācijas kanāla izveide

Ievads

Nosūta pieprasījumu uz serveri ar pieprasījumu, piešķirt jaunu kanālu.

Ievade

Kanāla parole

Apstrāde

Uz servera tiek izdalīts jauns kanāls ar lietotāja izvēlēto paroli.

Izvade

Kanāla identifikācijas numurs

1.3.4.2. Pieslēgšanās komunikācijas kanālam

Ievads

Nosūta uz serveri pieprasījumu pieslēgties kanālam

Ievade

1. Kanāla identifikācijas numurs
2. Kanāla parole

Apstrāde

1. Tiek pārbaudīts, vai eksistē šāds kanāls
2. Tiek pārbaudīts, vai parole ir pareiza

Izvade

Atbilde par pieslēgšanās mēģinājuma sekmīgumu

1.3.4.3. Ziņu nosūtīšana

Ievads

Nosūta ziņu uz servera norādīto kanālu.

Ievade

Komanda no spēles loģikas moduļa.

Kanāla pieslēgšanās parametri.

Apstrāde

Komanda tiek pārveidota uz tekstu. Ziņa tiek nosūtīta uz norādīto kanālu.

Ziņa tiek saglabāta kā kanāla pēdējā saņemtā ziņa.

Izvade

Atbilde par ziņas nosūtīšanas sekmīgumu.

1.3.4.4. Ziņu saņemšana

Ievads

Iegūst pēdējo ziņu no norādītā kanāla.

Ievade

Kanāla pieslēgšanās parametri.

Apstrāde

Tiek iegūta pēdējā ziņa kanālā. Iegūtā ziņa tiek pārveidota uz spēles loģikas modulim saprotamu komandu.

Izvade

Ziņa tiek nosūtīta spēles loģikas modulim.

1.3.5. Servera prasības

Serverim ir jāspēj saņemt un atgriezt ziņas String formātā. Jārealizē saņemto ziņu iedalīšana pēc kanāliem. Kanālus jāidentificē ar skaitlisku kodu, tiem jābūt aizsargātiem ar paroli. Spēles lietotājiem jābūt iespējai pēc vajadzības definēt jaunus kanālus, norādot to paroli. Kanālos jāglabā viena ziņa - pēdējā saņemtā.

1.4. Ārējās saskarnes prasības

1.4.1. Lietotāja saskarne

- Saskarnei ir jābūt angļu valodā.
- Attēlotajiem spēles elementiem ir jābūt vizuāli līdzīgiem, kā galda spēlē "Chaos in the Old World".
- Ir jāuztur atskaite par lietotāju veiktajiem gājieniem spēles konsolē
- Lietotājs ir jāinformē, vai izvēlētais gājiens ir izpildījies. Ja gājiens nav veiksmīgi izpildījies ir jāinformē lietotājs kāpēc.

1.4.2. Aparatūras saskarne

Izstrādātajai programmai jābūt pārskatāmai 1366x768 pikseļu izšķirtspējā.

1.5. Nefunkcionālās prasības

1.5.1. Veiktspējas prasības

Spēles klientam jāspēj īstenot citu spēlētāju veiktās darbības trīs sekunžu laikā.

1.5.2. Drošība

Spēles klientu nosūtīto komandu saturs nav īpaši jāaizsargā, jo tas visiem spēlētājiem ir brīvi pieejams. Komunikācijas kanālam ir jābūt aizsargātam ar paroli.

1.5.3. Pārnesamība

Programmai jābūt viegli pārnesamai un uzstādāmai uz citiem datoriem.

2. PROGRAMMATŪRAS PROJEKTĒJUMA APRAKSTS

2.1. Ievads

Šīs nodaļas nolūks ir aprakstīt projekta "Galda spēles realizācija tiešsaistē" projektējumu. Programma tiek izstrādāta izmantojot Java objektorientēto pieeju. Nodaļā ir veikta PPS specificēto moduļu dekompozīcija, to funkciju savstarpējo atkarību apraksts, izstrādātas atbilstošās datu plūsmu diagrammas. Moduļiem ir veikts detalizēts projektējums, aprakstot nepieciešamās klases un to metodes moduļu funkciju realizācijai.

2.2.Moduļu dekompozīcija

Šīs nodaļas uzdevums ir aprakstīt realizēto moduļu un to funkcijas dekompozīciju. Moduļu savstarpējā datu apmaiņa ilustrēta nodaļā 2.3.2 .

Identificējums: 1. Spēles inicializācijas modulis

Funkcijas: 1.1 Parametru izvēle, 1.2 Spēles inicializācija

Atkarības: Nav

Identificējums: 2.Spēles loģikas modulis

Funkcijas: 2.1 Spēles plūsmas kontrole , 2.2 Kāršu inicializācija, 2.3 Figūru inicializācija, 2.4 Aktīvā spēlētāja nomaiņa, 2.5 Komandu nosūtīšana, 2.6 Komandu saņemšana, 2.7 Komandu validācija.

Atkarības: 1. Spēles inicializācijas modulis 3. Grafiskās saskarnes modulis, 4. Tīkla modulis

Identificējums: 3.Grafiskās saskarnes modulis

Funkcijas: 3.1 Elementu attēlošana , 3.2 Elementu pārvietošana, 3.3 Kontroles panelis

Atkarības: 1. Spēles inicializācijas modulis 2. Spēles loģikas modulis

Identificējums: 4.Tīkla modulis

Funkcijas: 4.1 Komunikācijas kanāla izveide , 4.2 Pieslēgšanās komunikācijas kanālam, 4.3 Ziņu nosūtīšana, 4.4 Ziņu saņemšana

Atkarības: 1. Spēles inicializācijas modulis 2. Spēles loģikas modulis

2.2.1. Spēles Inicializācijas moduļa dekompozīcija

Moduļa dekompozīcija ilustrēta nodaļā 2.3.3.1.

Identificējums: 1.1 Parametru izvēle

Nodrošina prasību: 1.3.1.1

Atkarības: Nav

Identificējums: 1.2 Spēles inicializācija

Nodrošina prasību: 1.3.1.2

Atkarības: 1.1 Parametru izvēle

2.2.2. Spēles loģikas moduļa dekompozīcija

Moduļa dekompozīcija ilustrēta nodaļā 2.3.3.2

Identificējums: 2.1 Spēles plūsmas kontrole

Nodrošina prasību: 1.3.2.1

Atkarības: 1. Spēles inicializācijas modulis, 2.4 Aktīvā spēlētāja nomaiņa, 2.5 Komandu nosūtīšana, 2.6 Komandu saņemšana

Identificējums: 2.2 Figūru inicializācija

Nodrošina prasību: 1.3.2.2

Atkarības: 1. Spēles inicializācijas modulis,

Identificējums: 2.3 Kāršu inicializācija

Nodrošina prasību: 1.3.2.3

Atkarības: 1. Spēles inicializācijas modulis

Identificējums: 2.4 Aktīvā spēlētāja nomaiņa

Nodrošina prasību: 1.3.2.4

Atkarības: 2.1 Spēles plūsmas kontrole

Identificējums: 2.5 Komandu nosūtīšana

Nodrošina prasību: 1.3.2.5

Atkarības: 2.1 Spēles plūsmas kontrole

Identificējums: 2.6 Komandu saņemšana

Nodrošina prasību: 1.3.2.6

Atkarības: 2.7 Komandu validācija

Identificējums: 2.7 Komandu validācija

Nodrošina prasību: 1.3.2.7

Atkarības: 3.Grafiskās saskarnes modulis 4. Tīkla modulis

2.2.3. Grafiskās saskarnes moduļa dekompozīcija

Moduļa dekompozīcija ilustrēta nodaļā 2.3.3.3

Identificējums: 3.1 Elementu attēlošana

Nodrošina prasību: 1.3.3.1

Atkarības: 2. Spēles loģikas modulis

Identificējums: 3.2 Elementu pārvietošana

Nodrošina prasību: 1.3.3.2

Atkarības: 2. Spēles loģikas modulis, 3.1 Elementu attēlošana

Identificējums: 3.3 Kontroles panelis

Nodrošina prasību: 1.3.3.3

Atkarības: 2. Spēles loģikas modulis

2.2.4. Tīkla moduļa dekompozīcija

Moduļa dekompozīcija ilustrēta nodaļā 2.3.3.4

Identificējums: 4.1 Komunikācijas kanāla izveide

Nodrošina prasību: 1.3.4.1

Atkarības: 1. Spēles inicializācijas modulis, 2. Spēles loģikas modulis, Serveris

Identificējums: 4.2 Pieslēgšanās komunikācijas kanālam

Nodrošina prasību: 1.3.4.2

Atkarības: 1. Spēles inicializācijas modulis, 4.1 Komunikācijas kanāla izveide, Serveris

Identificējums: 4.3 Ziņu nosūtīšana

Nodrošina prasību: 1.3.4.3

Atkarības: 2. Spēles loģikas modulis, 4.2 Pieslēgšanās komunikācijas kanālam

Identificējums: 4.4 Ziņu saņemšana

Nodrošina prasību: 1.3.4.4

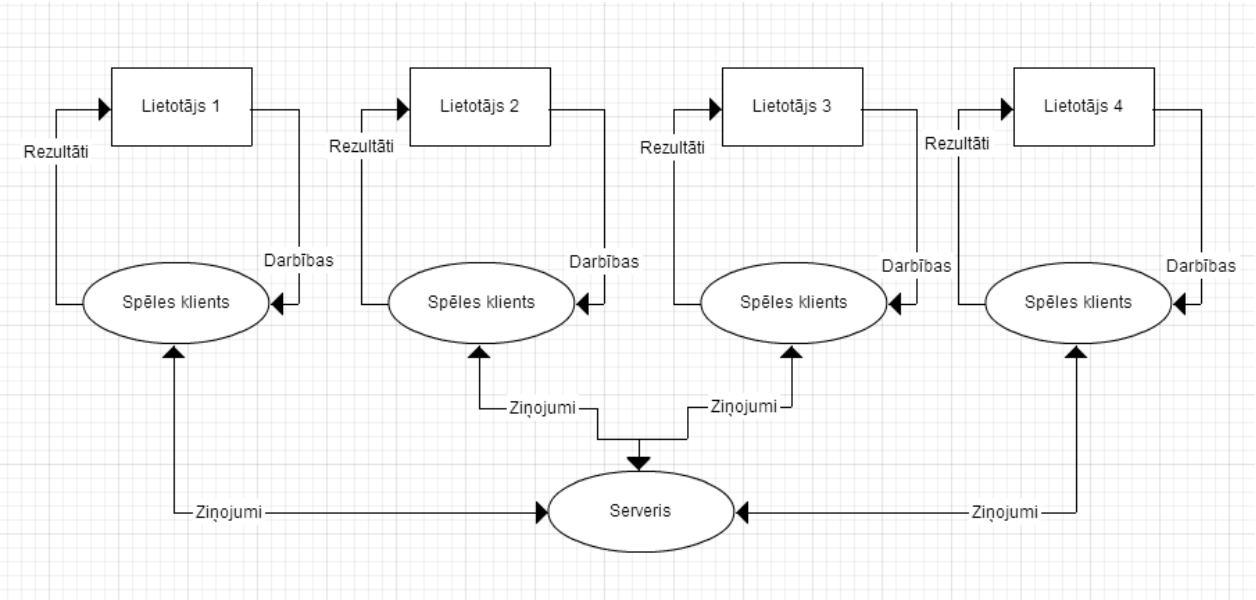
Atkarības: 2. Spēles loģikas modulis, 4.2 Pieslēgšanās komunikācijas kanālam

2.3.Datu plūsmu diagrammas

Šajā nodaļa ir vizualizēti izstrādātās programmas procesi un modeļu mijiedarbība, izmantojot datu plūsmu diagrammas (DPD).

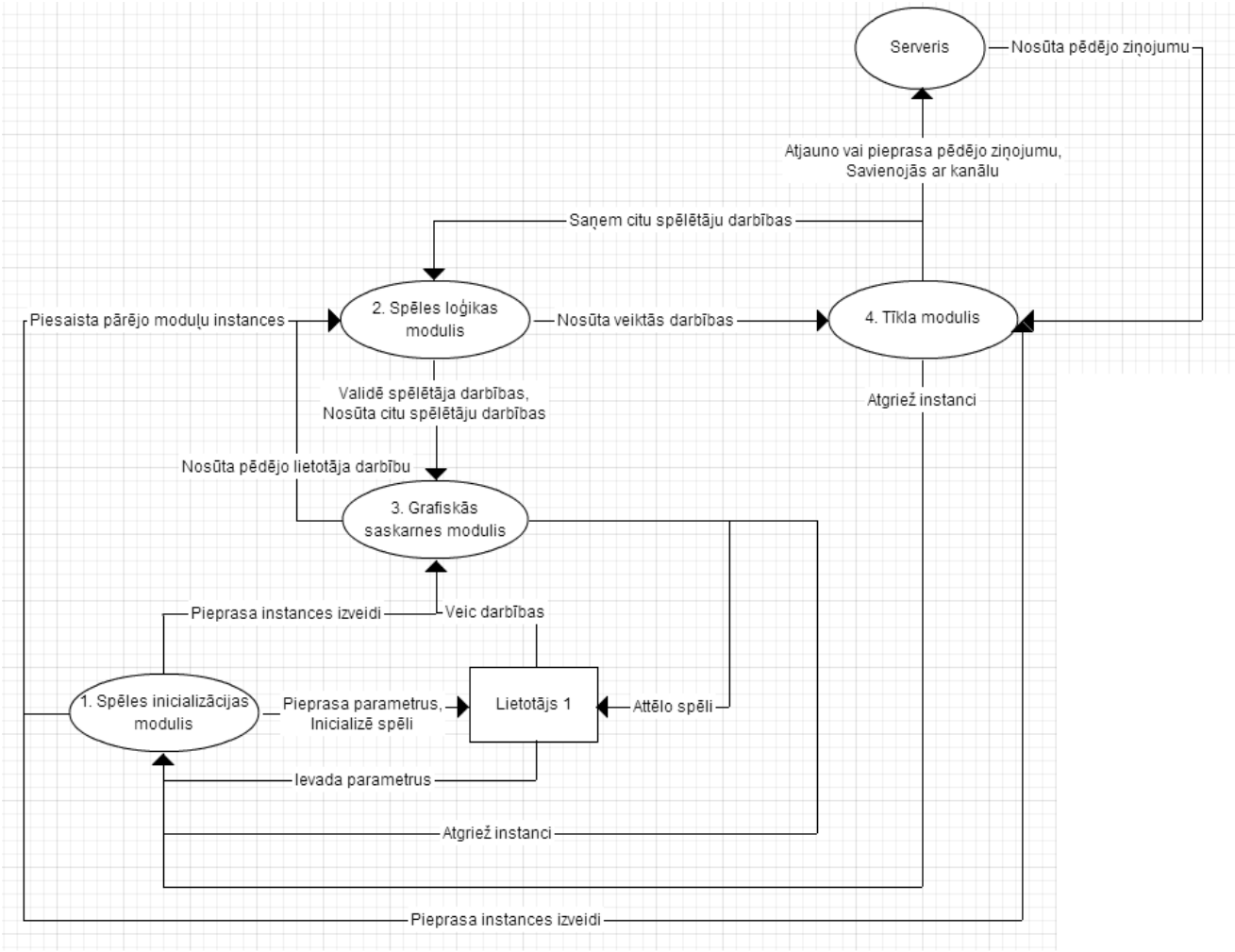
2.3.1. DPD 0. Līmenis

Nulles līmeņa DPD paskaidro programmas klientu un servera datu apmaiņas procesu.



2.3.2. DPD 1. Līmenis

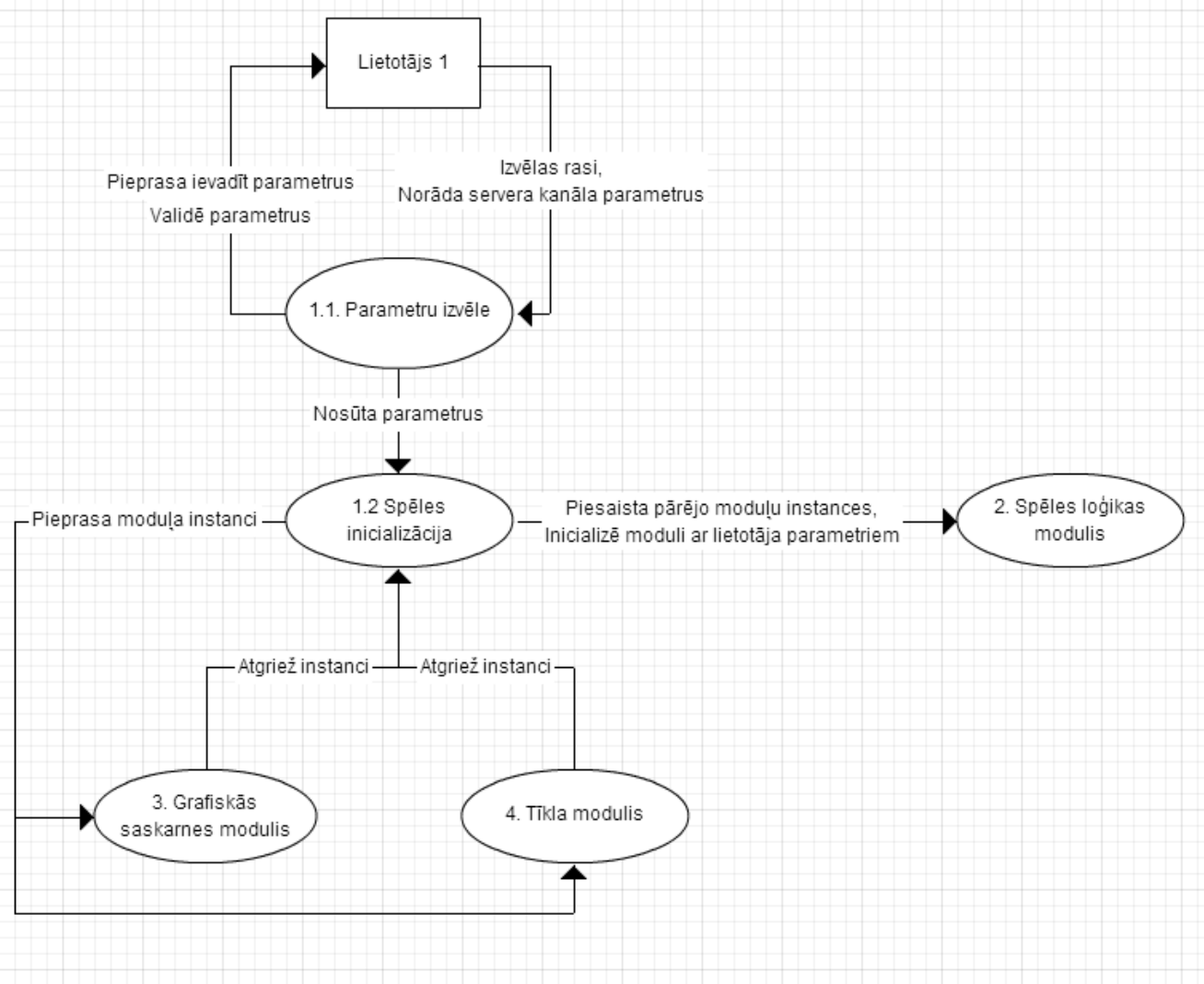
Pirmā līmeņa DPD apraksta programmas moduļu mijiedarbību.



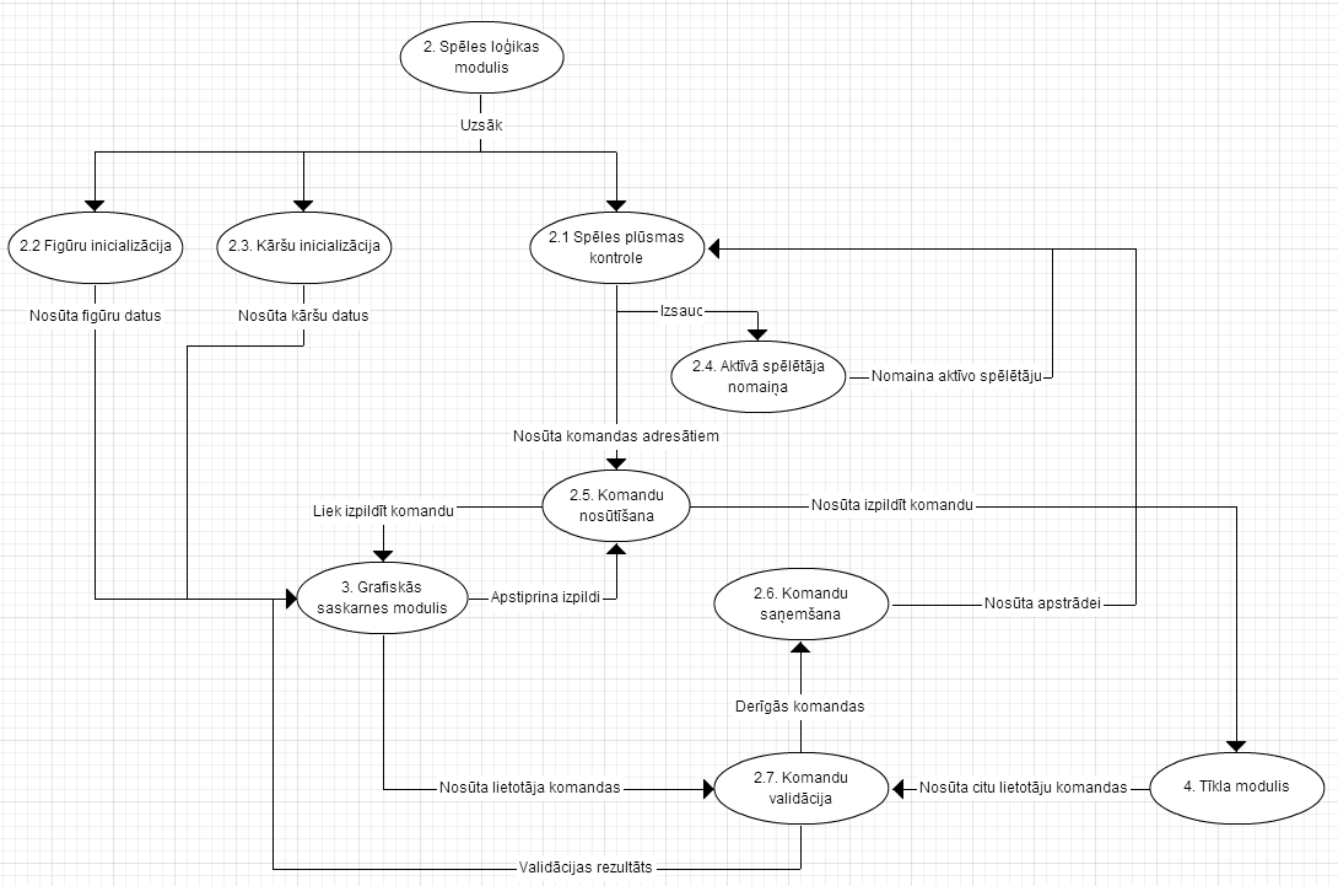
2.3.3. DPD 2. Līmenis

Otrā līmeņa DPD apraksta moduļu funkciju mijiedarbību.

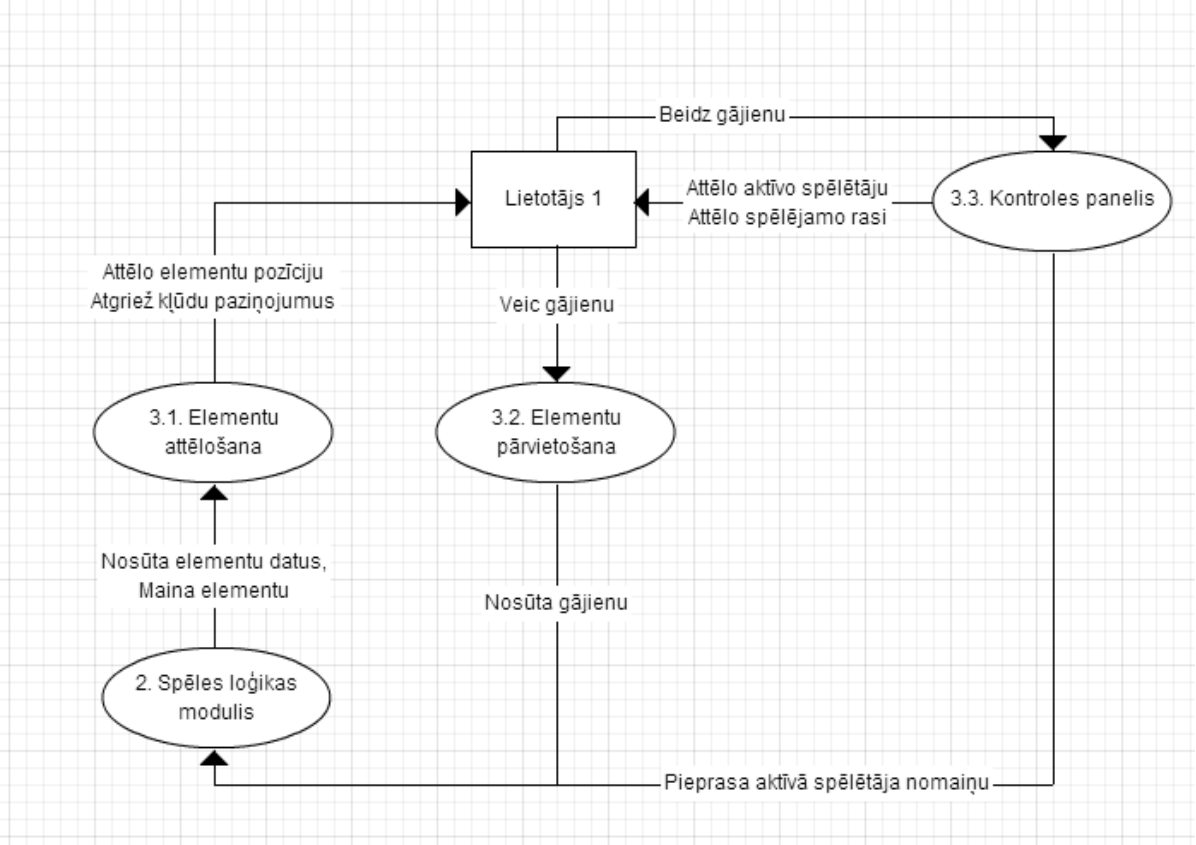
2.3.3.1. Spēles inicializācijas modulis



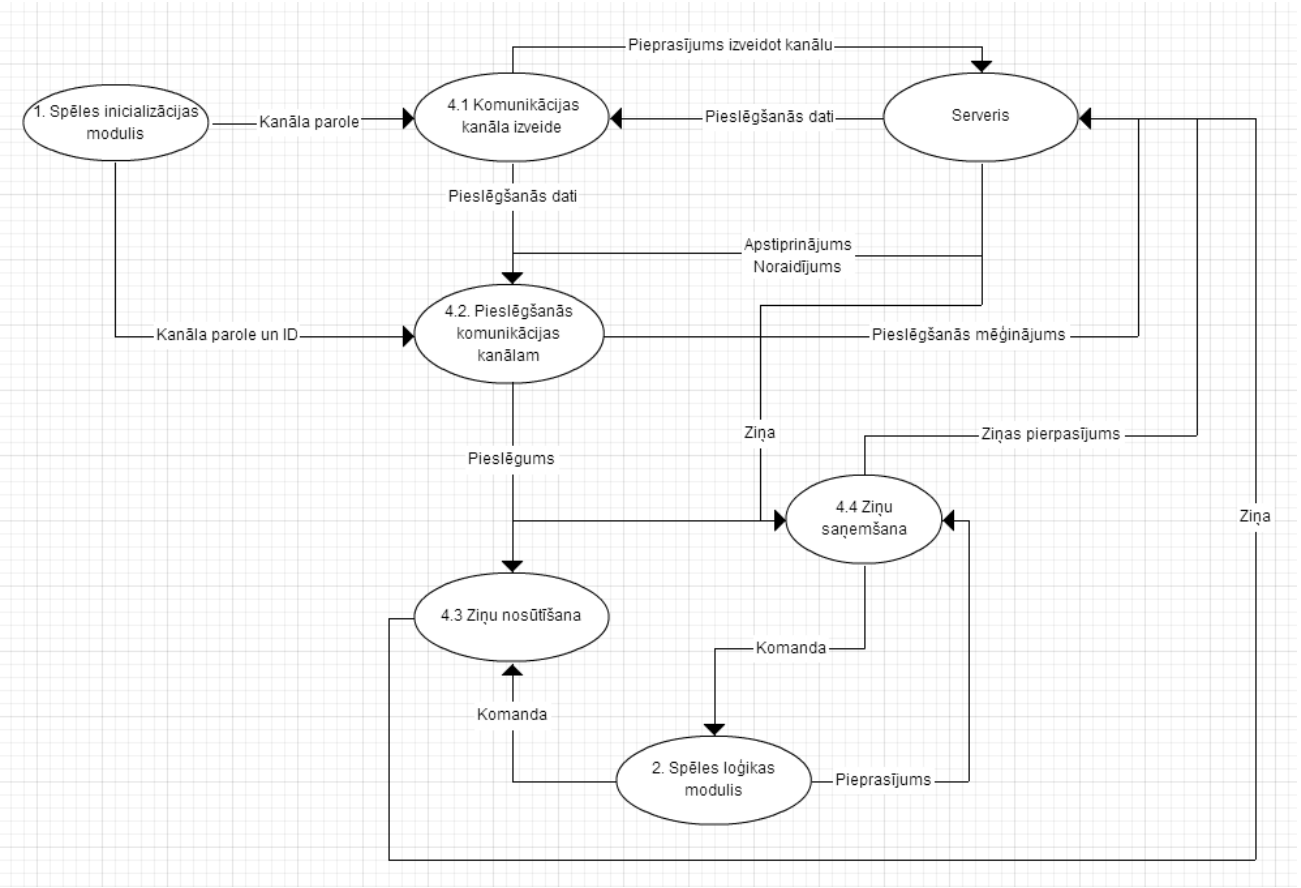
2.3.3.2. Spēles loģikas modulis



2.3.3.3. Grafiskās saskarnes modulis



2.3.3.4. Tīkla modulis



2.4.Moduļu detalizēts projektējums

2.4.1. Spēles Inicializācijas modulis

2.4.1.1. Klase Main

Apraksts: Palaiž main funkciju un veic spēles inicializāciju atbilstoši spēlētāja norādītajiem parametriem

Realizē funkcijas: 1.1 Parametru izvēle, 1.2 Spēles inicializācija

Metodes:

Tips	Nosaukums	Apraksts
String	ask(String question, String options[ ])	Uzdod lietotājam jautājumu, norādot atbilžu variantus
String	ask(String question)	Uzdod lietotājam jautājumu.

Mainīgie:

Tips	Nosaukums	Apraksts
int	playerChoice	Izvēlētā rase
int	channelChoice	Pieslēgties kanālam, vai izveidot jaunu kanālu
String	gameIdOnServer	Kanāla identifikācijas numurs
String	gamePassword	Kanāla parole
BoardGame	boardGame	Spēles loģikas moduļa instance
IPlayerHandler	localHandler	Grafiskās saskarnes instance
IPlayerHandler	networkHandler	Tīkla moduļa instance

2.4.2. Spēles loģikas modulis

2.4.2.1. Klase BoardGame

Apraksts: Nodrošina spēles loģiku un plūsmu

Realizē funkcijas: 2.1 Spēles plūsmas kontrole , 2.2 Kāršu inicializācija, 2.3 Figūru inicializācija, 2.4 Aktīvā spēlētāja nomaiņa, 2.5 Komandu nosūtīšana, 2.6 Komandu saņemšana

Metodes:

Tips	**Nosaukums**	**Apraksts**
konstruktors	boardGame( )	Veic kāršu un figūru inicializāciju,
Void	setHandlers(int race, IPlayerHandler localHandler, IPlayerHandler networkHandler)	Pievieno spēles kontrolierus to atbilstošajām rasēm. localHandler - grafiskās saskarnes instance networkHandler - Tīkla moduļa instance
void	startGame( )	Uzsāk spēles plūsmu. Tiek veikta spēlētāju kontrolieru sasaiste. Ja trūkst kontrolieru, funkcija nogaida 5 sekundes.
void	swapActivePlayer( )	Nomaina aktīvo spēlētāju.
void	waitForMove( )	Saņem gājienu no aktīvā kontroliera izsaucot atbilstošā kontroliera metodi getMove(), iegūto gājienu validē ar metodi isMoveValid(move).
void	createAndAddElement int race, int type, int row, int column)	Izveido elementu un piesaista to elementu sarakstam
boolean	moveElement(Move move)	Veic gājiena loģisko pārvietošanu Atgriež true, ja elements veiksmīgi pakustināts
boolean	isGameEndConditionReached( )	Pārbauda, vai nav sasniegts spēles beigu

		nosacījums.
Element	getElementAtLocation(int row, int column)	Atgriež elementu, kas atrodas norādītajā rindā un kolonnā
int	getGameState( )	Atgriež aktīvā spēlētāja identifikatoru
List<Element>	getElements( )	Atgriež elementu sarakstu
void	addCard(race)	Piešķir norādītajam spēlētājam nejaušu kārti, no atbilstošās rases kāršu saraksta
void	createCards( )	Inicializē kāršu datus

Mainīgie:

Tips	**Nosaukums**	**Apraksts**
int	localPlayerRace	Lokālā spēlētāja rase
Stirng	lostMoveStr	Pēdējā kustība
List <Element>	elements	Saraksts ar spēles loģiskajiem elementiem
List <Element>	cards	Spēles kāršu saraksts
List <Element>	cardsKhorne	Saraksts ar Khorne spēlētāja aktīvajām kārtīm
List <Element>	cardsNurlge	Saraksts ar Nurlge spēlētāja aktīvajām kārtīm
List <Element>	cardsTzeentch	Saraksts ar Tzeentch spēlētāja aktīvajām kārtīm
List <Element>	cardsSlaanesh	Saraksts ar Slaanesh spēlētāja aktīvajām kārtīm
MoveValidator	moveValidator	Kustību validātora instance
IPlayerHandler	khronePlayerHandler	Khorne spēlētāja kontrolieris
IPlayerHandler	nurglePlayerHandler	Nurgle spēlētāja kontrolieris
IPlayerHandler	tzeentchPlayerHandler	Tzeentch spēlētāja kontrolieris
IPlayerHandler	slaaneshPlayerHandler	Slaanesh spēlētāja kontrolieris
IPlayerHandler	localPlayerHandler	Grafiskās saskarnes instance
IPlayerHandler	networkPlayerHandler	Tīkla moduļa instance
IPlayerHandler	activePlayerHandler	Aktīvā spēlētāja spēles kontrolieris

2.4.2.2. Klase Element

Apraksts: Klasē glabājas informācija par kādu spēles elementu - figūru vai kārti

Realizē metodes: 2.2 Kāršu inicializācija, 2.3 Figūru inicializācija

Metodes:

Tips	**Nosaukums**	**Apraksts**
String	getRaceText(int race)	Pārveido rases identifikatoru par rases nosaukumu String formā.
konstruktors	Element(int race, int cost, String name, int type)	Konstruktors elementiem, kas neatrodas laukumā, piemēram kāršu sarakstam.
konstruktors	Element(int race, int cost, String name, int type, int row, int column)	Konstruktors ar pilnu parametru sarakstu
konstruktors	Element(int race, int type, int row, int column)	Konstruktors ar minimālu parametru daudzumu

Mainīgie:

Tips	**Nosaukums**	**Apraksts**
int	race	Definē elementa piederību
int	type	Definē elementa tipu (1-13). Tips nosaka, kura kārts vai figūra elements ir.
int	row	Laukuma rinda, kurā elements atrodas
int	column	Laukuma kolonna, kurā elements atrodas
int	cost	Elementa cena
int	name	Elementa nosaukums
boolean	used	Definē, vai elements tiek lietots

2.4.2.3. Saskarne IPlayerHandler

Apraksts: IPlayerHandler saskarnē (interface) ir definēti metožu prototipi, kurus izmanto spēles loģikas modulis, komunikācijā ar Grafiskās saskarnes un Tīkla moduļiem.

Realizē funkcijas: 2.4 Aktīvā spēlētāja nomaiņa, 2.5 Komandu nosūtīšana, 2.6 Komandu saņemšana

Metodes:

Tips	**Nosaukums**	**Apraksts**
Move	getMove( )	Atgriež pēdējo veikto gājienu
void	moveSuccessfullyExecuted(Move move)	Nosūta validētu gājienu izpildei
void	endturn( )	Nosūta pavēli beigt gājienu
void	clearLastMove()	Izdzēš atskaiti par pēdējo veikto gājienu

2.4.2.4. Klase Move

Apraksts: Klasē tiek definēti kustības parametri

Realizē funkcijas: 2.4 Aktīvā spēlētāja nomaiņa, 2.5 Komandu nosūtīšana, 2.6 Komandu saņemšana

Metodes:

Tips	**Nosaukums**	**Apraksts**
boolean	isMoveValid(Move move, boolean output)	Pārbauda, vai kustība ir korekta. Parametrs output norāda, vai veikt izdruku par saņemtajām kļūdām. Atgriež true, ja kustība ir korekta. Funkcijai jāpārbauda, vai elements netiek kustināts uz jau aizņemtu lauciņu, vai elements netiek kustināts uz savu lauciņu, vai norādītās kustības sākuma lauciņā vispār atrodas elements.
void	print( )	Izdrukā paziņojumu, ja izdruka ir iespējota

Mainīgie:

Tips	**Nosaukums**	**Apraksts**
int	sourceRow	Elementa atrašanās vietas rinda
int	souceColumn	Elementa atrašanās vietas kolonna
int	targetRow	Kustības mērķa rinda
int	targetColumn	Kustības mērķa kolonna
boolean	endTurn	True, ja klases instance tiek izmantota, lai beigtu gājienu

2.4.2.5. Klase MoveValidator

Apraksts: Veic gājienu pārbaudi

Realizē funkcijas: 2.7 Komandu validācija.

Metodes:

Tips	Nosaukums	Apraksts
konstruktors	Move(int sourceRow, int sourceColumn, int targetRow, int targetColumn)	Konstruktors elementu kustināšanai. Satur kustības sākuma rindu, kolonnu un beigu rindu, kolonnu.
konstruktors	Move(boolean endTurn)	Konstruktors, izveido klases instanci, kura informē, par gājiena beigām.
String	convertMoveToString(Move move)	Pārvērš klases Move instances parametrus par tekstu. Elementa kustības dati jāpārveido formātā "SRSC-MRMK" , kur SR - sākuma rinda, SC - sākuma kolonna, MR - mērķa rinda, MK - mērķa kolonna. Piemēram "0107-0406". Ja tiek saņemta Move instance, kas norāda, ka jāmaina aktīvais spēlētājs, jāatgriež vērtība "end_turn".
void	convertStringToMove(String input)	Pārvērš teksta komandu par klases Move instanci, pēc līdzīga principa kā definēts metodē convertMoveToString( ).

Mainīgie:

Tips	Nosaukums	Apraksts
BoardGame	boardGame	Spēles loģikas moduļa instance
Element	sourceElement	Elements, kurš tiek kustināts
Element	targetElement	Elements, kurš atrodas kustības mērķa laukumā
boolean	output	Nosaka, vai veikt izdruku par veiktajām pārbaudēm

2.4.3. Grafikās saskarnes modulis

2.4.3.1. Klase ElementsDragAndDropListener

Apraksts: Nodrošina elementu kustināšanu spēles laukumā fiksējot spēlētāja peles darbības

Realizē funkcijas: 3.2 Elementu pārvietošana

Metodes:

Tips	**Nosaukums**	**Apraksts**
konstruktors	ElementsDragAndDropListener(List<GuiElement> guiElements, SwingPlayerHandler boardGui)	Izveido klases instanci un piesaista to, pie norādītās grafiskās saskarnes un tās elementiem.
void	mousePressed (MouseEvent evt)	Apstrādā peles pogas nospiešanu. Funkcija meklē elementu, virs kura atrodas peles kursors.
void	mouseReleased(MouseEvent evt)	Apstrādā peles pogas atlaišanu. Atlaižot peles pogu funkcija izrēķina paceltā elementa jauno atrašanās vietu, saglabā kustības datus kā klases Move instanci
void	mouseDragged(MouseEvent evt)	Pārzīmē izvēlēto elementu zem peles kursora.
boolean	mouseOverElement(GuiElement guiElement, int x, int y)	Atgriež true, ja peles kursors norāda uz šo elementu.

Mainīgie:

List<GuiElement>	guiElements	Spēles grafiskie elementi
SwingPlayerHandler	boardGui	Spēles grafiskās saskarnes instance
int	dragOffsetX	Novirze elementu vilkšanai
int	dragOffsetY	Novirze elementu vilkšanai

2.4.3.2. Klase GuiElement

Apraksts: Satur informāciju par elementu grafisko attēlošanu

Realizē funkcijas: 3.1 Elementu attēlošana

Metodes:

Tips	**Nosaukums**	**Apraksts**
konstruktors	GuiElement(Image img, Element element)	Piesaista Element Klases instancei attēlu, uzzīmē elementu laukumā.
String	toString( )	Atgriež informāciju par elementa x un y koordinātām String formātā
void	resetToUnderlyingElementPosition()	Uzzīmē pozīcijā, kas atbilst tā rindai un kolonnai. Metode tiek izsaukta pēc elementa pārvietošanas, lai elements tiktu pārzīmēts, tā jaunajā pozīcijā.
Element	getElement( )	Atgriež piesaistīto loģiskā elementa instanci

Mainīgie:

Image	img	Elementa attēls
int	x	Elementa x koordināta
int	y	Elementa y koordināta
Element	element	Elementa loģiskie dati

2.4.3.3. Klase SwingPlayerHandler

Apraksts: Nodrošina programmas grafisko saskarni.

Realizē funkcijas: 3.1 Elementu attēlošana , 3.2 Elementu pārvietošana, 3.3 Kontroles panelis

Metodes:

Tips	**Nosaukums**	**Apraksts**
konstruktors	SwingPlayerHandler(BoardGame boardGame, int localPlayer)	Izveido grafiskās saskarnes instanci, kura ir piesaistīta norādītajam spēles loģikas modulim.
String	getGameStateAsText( )	Atgriež informāciju par aktīvo spēlētāju
void	createAndAddGuiElement(Element Element)	Izveido loģiskā elementa grafisko instanci
Image	getImageForElement(int race, int type)	Atgriež elementa bildi atkarībā no tā tipa un rases
void	paintComponent(Graphics g)	Nodrošina laukuma un elementu zīmēšanas funkcionalitāti
int	convertColumnToX	Pārvērš kolonnas vērtību x koordinātā
int	convertRowToY	Pārvērš rindas vērtību y koordinātā
int	convertXToColumn	Pārvērš x koordinātu kolonnā
int	convertYToRow	Pārvērš y koordinātu rindā
void	setNewElementLocation(GuiElement dragElement, int x, int y)	Pārvieto elementu uz norādīto pozīciju
Move	getMove()	Ar 100 mili sekunžu pauzēm veic pārbaudi, vai spēlētājs ir veicis jaunu kustību.
void	moveSuccessfullyExecuted(Move move)	Pārbauda, vai elements ir veiksmīgi pārvietots
void	endturn()	Nomaina aktīvo spēlētāju
GuiElement	getGuiElementAt(int row, int column)	Atgriež elementa instanci, kas atrodas norādītajā pozīcijā
Void	clearLastMove( )	Notīra pēdējo veikto kustību

Mainīgie:

BoardGame	boardGame	Spēles loģikas modulis
JLabel	lblGameState	Attēlo aktīvā spēlētāja rasi
JLabel	lblPlayingAs	Attēlo informāciju par rasi, ar kuru spēlē lokālais spēlētājs
Image	imgBackground	Spēles saskarnes fons
List<Element>	guiElements	Spēles elementu saraksts
GuiElement	dragElement	Elements, kurš tiek kustināts
Move	currentMove	Pēdējā izpildītā darbība
boolean	isMovingEnabled	Nosaka elementu kustināšanas iespējošanu

2.4.4. Tīkla modulis

2.4.4.1. Klase NetworkPlayerHandler

Apraksts: Nodrošina ziņu nosūtīšanu un saņemšanu no tiešsaistes spēlētājiem

Realizē funkcijas: 4.1 Komunikācijas kanāla izveide , 4.2 Pieslēgšanās komunikācijas kanālam, 4.3 Ziņu nosūtīšana, 4.4 Ziņu saņemšana

Metodes:

Tips	**Nosaukums**	**Apraksts**
konstruktors	NetworkPlayerHandler(String aGameIdOnServer, String aGamePassword)	Izveido savienojumu ar jaunu kanālu, vai pievienojas jau eksistējošam
Move	getMove( )	Pēc metodes izsaukšanas katras 2 sekundes tiek iegūta ziņa no servera līdz saņemtā ziņa ir derīga, ziņu atgriež kā Move klases instanci.
void	moveSuccessfullyExecuted(Move move)	Pārliecinās par kustības veiksmīgu izpildi
void	endTurn( )	Nosūta gājiena beigšanas ziņu
String	getLastMove(String channelID, String channelPassword)	Atgriež pēdējo ziņu no servera
String	sendMove(String channelID, String channelPassword, String message)	Nosūta ziņu teksta formātā
boolean	isConnectionValid(String channelID, String channelPassword)	Pārbauda pieslēgšanās parametru derīgumu
String	createChannel(String channelPassword)	Izveido jaunu kanālu un atgriež kanāla ID.

Mainīgie:

String	gameIdOnServer	Kanāla ID
String	gamePassword	Kanāla parole
Sring	lastMoveStrReceivedFromNetwork	Pēdējā saņemtā ziņa
String	lastMoveStrSentToNetwork	Pēdējā nosūtītā ziņa
XmlRpcClient	XmlRpcClient	Xml-rpc protokola datu pārraides klients

2.4.5. Spēles servera projektējums

2.4.5.1. Tabula channels

Apraksts:
Glabā datus par izveidotajiem kanāliem

Tips	**Nosaukums**	**Apraksts**
int (Primary Key)	channel_Id	Kanāla identifikācijas numurs
String	channelPassword	Kanāla parole
datetime	creationDate	Kanāla izveides datums

2.4.5.2. Tabula messages

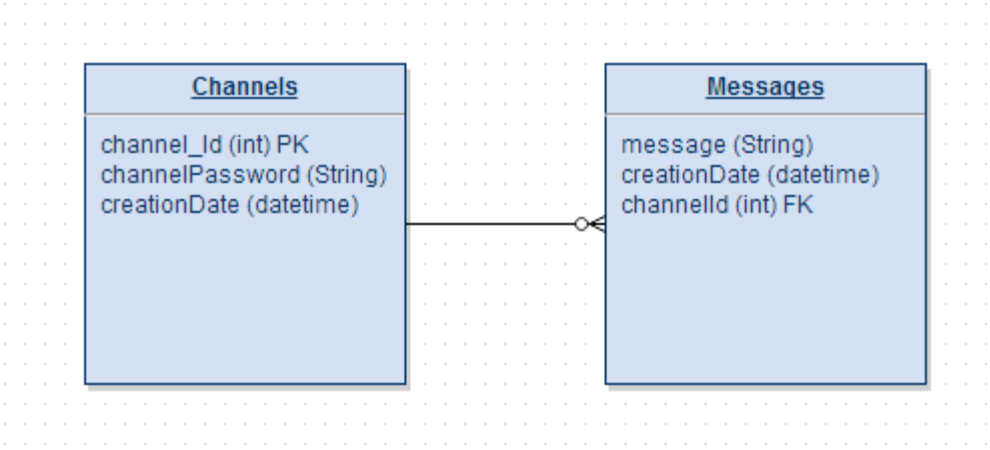
Apraksts:
Glabā saņemtās ziņas

Tips	**Nosaukums**	**Apraksts**
String	message	Saņemtā ziņa
datetime	creationDate	Ziņas nosūtīšanas datums
int (Foreign Key)	channelId	Kanāla identifikācijas numurs

2.4.5.3. Servera funkcijas

Tips	Nosaukums	Apraksts
String	createChannel(String password)	Izveido jaunu kanālu ar norādīto paroli. Atgriež izveidotā kanāla Identifikācijas numuru kā String vērtību.
boolean	validateConnection(String channelId, String password)	Veic kanāla pieslēgšanās parametru pārbaudi. Atgriež true, ja parametri ir korekti
String	getMessage(String channelId, String password)	Atgriež jaunāko ziņu, kas glabājas norādītajā kanālā.
boolean	storeMessage(String message, String channelId, String password)	Izveido jaunu ierakstu tabulā messages atbilstoši ievaddatiem. Veiksmīgas datus iespraušanas gadījumā atgriež true. Atgriež false, ja ievaddati ir bijuši nederīgi.
void	clearOldEntries( )	Funkcija dzēš visus ierakstus par kanāliem un ziņām, kas ir vecāki par noteiktu laika apjomu. Noklusējuma uzstādījums ir 7 dienas.

2.4.5.4. Servera tabulu relāciju diagramma



3. TESTĒŠANAS DOKUMENTĀCIJA

3.1. Testējamie vienumi

Spēles inicializācijas modulis:

- Rases izvēle
- Kanāla izvēle
- Kanāla parametru norādīšana

Spēles loģikas modulis

- Sasaiste grafisko saskarni un tīkla moduli
- Spēles plūsmas kontrole
- Gājienu validācija

Grafiskās saskarnes modulis

- Elementu kustināšanas funkcionalitāte
- Kāršu izdalīšana
- Kontroles panelis

Tīkla modulis

- Kanāla izveide
- Pieslēgšanās kanālam
- Gājienu pārsūtīšana uz citām spēles instancēm

3.2. Testpiemēru specifikācija

3.2.1. Spēles inicializācijas modulis

Identificējums: INIC.1

Testējamais vienums: Rases izvēle

Apraksts: Jāpārliecinās, ka spēli var palaist ar visām pieļaujamajām vērtībām. Nepieļaujamas vērtības gadījumā lietotājam jālūdz ievads atkārtot.

Ievads: Pieļaujamās vērības: 0, 1 ,2 ,3.

Izvads: Lietotāja izvēle

Atkarības: Nav

Identificējums: INIC.2

Testējamais vienums: Kanāla izvēle

Apraksts: Liek lietotājam izvēlēties starp jauna kanāla izveidi un pievienošanos kanālam. Nepieļaujamas vērtības gadījumā lietotājam jālūdz ievads atkārtot.

Ievads: Pieļaujamās vērības: 0, 1.

Izvads: Lietotāja izvēle

Atkarības: INIC.1

Identificējums: INIC.3

Testējamais vienums: Kanāla parametru norādīšana

Apraksts: Liek lietotājam norādīt kanāla paroli. Ja lietotājs vēlas pievienotie eksistējošam kanālam jānorāda arī Kanāla ID. Gadījumā, ja dati ir nederīgi, jāpaskaidro iemesls.

Ievads: Pieļaujamās vērības:

Kanāla parole: ne null String vērtības; Kanāla ID: Sešpadsmit ciparu integer tipa vērtība.

Izvads: Lietotāja ievadītās vērtības. Ja tika izveidots jauns kanāls, jāizvada kanāla ID numurs.

Atkarības: INIC.2

3.2.2. Spēles loģikas modulis

Identificējums: LOGIC.1

Testējamais vienums: Sasaiste ar grafisko saskarni un tīkla moduli.

Apraksts: Jāpārliecinās, ka loģikas modulis uztver lietotāja veiktās darbības. Jāpārliecinās, ka komandas tiek nosūtītas uz servera norādīto kanālu.

Ievads: Lietotāja darbība.

Izvads: Ziņojums par darbību servera kanālā.

Atkarības: Nav

Identificējums: LOGIC.2

Testējamais vienums: Spēles plūsmas kontrole.

Apraksts: Jāpārliecinās, ka gājienus var izdarīt tikai un vienīgi aktīvā spēlētāja klients. Jāpārliecinās, ka pārējie klienti gājienus saņem. Jāpārliecinās, ka pareizi notiek aktīvā spēlētāja nomaiņa.

Ievads: Komanda aktīva spēlētāja klientā.

Izvads: Komanda izpildās visu spēlētāju klientos.

Atkarības: LOGIC.1, GUI.2, GUI.3, NET.1, NET.2, NET.3

Identificējums: LOGIC.3

Testējamais vienums: Gājienu validācija.

Apraksts: Jāpārliecinās, ka elementus var kustināt tikai aktīvais spēlētājs. Elementus atļauts kustināt, tikai uz spēles laukumā esošiem brīviem lauciņiem.

Ievads: Nekorekts gājiens.

Izvads: Kļūdas paziņojums.

Atkarības: Nav

3.2.3. Grafiskās saskarnes modulis

Identificējums: GUI.1

Testējamais vienums: Kāršu izdalīšana.

Apraksts: Jāpārliecinās, ka kārtis tiek izdalītas nejauši.

Ievads: Nav

Izvads: Kārtis parādās grafiskajā saskarnē.

Atkarības: LOGIC.1

Identificējums: GUI.2

Testējamais vienums: Elementu kustināšanas funkcionalitāte

Apraksts: Jāpārliecinās, ka adekvāti funkcionē elementu pacelšana un nomešana (drag and drop)

Ievads: Elementa kustība

Izvads: Elements pārvietojas uz norādīto lauciņu

Atkarības: LOGIC.1, LOGIC.3

Identificējums: GUI.3

Testējamais vienums: Kontroles panelis

Apraksts: Jāpārliecinās, ka pareizi tiek attēlots aktīvais spēlētājs un lokālā spēlētāja rase. Jāpārliecinās, ka aktīvajam spēlētājam nospiežot pogu "End Action" gājiens pāriet nākamajam spēlētājam.

Ievads: Tiek nospiesta poga "End Action"

Izvads: Informācija par aktīvo spēlētāju un lokālā spēlētāja rasi

Atkarības: LOGIC.1

3.2.4. Tīkla modulis

Identificējums: NET.1

Testējamais vienums: Kanāla izveide

Apraksts: Jāpārliecinās, ka tiek izveidots kanāls ar norādīto paroli

Ievads: Kanāla parole

Izvads: Izveidotā kanāla ID.

Atkarības: INIC.2, INIC.3

Identificējums: NET.2

Testējamais vienums: Pieslēgšanās kanālam

Apraksts: Jāpārliecinās, ka kanālam var pieslēgties

Ievads: Kanāla ID, kanāla parole

Izvads: Apstiprinājums

Atkarības: NET.1, INIC.2, INIC.3

Identificējums: NET.3

Testējamais vienums: Gājienu pārsūtīšana citām spēles instancēm

Apraksts: Jāpārliecinās, ka nosūtītie gājieni nonāk norādītajā servera kanālā. Jāpārliecinās, ka citas spēles instances spēj ziņojumus saņemt

Ievads: Ziņojums

Izvads: Ziņa nonāk norādītajā servera kanālā

Atkarības: NET.1, NET.2

3.3. Testēšanas žurnāls

3.3.1. Testēšanas iterācija nr.1

Testēšanas datums: 29.05.2014

Testpiemēra identifikators	**Rezultāts**
INIC.1	Izpildās
INIC.2	Izpildās
INIC.3	Izpildās
LOGIC.1	Izpildās
LOGIC.2	Izpildās
LOGIC.3	Izpildās
GUI.1	Testpiemērs tika palaists 10 reizes. Iegūtie kāršu komplekti visos gadījumos bija dažādi. Var uzskatīt, ka testpiemērs izpildās
GUI.2	Izpildās
GUI.3	Izpildās
NET.1	Izpildās
NET.2	Izpildās
NET.3	Izpildās

Kopā no 12 testpiemēriem veiksmīgi izpildījās 12.

Programmas izstrādes laikā regulāri veicu izstrādātās funkcionalitātes pārbaudes, jo programmas moduļi ir ļoti atkarīgi viens no otra, tādēļ vienībtestēšanā netika atrasta neviena kļūda.

Lai pilnveidotu testēšanas kvalitāti būtu vērtīgi piesaistīt ar programmas izstrādi nesaistītu personu un veikt melnās kastes testēšanu.

4. PROJEKTA ORGANIZĀCIJA

Projekta izstrāde noritēja pēc tuvināta ūdenskrituma moduļa.

Sākotnēji tika definēti vispārīgi produkta mērķi un vīzija. Vadoties pēc šīs produkta vīzijas tika veikta iespējamo tehnoloģiju un risinājumu izpēte. Tika izvēlēta Java programmēšanas valoda, jo man iepriekš ir bijusi neliela pieredze mazu programmu izstrādē ar java swing un tādēļ, ka Java ir valoda, kurā vēlos specializēties.

Tika veikts aptuvens klašu un funkcionalitātes projektējums. Sākotnējā projekta iecere izrādījās pārāk apjomīga, jo lai pilnvērtīgi realizētu projektu ir nepieciešams izveidot arī serveri, kas nodrošina komunikāciju starp spēles klientiem. Uz doto mirkli man trūka pieredzes, lai spētu skaidri pārredzēt nepieciešamos elementus un funkcionalitāti servera pusē. Pēc konsultācijas ar zinošākiem speciālistiem, izvēlējos izmantot xml-rpc protokolu un Google App Engine servera hostēšanai.

Pēc papildus izpētes, projekta funkcionalitāte tika iedalīta moduļos, moduļi aprakstīti pseidokodā.
Sākotnēji tika izstrādāts spēles loģikas un grafiskās saskarnes modulis, pēc kuru notestēšanas tika pievienots Tīkla modulis.
Izstrādes beigās tika veikta programmas vienībtestēšana un detalizētāka projektējuma izveide.

5. KONFIGURĀCIJAS PĀRVALDĪBA

Programmatūras izstrādes laikā tika izmantota Eclipse izstrādes vide. Koda versiju pārvaldībai tika izmantots Eclipse spraudnis Subclipse, kas izmanto Subversion koda pārvaldības sistēmu, kura saglabā koda izmaiņas un nodrošina iespēju salīdzināt esošo pirmkodu ar tā iepriekšējām versijām.

6. KVALITĀTES NODROŠINĀŠANA

Lai nodrošinātu izstrādātā produkta kvalitāti, tika sastādīta dokumentācija atbilstoši standartiem [1] un [2].

Programmējot tika ievērota Java objekt orientētā pieeja. Spēles loģiskajām operācijām tika definētas jaunas konstantes. Aizsargājamie klases mainīgie tika deklarēti kā privāti, to piekļuve tika nodrošināta caur get un set metodēm. Klašu metodēm tika pievienoti Javadoc formāta komentāri, kas satur metožu to funkcionalitātes un atgriežamo vērtību aprakstu. Tika ievērota labā programmēšanas praksē mainīgo un klašu nosaukumu veidošanā.

Programmas izstrādes laikā, pēc lielākas funkcionalitātes pievienošanas tika veikti regresa testi. Izstrādes beigās tika veikta vienībtestēšana.

7. DARBIETILPĪBAS NOVĒRTĒJUMS

Lai novērtētu darbietilpību izmantoju "Basic COCOMO " [3] darbietilpības novērtēšanas metodi.

Sākotnēji bija grūti prognozēt kopējo koda rindiņu skaitu, jo nekad nebiju veidojis programmu klientus, kas savā starpā komunicē ar datu pārraides protokolu.

Konsultējoties ar nozares profesionāli ar piecu gadu pieredzi un aprakstot vēlamo projektu, ieguvu atbildi, ka aptuvenais Java rindiņu skaits varētu būt no aptuveni virs 1000 atkarībā no realizācijas.

Izmantojot CLOC rindiņu skaitīšanas rīku [5] ieguvu reālo rindiņu skaitu.

Kopējais koda rindiņu skaits sastādīja:

- 1316 rindiņas valodā Java priekš spēles klienta.
- 70 rindiņas priekš servera

Kopā aptuveni 1400 rindiņas.

"Basic COCOMO" metode balstās uz šādu formulu: **PM = A*(LOC/1000)^B** , kur

- A=2,4 un B=1,05 ir konstantes izvēloties projekta tipu "organic". Šāds tips tika izvēlēts, jo ir maza izstrādes komanda: 1 cilvēks.
- LOC - koda rindiņu skaits (lines of code)
- PM - projekta paveikšanai nepieciešamie personmēneši

Rezultāti:

Prognozētais: 2,4*(1100/1000)^1,05=2,65

Reālais: 2,4*(1400/1000)^ 1,05=3,4

Secinājumi:

Iegūtie rezultāti ir samērā precīzi atspoguļo atšķirības starp reālo un prognozēto darbietilpību. Sākotnēji prognozētais rindiņu skaits un darbietilpība tika pārsniegta, jo trūka pieredzes spēlētāju gājienu pārsūtīšanas realizācijā un servera uzstādīšanā. Darbietilpību palīdzēja samazināt faktors, ka programmas pasūtītājs biju es pats, un man bija skaidrs programmas vēlamais rezultāts un mērķi jau no sākuma.

8. IZMANTOTĀ LITERATŪRA UN AVOTI

1. **LVS 68:1996** Programmatūras prasību specifikācijas ceļvedis.

2. **LVS 72:1996** Ieteicamā prakse programmatūras projektējuma aprakstīšanai.

3. "Basic COCOMO": [atsauce 29.05.2014]

http://groups.engin.umd.umich.edu/CIS/course.des/cis525/js/f00/kutcher/kutcher.html

https://files.ifi.uzh.ch/rerg/arvo/courses/seminar_ws02/reports/Seminar_4.pdf

4. Google App engine: [atsauce 29.05.2014]

https://developers.google.com/appengine/?csw=1

5. Cloc - Count Lines Of Code: [atsauce 29.05.2014]

http://cloc.sourceforge.net/

6. Tiešsaistes kauliņu mešana: [atsauce 29.05.2014]

http://rolz.org/

7. "Chaos in the Old World" noteikumi: [atsauce 29.05.2014]

http://www.fantasyflightgames.com/ffg_content/chaos_in_the_old_world/chaos-in-the-world-rule-book/chaos-in-the-old-world-rule-book-web.pdf

8. Oracle Java 1.7 dokumentācija: [atsauce 29.05.2014]

http://docs.oracle.com/javase/specs/jls/se7/html/index.html

9. PROGRAMMATŪRAS KODS

9.1. Klase SwingPlayerHandler

```
1. package warhammer.gui;
2. 
3. /**
4.  * Spēles grafiskā saskarne
5.  * x un y koordinātas norāda uz elementa augšējo kreiso stūri
6.  */
7. public class SwingPlayerHandler extends JPanel implements IPlayerHandler
   {
8. 
9.     private static final long serialVersionUID = -3715283061283593819L;
10.       //Laukuma sākuma atskaites punkts
11.       private static final int BOARD_START_X = 0;
12.       private static final int BOARD_START_Y = 0;
13.       //Laukuma rūtiņu lielums
14.       private static final int SQUARE_WIDTH = 36;
15.       private static final int SQUARE_HEIGHT = 50;
16.       //Elementu lielums
17.       private static final int Element_WIDTH = 36;
18.       private static final int Element_HEIGHT = 48;
19. 
20.       private static final int ElementS_START_X = BOARD_START_X + (int)
   (SQUARE_WIDTH / 2.0 - Element_WIDTH / 2.0);
21.       private static final int ElementS_START_Y = BOARD_START_Y + (int)
   (SQUARE_HEIGHT / 2.0 - Element_HEIGHT / 2.0);
22. 
23.       //Elementu kustināšanas parametri
24.       private static final int DRAG_TARGET_SQUARE_START_X =
   BOARD_START_X - (int) (Element_WIDTH / 2.0);
25.       private static final int DRAG_TARGET_SQUARE_START_Y =
   BOARD_START_Y - (int) (Element_HEIGHT / 2.0);
26. 
27.       private Image imgBackground; // Spēles fons
28.       private JLabel lblGameState; // Norāda aktīvo spēlētāju
29.       private JLabel lblPlayingAs; // Norāda lokālā spēlētāja rasi
30.       private JButton endActionButton;
31.       private JPanel controlPanel;
32.       private JFrame f;
33. 
```

```
34.       private BoardGame boardGame; // Speles loģikas moduļa instance
35.       private List<GuiElement> guiElements = new
   ArrayList<GuiElement>(); //Spēles elementu saraksts
36.
37.       private GuiElement dragElement; // Elements, kurš tiek kustināts
38.       private Move currentMove; //Veiktā kustība
39.
40.       private boolean isMovingEnabled; //Vai atļauts viekt gājienu?
41.
42.       public SwingPlayerHandler(BoardGame boardGame, int localPlayer) {
43.           this.setLayout(null);
44.
45.           // Fona ielāde
46.           URL urlBackgroundImg =
   getClass().getResource("/warhammer/gui/img/bo.png");
47.           this.imgBackground = new
   ImageIcon(urlBackgroundImg).getImage();
48.
49.           // Piešķir loģikas moduļa instanci
50.           this.boardGame = boardGame;
51.
52.           // Spēles loģisko elementu pievienošana
53.           for (Element element : this.boardGame.getElements()) {
54.               createAndAddGuiElement(element);
55.           }
56.
57.           //
58.           ElementsDragAndDropListener listener = new
   ElementsDragAndDropListener(this.guiElements, this);
59.           this.addMouseListener(listener);
60.           this.addMouseMotionListener(listener);
61.
62.           // Kontroles paneļa izveide
63.           controlPanel = new JPanel();
64.           controlPanel.setBackground(Color.GRAY);
65.           controlPanel.setPreferredSize(new Dimension(160, 200));
66.
67.           String labelText;
68.           // Izvēlētā rases attēlošana
69.           labelText = Element.getRaceText(localPlayer);
70.           this.lblPlayingAs = new JLabel("Playing As:    " +
   labelText);
```

```
71.             lblPlayingAs.setForeground(Color.WHITE);
72.             controlPanel.add(lblPlayingAs);
73.
74.             // Aktīvā spēlētāja attēlošana
75.             labelText = this.getGameStateAsText();
76.             this.lblGameState = new JLabel(labelText);
77.             lblGameState.setForeground(Color.WHITE);
78.             controlPanel.add(lblGameState);
79.
80.             // Gājiena beigšanas poga
81.             this.endActionButton = new JButton("End Action");
82.             controlPanel.add(endActionButton);
83.
84.             // Gājiena beigšanas pogas klausītājs
85.             endActionButton.addActionListener(new ActionListener() {
86.                 @Override
87.                 public void actionPerformed(ActionEvent e) {
88.                     currentMove = new Move(true);
89.                 }
90.             });
91.
92.             // Saskarnes loga izveide
93.             f = new JFrame();
94.             f.setSize(1500, 1500);
95.             f.setVisible(true);
96.             f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
97.             f.add(this);
98.             f.add(controlPanel, BorderLayout.EAST);
99.             f.setSize(imgBackground.getWidth(null) + 140,
    imgBackground.getHeight(null) + 20);
100.
101.        }
102.
103.        /**
104.         * @return Atgriež informāciju par aktīvo spēlētāju
105.         */
106.        private String getGameStateAsText() {
107.            String state = "unknown";
108.            switch (this.boardGame.getGameState()) {
109.            case BoardGame.GAME_STATE_KHORNE:
110.                state = "Khorne";
111.                break;
```

```
112.            case BoardGame.GAME_STATE_NURGLE:
113.                state = "Nurgle";
114.                break;
115.            case BoardGame.GAME_STATE_TZEENTCH:
116.                state = "Tzeentch";
117.                break;
118.            case BoardGame.GAME_STATE_SLAANESH:
119.                state = "Slaanesh";
120.                break;
121.            case BoardGame.GAME_STATE_END:
122.                state = "end";
123.                break;
124.            }
125.            return state;
126.        }
127.
128.        /**
129.         * Izveido elementa grafisko instanci
130.         * @param type - elementa tips
131.         * @param x -  augšējā kreisā stūra x koordināta
132.         * @param y - augšējā kreisā stūra y koordināta
133.         */
134.        private void createAndAddGuiElement(Element Element) {
135.            Image img = this.getImageForElement(Element.getRace(),
    Element.getType());
136.            GuiElement guiElement = new GuiElement(img, Element);
137.            this.guiElements.add(guiElement);
138.        }
139.
140.        /**
141.         * Ielādē elementa bildi atkarībā no tā rases un tipa
142.         */
143.        private Image getImageForElement(int race, int type) {
144.
145.            String filename = "";
146.
147.            switch (race) {
148.            case Element.RACE_KHORNE:
149.                filename += "k";
150.                break;
151.            case Element.RACE_NURGLE:
152.                filename += "n";
```

```
153.             break;
154.         case Element.RACE_TZEENTCH:
155.             filename += "t";
156.             break;
157.         case Element.RACE_SLAANESH:
158.             filename += "s";
159.             break;
160.         }


161.         switch (type) {
162.         case Element.TYPE_CARD1:
163.             filename += "1";
164.             break;
165.         case Element.TYPE_CARD2:
166.             filename += "2";
167.             break;
168.         case Element.TYPE_CARD3:
169.             filename += "3";
170.             break;
171.         case Element.TYPE_CARD4:
172.             filename += "4";
173.             break;
174.         case Element.TYPE_CARD5:
175.             filename += "5";
176.             break;
177.         case Element.TYPE_CARD6:
178.             filename += "6";
179.             break;
180.         case Element.TYPE_CARD7:
181.             filename += "7";
182.             break;
183.         case Element.TYPE_CARD8:
184.             filename += "8";
185.             break;
186.         case Element.TYPE_CULTIST:
187.             filename += "c";
188.             break;
189.         case Element.TYPE_WARRIOR:
190.             filename += "w";
191.             break;
192.         case Element.TYPE_BOSS:
```

```
193.                 filename += "b";
194.                 break;
195.             }
196.             filename += ".png";
197.
198.             URL urlElementImg =
    getClass().getResource("/warhammer/gui/img/" + filename);
199.             return new ImageIcon(urlElementImg).getImage();
200.         }
201.
202.         @Override
203.         protected void paintComponent(Graphics g) {
204.
205.             // Uzzīmē fonu
206.             g.drawImage(this.imgBackground, 0, 0, null);
207.
208.             // Uzzīmē elementus
209.             for (GuiElement guiElement : this.guiElements) {
210.                     g.drawImage(guiElement.getImage(), guiElement.getX(),
    guiElement.getY(), null);
211.             }
212.
213.             // Attēlo informāciju par aktīvo spēlētāju
214.             this.lblGameState.setText(this.getGameStateAsText());
215.         }
216.
217.         /**
218.          * @return Atgriež informāciju ar aktīvo spēlētāju
219.          */
220.         public int getGameState() {
221.             return this.boardGame.getGameState();
222.         }
223.
224.         /**
225.          * convert pārvērs kolonnu x koordinātā
226.          * @param column
227.          * @return x
228.          */
229.         public static int convertColumnToX(int column) {
230.             return ElementS_START_X + SQUARE_WIDTH * column;
231.         }
232.
```

```
233.      /**
234.       * pārvērš rindu y koordinātā
235.       * @param row
236.       * @return y
237.       */
238.      public static int convertRowToY(int row) {
239.          return ElementS_START_Y + SQUARE_HEIGHT * (Element.ROW_16 -
   row);
240.      }
241.
242.      /**
243.       * pārvērš x koordinātu kolonnā
244.       * @param x
245.       * @return kolonna
246.       */
247.      public static int convertXToColumn(int x) {
248.          return (x - DRAG_TARGET_SQUARE_START_X) / SQUARE_WIDTH;
249.      }
250.
251.      /**
252.       * pārvērš y koordinātu rindā
253.       * @param y
254.       * @return logical row for y coordinate
255.       */
256.      public static int convertYToRow(int y) {
257.          return Element.ROW_16 - (y - DRAG_TARGET_SQUARE_START_Y) /
   SQUARE_HEIGHT;
258.      }
259.
260.      /**
261.       * Pārvieto elementu uz norādīto pozīciju. Ja pozīcija nav derīga
   atgriež elementu uz sākumpozīciju
262.       * @param dragElement
263.       * @param x
264.       * @param y
265.       */
266.      public void setNewElementLocation(GuiElement dragElement, int x,
   int y) {
267.          int targetRow = SwingPlayerHandler.convertYToRow(y);
268.          int targetColumn = SwingPlayerHandler.convertXToColumn(x);
269.
```

```
270.          Move move = new Move(dragElement.getElement().getRow(),
   dragElement.getElement().getColumn(), targetRow, targetColumn);
271.
272.          if (this.boardGame.getMoveValidator().isMoveValid(move,
   true)) {
273.              this.currentMove = move;
274.
275.          } else {
276.              dragElement.resetToUnderlyingElementPosition();
277.          }
278.      }
279.
280.      /**
281.       * @param guiElement - saglabā datus par elementu, kuru kustina
   lietotājs
282.       */
283.      public void setDragElement(GuiElement guiElement) {
284.          this.dragElement = guiElement;
285.      }
286.
287.      /**
288.       * @return atgriež elementu, kuru lietotājs kustina
289.       */
290.      public GuiElement getDragElement() {
291.          return this.dragElement;
292.      }
293.
294.      @Override
295.      public Move getMove() {
296.          //Kustināšanas iespējošana
297.          this.isMovingEnabled = true;
298.
299.          //Atgriež veikto kustību
300.          Move moveForExecution = this.currentMove;
301.          this.currentMove = null;
302.          return moveForExecution;
303.      }
304.
305.      @Override
306.      public void moveSuccessfullyExecuted(Move move) {
307.
308.          // Pārbaude, vai elements veiksmīgi pakustināts
```

```
309.            GuiElement guiElement = this.getGuiElementAt(move.targetRow,
    move.targetColumn);
310.            if (guiElement == null) {
311.                throw new IllegalStateException("no guiElement at " +
    move.targetRow + "/" + move.targetColumn);
312.            }
313.            guiElement.resetToUnderlyingElementPosition();
314.
315.            // Atspējo elementu kustināšanu
316.            this.isMovingEnabled = false;
317.
318.            // Uzzīmē elementa jauno atrašanās vietu
319.            this.repaint();
320.
321.        }
322.
323.        @Override
324.        public void endTurn() {
325.            this.boardGame.swapActivePlayer();
326.            this.repaint();
327.        }
328.
329.        public boolean isMovingEnabled() {
330.            return isMovingEnabled;
331.        }
332.
333.
334.        /**
335.         * @param row
336.         * @param column
337.         * @return Atgriež elementu, kas atrodas norādītajā rindā un
    kolonnā
338.         */
339.        private GuiElement getGuiElementAt(int row, int column) {
340.            for (GuiElement guiElement : this.guiElements) {
341.                if (guiElement.getElement().getRow() == row &&
    guiElement.getElement().getColumn() == column) {
342.                    return guiElement;
343.                }
344.            }
345.            return null;
346.        }
```

```
347.
348.      public void clearLastMove() {
349.          this.currentMove = null;
350.      }
351.  }
```

9.2 Klase Move

```
package warhammer.logic;

/**
 * Satur elementu kustības informāciju
 */
public class Move {
    public int sourceRow;
    public int sourceColumn;
    public int targetRow;
    public int targetColumn;
    /**
     * Mainīgais norāda, vai gājiens jānodod nākamajam spēlētājam
     */
    public boolean endTurn = false;

    /**
     * @param sourceRow
     * @param sourceColumn
     * @param targetRow
     * @param targetColumn
     */
    public Move(int sourceRow, int sourceColumn, int targetRow, int
targetColumn) {
        this.sourceRow = sourceRow;
        this.sourceColumn = sourceColumn;
        this.targetRow = targetRow;
        this.targetColumn = targetColumn;
    }


    /**
     * @param Izveido klases instanci, kas informē par gājiena beigām.
     */
    public Move(boolean endTurn) {
        this.endTurn = endTurn;
    }

    /**
     * Pārvērš klases Move instanci par String komandu
     * @param klases Move elements
     * @return kustības String vērtība
```

```
     */
    public static String convertMoveToString(Move move) {
        String result = "";
        // Formāts: SRSC-TRTC
        // Sākum lauciņš
        if (move.sourceRow < 10) {
            result += "0" + Integer.toString(move.sourceRow);
        } else {
            result += Integer.toString(move.sourceRow);
        }

        if (move.sourceColumn < 10) {
            result += "0" + Integer.toString(move.sourceColumn);
        } else {
            result += Integer.toString(move.sourceColumn);
        }

        result += "-";

        // Mērķa lauciņš
        if (move.targetRow < 10) {
            result += "0" + Integer.toString(move.targetRow);
        } else {
            result += Integer.toString(move.targetRow);
        }

        if (move.targetColumn < 10) {
            result += "0" + Integer.toString(move.targetColumn);
        } else {
            result += Integer.toString(move.targetColumn);
        }

        return result;
    }

    /**
     * @param Gājiena komanda String veidā
     * @return Move klases instance
     */
    public static Move convertStringToMove(String input) {
        if (input == null) {
            return null;
```

```
        }

        if (input.equals("end_turn")) {
            return new Move(true);
        }

        String stringSR = input.substring(0, 2);
        String stringSC = input.substring(2, 4);
        String stringTR = input.substring(5, 7);
        String stringTC = input.substring(7, 9);

        int sourceRow = Integer.parseInt(stringSR);
        int sourceColumn = Integer.parseInt(stringSC);
        int targetRow = Integer.parseInt(stringTR);
        int targetColumn = Integer.parseInt(stringTC);

        return new Move(sourceRow, sourceColumn, targetRow, targetColumn);
    }
}
```

Kvalifikācijas darbs „***Galda spēles realizācija tiešsaiste***” 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: ***Reinis Elksnis*** _____ **30.05.2014**.

Rekomendēju darbu aizstāvēšanai
Darba vadītājs: **Mg. Dat.** ***Jānis Iļjins*** _____ **30.05.2014.**

Recenzents: **M. Dat. Agnis Škuškovniks**

Darbs iesniegts 02.06.2014.
Kvalifikācijas darbu pārbaudījumu komisijas sekretārs: Imants Gorbāns _____

Darbs aizstāvēts kvalifikācijas darbu pārbaudījuma komisijas sēdē
___.06.2014. prot. Nr. _____
Komisijas sekretārs(-e): _____