LATVIJAS UNIVERSITĀTE
DATORIKAS FALUKTĀTE

VAIRĀKPLATFORMU SPĒLES IZSTRĀDES PROCESS, IZMANTOJOT LIBGDX IETVARU, UN TĀ SALĪDZINĀJUMS AR LĪDZĪGIEM RĪKIEM

MAĢISTRA DARBS

Autors: **Oskars Zandersons**

Studenta apliecības nr.: oz11012

Darba vadītājs: asociētais profesors Dr. sc. comp. Edgars Celms

Rīga 2017

ANOTĀCIJA

Maģistra darbā tiek detalizēti un pilnīgi apskatīts un aprakstīts spēles izstrādes process. Procesa apskats notiek strukturēti, to iedalot trīs etapos – plānošanas, ražošanas un uzturēšanas, katru no šiem etapiem iedalot smalkākos, atbilstoši izstrādes procesa specifikai.

Darbā tiek sniegts izstrādes procesa plānošanas etapa rezultātu piemērs uz izstrādājamās spēles pamata. Kā arī spēles konceptuāls un detalizēts projektējums un sākotnējās prototipēšanas etapa rezultātu apkopojums. Projektēšana notiek, ņemot vērā izpētītā libGDX ietvara sniegtās iespējas, ar mērķi demonstrēt ietvara funkcionalitātes pielietošanas iespējas praksē.

Maģistra darbā tiek piedāvāts arī apskatīt vairākus iespējamos rīkus aprakstītās spēles izstrādei, kuri atbilst izvirzītajām prasībām. Rīki tiek apskatīti, apskates rezultāti tiek analizēti un uz tā pamata tiek argumentēta autora izvēle par labu libGDX rīkam.

Darbā tiek detalizēti un dziļi pētītas libGDX ietvara piedāvātās iespējas spēles izstrādes procesa realizācijai, kā arī iespējas, ko apskatāmais ietvars sniedz vairākplatoformu spēles izstrādes problemātikas risināšanai.

Atslēgvārdi:

Spēles izstrādes process, libGDX, vairākplatformu spēles izstrāde, spēļu izstrādes rīki

ANNOTATION

CROSS-PLATFORM GAME CREATION PROCESS USING LIBGDX FRAMEWORK AND ITS COMPARISON WITH SIMILAR TOOLS

At this article, reader can get information about in all aspects described game development process. It is described in a structured way. First comes planning, then development and after all maintenance of developed game. This is how it is described in this article.

Not just in theory game development is described, but also reader can see example of real game development process, which is going to be developed in terms of this article. It is supposed to create game high level concept, to process game modeling at conceptual and more detailed level and create prototypes.

Developing game is not possible without tools, so one of goals of this article is to see on what tools are available for defined target and choose the best option for us. There is also a good argumentation given by author why he chooses libGDX tool for his game development.

As author choose libGDX framework to be used as a base for development process, there is also a detailed research and analysis of libGDX features and how libGDX solves cross-platform development problems.

Key words:

Game development process, libGDX, cross-platform game development, game development frameworks

AUTOREFERĀTS

Šī maģistra darba ietvaros autors izpētīja spēles izstrādes procesu, veica literatūras analīzi, lai noteiktu, no kādiem etapiem tas sastāv, un kas veicams katrā no tiem. Autors pielietoja iegūtās zināšanas, lai veiktu atsevišķu izstrādes etapu realizēšanu, piemēram, veica plānošanas un projektēšanas pasākumus.

Autors veica izstrādes ietvaru, kurus paredzēts izmantot vairākplatformu spēles un lietotņu izstrādei, salīdzinājumu, kā arī veica detalizētu libGDX ietvara izpēti kā arī apskatīja, kā tiek risinātas vairākplatformu izstrādes problēmas šī ietvara robežās.

Autors nonāca pie vērtīgiem secinājumiem, ka libGDX ietvars pilnībā nodrošina vairākplatformu atbalstu programmatūras koda izstrādes līmenī, taču, lai nodrošināt veiktspējas un lietotāja pieredzes atbilstību dažādu platformu ierobežojumiem, nepieciešams veikt atbilstošu projektēšanu un saskarnes veidošanu.

Lai izvēlēties rīkus, kuri tiek apskatīti maģistra darba ietvaros, autors definēja prasības attiecībā pret tiem. Rīku salīdzināšanas rezultātā autors nonāca pie secinājuma, ka mūsdienās ir virkne vienlīdzīgu pēc iespējām un funkcionalitātes rīku, kurus var veiksmīgi izmantot efektīvai vairākplatformu spēļu un lietotņu izstrādei.

Darba ietvaros tika veikta plaša literatūras analīze, tika pielietotas praktiskās zināšanas, kas tika iegūtas gan veicot literatūras analīzi un pētījumu, gan iegūtas apmācības laikā.

Darba rezultātā ir iegūti secinājumi par spēles izstrādes procesu, par vairākplatformu izstrādei pieejamajiem rīkiem, kuri padara vairākplatformu izstrādi efektīvu, kā arī tika veikti praktiski projektēšanas pasākumi spēles izstrādes procesa ietvaros.

Par īpaši interesantu autors uzskata ceturtajā nodaļā veikto libGDX ietvara funkcionalitātes analīzi, kuras ietvaros apskatīti arī piedāvātie abstrakcijas līmeņi vienlaicīgam darbam ar vairākām platformām.

SATURS

APZĪMĒJUMU SARAKSTS

Apzīmējums	**Paskaidrojums**
OpenGL ES 2.0	OpenGL for Embedded Systems– atvērtā grafiskā bibliotēka iebūvētajām sistēmām.
FBX	Filmbox – datņu formāts 3D modeļu glabāšanai
HUD	Head-up display – lietotāja saskarne, kas paredzēta attēlošanai virs virtuālas vides, piemēram, spēlē, saskarnes elementi, kas redzami spēles laikā.
UI	User Interface – lietotāja saskarne.
UML	Unified modeling languege – vienotā modelēšanas valoda – grafiskās reprezentācijas valoda, kuru izmanto objektorientētajā modelēšanā organizacionālo struktūru vizualizācijai.
API	Lietotņu programmēšanas saskarne – gatavu klašu, metožu, procedūru, konstanšu kopa, kuras var izmantot izstrādēs procesā sadarbībai ar apskatāmo moduli.
JSON	JavaScript Object Notation – datu formāts.
XML	e**X**tensible **M**arkup **L**anguage – paplašināmā marķēšanas valoda. XML formātā iespējams glabāt datus analoģiski kā JSON formātā.

IEVADS

Spēles izstrādes process ir ļoti atšķirīgs no vairuma cita veida programmatūras izstrādes procesiem. Spēles izstrāde ietver sevī ļoti plašu un dažādu nozaru speciālistu sadarbību, kas padara spēles izstrādes procesu par pavisam neparastu un netriviālu.

Viens no darba mērķiem ir izpētīt no teorētiskā viedokļa, kāds ir un no kā sastāv spēles izstrādes process. Šīs zināšanas būs noderīgas autoram reālas savas spēles izstrādes projektēšanas etapu veikšanā maģistra darba ietvaros ar mērķi padziļināti no praktiskās puses iepazīt spēļu izstrādes procesu, kā arī izpētīt mūsdienīga spēļu izstrādes rīka sniegtās iespējas un priekšrocības vairākplatformu spēles izstrādē.

Spēles izstrādes procesa praktiskās jeb realizācijas daļas pētīšanu paredzēts veikt bāzējoties uz libGDX rīka sniegtajām iespējām. Šī nolūka realizēšanai, nepieciešams izpētīt detalizētā un dziļā līmenī libGDX rīku un tajā esošo funkcionalitāti kā arī tā piedāvātos risinājumus vairākplatformu izstrādes problemātikas risināšanā.

Maģistra darba ietvaros paredzēts to skaitā apskatīt arī alternatīvus risinājumus vairākplatformu spēles izstrādē un sniegt ieskatu to klāstā un iespējās, to skaitā salīdzinot tos ar libGDX rīku.

Atbilstoši apskatītajam spēles izstrādes procesam un izpētītajām libGDX sniegtajām iespējām, paredzēts veikt spēles projektēšanu konceptuālā un detalizētā līmenī, kā arī veikt sākotnējo prototipēšanu.

1. SPĒLES IZSTRĀDES PROCESS

Spēles izstrādes process ir programmatūras izstrādes process, jo datorspēle tā ir programmatūra, kuras darbībā īpašu lomu ieņem mākslinieciskā grafiskā sastāvdaļa, skaņas pavadījums jeb audio sastāvdaļa un spēles process no lietotāja jeb spēlētāja viedokļa [1, 2].

Spēles izstrādes procesā ir piemērojamas programmatūras izstrādes metodoloģijas un paņēmieni [1], taču spēles izstrādes process ir specifisks un parastās metodoloģijas tiek pārskatītas un pielāgotas [1].

Spēles ar vāji pārdomātu izstrādes procesu un nepietiekami vai nekorekti pielietotu izstrādes metodoloģiju bieži neiekļaujas izstrādes budžetā vai termiņā [1]. Izstrādes plānošana ir vienlīdz nozīmīga gan nelieliem, to skaitā, individuāliem projektiem, gan lieliem spēļu izstrādes projektiem [1].

Līdzīgi kā citu veidu programmatūras produktu izstrādes procesi, arī spēļu izstrādes procesā ietilpst plānošanas un projektēšanas etaps, izstrādes un testēšanas etaps, kā arī pēcizstrādes uzturēšanas etaps [2]. Taču spēles izstrādes procesā katram no šiem etapiem piemīt tikai spēles izstrādei specifiski būtiski elementi un atšķirības.

Šajā nodaļā plānots pakāpeniski un secīgi apskatīt visus spēles izstrādes procesa etapus un to elementus.

1.1. Plānošanas un projektēšanas etaps

Praktiski jebkura programmatūras un ne tikai programmatūras projekta izstrādes pirmais posms ir plānošanas un projektēšanas pasākumi. To nepieciešamais pilnības līmenis, lai varētu sākt izstrādi, ir atkarīgs no izvēlētās izstrādes metodoloģijas un projekta specifikas, taču tā vai citādi viss sākas ar projektēšanu un plānošanu.

Plānošanas un projektēšanas pasākumus, kas ietilpst pirmsražošanas etapā, var iedalīt trijās daļās – koncepcijas radīšana, spēles sākotnējā dizaindokumenta izveide, prototipu veidošana [1]. Koncepcijas radīšanas process, savukārt, arī iedalāms trīs apakšposmos – augsta līmeņa koncepcijas definēšana, spēles priekšlikuma definēšana un spēles kopējās koncepcijas izveide [1].

1.1.1. Augsta līmeņa koncepcija

Augsta līmeņa koncepcijas definēšana ir pirmais no trim spēles koncepcijas izveides posmiem. Pēc būtības augsta līmeņa koncepcija ir īss vispārējs spēles apraksts. Pēc formas tas atbilst apmēram diviem līdz trim teikumiem, kas atbild uz jautājumu: "Par ko ir šī spēle?" [1].

1.1.2. Spēles priekšlikums

Spēles priekšlikums ir īss dokuments, kurā tiek pamatota spēles izstrādes lietderība no ekonomiskā viedokļa. Projektos, kuros izstrādātājs un izdevējs ir dažādas kompānijas, izstrādātāji vispirms sagatavo spēles priekšlikumu un vēlā to prezentē izdevējam. Iespējamas vairākas tikšanās ar labojumu veikšanu līdz projekts tiek apstiprināts. Spēles priekšlikumu iespējams papildināt ar prototipiem vai paraugversijām, taču tas var arī nebūt nepieciešams.

Gadījumā kad izstrādātājs un izdevējs ir viena kompānija, vai vienas kompānijas saistīti apakšuzņēmumu, tad prezentēšana notiek augstākajai pārvaldībai, kas pieņem lēmumu attiecībā par projekta apstiprināšanu.

Individuālu izstrādātāju gadījumā spēles priekšlikums nav tik būtisks no projekta apstiprināšanas viedokļa, taču var būt lietderīgs, lai novērtēt spēles iespējas no ekonomiskā viedokļa.

1.1.3. Spēles kopējā koncepcija

Kopējā spēles koncepcija ir dokuments, kurā detalizētāk nekā iepriekš minētajos tiek aprakstīta spēle un spēles projekts. Šajā dokumentā ietilpst augsta līmeņa koncepcija, spēles žanrs, spēles procesa no lietotāja jeb spēlētāja viedokļa apraksts, spēles īpatnības un īpašās iespējas un funkcionalitāte, konfigurācija, spēles stāsts, definēta mērķauditorija, atbalstāmās platformas, paredzētais realizācijas laiks, tirgus analīze, prasības pret izstrādes komandu, risku analīze.

Pirms spēles projektējums ir pabeigts un apstiprināts, izstrādes komanda jau var ķerties klāt darbam, veicot prototipu izstrādi, kuros tiek demonstrēta spēles funkcionalitāte, kuru ieinteresētās puses gribētu redzēt gala produktā. Mākslinieki var sākt darbu ar grafiskajām komponentēm, veidojot to pirmās versijas, uz kuru pamata vēlāk varēs veidot galējās versijas. Šajā izstrādes etapā projekta vadītāji saskaņo un plāno kopā ar komandu izpildāmo darbu pakotnes un to realizācijai paredzēto laiku un izmaksas.

1.1.4. Spēles dizaindokuments

Spēles dizaindokuments ir detalizēts izstrādājamās spēles apraksts. To veido un maina izstrādes komanda un tas tiek pielietots, lai organizēt izstrādes komandas darbu. Dokuments rodas dizaineru, mākslinieku un programmētāju savstarpējas kooperācijas rezultātā, kā instrukcija, kas tiek pielietota izstrādes procesa laikā. Izstrādātājiem nepieciešams pieturēties pie dizaindokumenta izstrādes laikā, taču notiek arī tā attīstība atbilstoši projekta attīstībai.

Dizaindokumenta mērķis ir detalizēti un viennozīmīgi aprakstīt spēles procesu no spēlētāja viedokļa, grafiku, spēles līmeņus, spēles stāstu, t.i. sižetu, spēlē esošos personāžus, spēlētāja saskarni, kā arī spēles komerciālos aspektus, tās mērķauditoriju [3].

Lai sasniegt noteiktu mērķi, katrai prasībai pret noteiktu spēles daļu ir jābūt detalizēti aprakstītai un saprotamai atbilstošajiem izstrādes komandas locekļiem – māksliniekiem, programmētājiem, u.t.t. Dokumentam ir jābūt apzināti sadalītam tādās daļās, lai spēles izstrādātāji varētu uzturēt atsevišķas tā daļas tālākās izstrādes gaitā [3].

Dizaindokuments pēc būtības ir darba plāns no projekta sākuma līdz galam. Tas nozīmē, ka tajā jāietver visi pamatuzdevumi un to aptuvenas risinājuma metodes. Dizaindokuments var saturēt sekojošas daļas [3]:

- spēles saturisko shēmu, kura atbild uz jautājumu, kas jādara spēlētājam, kāds ir tā galamērķis spēles ietvaros un kas traucēs to sasniegt;
- saskarnes aprakstu, kurā izklāstīta saskarnes funkcionālā daļa – ko un kādā veidā spēlētājs var darīt, izmantojot kādas ievadierīces (pele, klaviatūra) un kādas taustiņu kombinācijas;
- spēles mehānikas aprakstu, kurā izstāstīts, kā iekārtota spēles pasaule, kādi parametri piemīt tās iemītniekiem, kustību formulas, spēles fizika, cīņas norises mehānika, lomu sistēma, u.t.t.;
- programmatūras mehānismu un algoritmu aprakstu, kas ietvers informāciju par to, kādi parametri būs spēles grafiskajai realizācijai, mākslīgajam intelektam, saskarnei, līmeņu un karšu rediģēšanas rīkam, skaņai, u.t.t.;
- grafikas aprakstu, kurā tiks noteikts, cik un kādi nepieciešami modeļi, animācijas, grafikas elementi, video, arī tādi elementi kā ekrānsaudzētāji spēles faniem, kurus arī vēlams ieplānot savlaicīgi, kā arī ir ļoti vēlamas skices un koncepti, kas ļaus noprast spēles vizuālo stilu;

- skaņas un mūzikas daļu, kurā tiek noteikts muzikālais pavadījums, skaņu veidi un to attēlošanas veidi, skaņu efektu kopnes;
- sižeta sadaļu, kurā aprakstīts spēles sižets kopskatā, sižeta kompāniju plāns, pamatuzdevumi u.t.t., atkarībā no spēles žanra, katrai no paredzētajām kartēm spēlē ir jābūt plānotai un aprakstītai šajā sadaļā;
- spēles virtuālās pasaules aprakstu, kurā ietilpst pamata personāžu, monstru, karaspēka veidu apraksts ar parametriem un aptuvenu atrašanās vietu vai ieguves un ražošanas veidiem;
- darbinieku, algu, darba termiņu un plāna sadaļu.

Šis saraksts nav izsmeļošs vai stingri noteikts, tas var mainīties atkarībā no izstrādājamās spēles īpatnībām [3]. Var netikt izmantotas visas aprakstītās sadaļas vai nākt klāt jaunas, kā arī iespējama sadaļu satura pielāgošana projekta vajadzībām. Dažas no sadaļām, kā piemēram darbinieku, viņu algu un darba termiņa un plāna sadaļa var atrasties atsevišķos dokumentos un nebūt iekš spēles dizaindokumenta. Tas attiecināms arī uz citām sadaļām, ja ir tāda nepieciešamība.

1.1.5. Prototipēšana

Spēles procesa un funkcionalitātes prototipu veidošana ir nozīmīga un palīdz programmētājiem un spēles projektētājiem eksperimentēt ar dažādiem algoritmiem un lietošanas scenārijiem. Prototipēšana var izrādīties ļoti noderīga projektēšanas un plānošanas etapa laikā dizaindokumenta veidošanā [1] un var palīdzēt definēt kādu funkcionalitāti vai spēles dizaina, projektējuma elementu.

Prototipi bieži tiek veidoti fiziski, piemēram, uz papīra [1]. Tas var palīdzēt ekonomēt laiku, netērējot to uz kādas idejas programmatūrisku implementāciju, gadījumā ja tā tik un tā tiks atcelta. Nereti to iespējams noteikt pateicoties konceptuālam fiziskam prototipam. Protams, ka prototipēšana notiek arī izstrādes etapa laikā, palīdzot testēt atsevišķas jaunas idejas pirms apstiprināt tās un realizēt spēlē.

Nereti prototipi tiek veidoti un kalpo kā kādas koncepcijas realizācija un ir domāti lai testēt jaunas idejas, pievienojot, modificējot vai izslēdzot noteiktu funkcionalitāti [1]. Lielākā daļa no prototipos esošajiem algoritmiem tiek pārcelti uz izstrādājamo spēli un izmantoti kā reālā koda sastāvdaļa, protams, kad tie ir pabeigti un apstiprināti.

Daudzos gadījumos, kad nepieciešams izstrādāt prototipu, uz tā izstrādi ir maz laika, jo tas nepieciešams burtiski pēc divdesmit minūtēm, lai veikt kādu testēšanu. Šādam uzdevumam parasti tiek izvirzīts kāds ļoti produktīvs programmētājs, kurš ar to ātri tiek galā [1]. Gadījumos, kad nepieciešams prototips fiziskā formā, programmētāji un projektētāji kopā ar dizaineriem to konstruēs izmantojot viegli pieejamus materiālus, piemēram, papīru vai metamos kauliņus, ja nepieciešams simulēt to darbību, piemēram veicot gadījumskaitļu ģenerēšanu vai arī ja spēlē vienkārši tie ir paredzēti.

Veiksmīgs izstrādes modelis ir iteratīva prototipēšana [1], kad dizains un projektējums balstās uz jaunākajiem sasniegumiem projekta izstrādē un pēdējiem izstrādes progresa soļiem.

1.2. Ražošanas etaps

Ražošanas etaps ir izstrādes pamatfāze. Šī etapa laikā tiek paveikt lielākais pēc apjoma darbs un darbā tiek iesaistītas vienlaicīgi visas projektā iesaistītās darbinieku grupas [1, 2]. Programmētāji raksta spēles kodu, mākslinieki nodrošina grafiskos risinājumus, to skaitā 3D modeļus spēles vajadzībām, skaņas inženieri izstrādā skaņas efektus, bet mūzikas kompozitori rada muzikālo pavadījumu. Rakstnieki un scenāristi veido dialogus starp personāžiem un tekstus spēles vajadzībām. Līmeņu izstrādātāji strādā pie spēles līmeņu un karšu radīšanas, bet spēles dizaineri un projektētāji turpina projektēt spēli līdzi izstrādes norisei un nodrošina, ka projekts atspoguļo aktuālo spēles redzējumu.

1.2.1. Spēles projektēšana, dizaindokumenta attīstība

Spēles projektēšana un tās kopējā dizaina veidošana ir ļoti nozīmīga visas komandas kopēja darbība. Spēles satura un noteikumu veidošana prasa gan mākslinieciskās prasmes, gan tehnisko kompetenci. Nepieciešams radošums un atvērta idejām uztvere.

Izstrādes procesa laikā, spēles projektētāji un dizaineri uzlabo un modificē esošo spēles projektējumu atbilstoši aktuālajam spēles redzējumam [1, 3]. Šis redzējums var mainīties un evolucionēt izstrādes laikā. Var izmainīties kā jau esošais spēles stāsts, tā arī iespējams, ka tiek nolemts atbalstīt vēl kādu tehnoloģisko platformu, kas ievieš savas korektīvas spēles dizainā un projektējumā.

Lielākā daļa no tāda veida izmaiņām tiek dokumentētas spēles dizaindokumentā, kas regulāri mainās izstrādes gaitā.

1.2.2. Spēles programmēšana

Spēles programmēšanu veic programmētāju grupa vai viens programmētājs, atkarībā no projekta lieluma un izstrādes grupas sastāva [1, 4]. Programmētāji ne tikai implementē paredzēto funkcionalitāti, bet arī nodarbojas ar prototipu izstrādi ne tikai plānošanas etapā, bet arī izstrādes posmā [1, 4].

Lielā daļā spēļu ražošanas projektu tiek izmantoti jau gatavi spēļu dziņi, taču arī šādos projektos ir nepieciešami programmētāji, lai šos dziņus iestatīt un konfigurēt visus neskaitāmos parametrus un pielāgot dzini spēles prasībām [4].

1.2.3. Spēles līmeņu veidošana

Līmeņu veidošana paredz to, kas noslēpts šīs disciplīnas nosaukumā – līmeņu veidošanu spēlei. Tas ietver gan karšu, gan uzdevumu, lokāciju, misiju veidošanu, kā arī pārējās apkārtējās virtuālās vides veidošanu. [5] Parasti līmeņi spēlēm tiek veidoti izmantojot speciāli radītu līmeņu redaktoru – tā ir programmatūra, kas ļauj ērti izmantojot ērtu lietotāja saskarni konstruēt lokācijas spēlei ievietojot tajās visu iepriekš minēto, kā arī veidot scenārijus personāžiem un definēt uzdevumus spēlētājiem.

Ir gadījumi, kad līmeņu redaktors kopā ar spēli ir pieejams arī spēlētājiem. Piemēram, spēles "Warcraft III" līmeņu redaktorā jebkurš var izveidot savu karti [6], kurā var realizēt patvaļīgu reljefu, pilnībā jebkādu pašizdomātu lokācijas shēmu, ievietot personāžus, pat pievienot tiem pašskriptētus darbības scenārijus [6]. Protams, tiek izmantoti jau esošie spēlē grafiskie resursi un elementi, taču rīcības brīvība ir maksimāli plaša. Tas ir viens no piemēriem, kas ļauj aplūkot, kā izskatās un darbojās līmeņu redaktors.

Līmeņu veidošana sākas no koncepcijas radīšanas [5]. Šīm vajadzībām pieaicina mākslinieku, kurš ir spējīgs radīt līmeņa skici, bet kurš nepārvalda tehniskās prasmes atbilstošā līmenī, lai radīt līmeni patstāvīgi.

Atkarībā no spēles žanra un spēles īpatnībās līmeņu veidošana var sastāvēt no atšķirīgiem apakšprocesiem, taču tās pamata posmi parasti ir sekojoši [5]:

- kartes teritorijas sadalīšana sektoros – kalni, pilsētas, tuneļi plači un ceļi, pa kuriem iespējams pārvietoties spēlētājam un pretiniekam;
- atsevišķu reģionu, kuros jānotiek kādai darbībai, piemēram resursu vākšanai, definēšana;

- dinamisku objektu atrašanās vietas noteikšana, piemēram durvis, atslēgas, pogas, dažādi mehānismi, ar kuriem var mijiedarboties, u.t.t.;
- organizācijas notikumu punktu noteikšana, piemēram, ienaidnieka atdzīvošanās vieta, spēlētāja atrašanās vieta, resursu uzkrāšanās teritorija, saglabāšanās punkti, u.t.t.;
- starta un beigu punktu noteikšana vienam vai vairākiem spēlētājiem;
- dažādu detaļu pievienošana, piemēram, grafiskās tekstūras, skaņas, animācijas, apgaismojums, muzikālais pavadījums, u.t.t.;
- skriptu un notikumu trigeru pievienošana līmenim;
- ceļa atrašanas skriptu pievienošana monstriem un spēles personāžiem, kurus nekontrolē spēlētājs, vietu noteikšana, kuros tie var atrasties, notikumu, kuri notiek pēc to nonākšanas noteiktās vietās, piesaistīšana, kā arī dialogu starp tiem un ar spēlētāju pievienošana.

Līmeņa veidošana ir iteratīva un var sastāvēt no vairākām iterācijām līdz tiek sasniegts apmierinošs rezultāts [5].

1.2.4. Spēles mākslinieciskās daļas veidošana

Spēles mākslinieciskās daļas veidošana sākas vēl plānošanas un projektēšanas etapā. Ar šī spēles izstrādes procesa aspekta veidošanu nodarbojas vizuālās mākslas speciālisti, kas vispirms rada aptuvenas personāžu skices, to konfigurāciju, objektu skices [1]. Protams šos sākotnējos aspektus varētu veidot arī spēles projektētāji, taču vēlāk ar konkrētu risinājumu izstrādi vienalga nodarbojas vizuālās mākslas speciālisti, kas pārveido skices par konkrētiem datormākslas darbiem un objektiem.

Līdzīgi kā ar programmētājiem, arī mākslinieku komandas skaits var variēties atkarībā no izstrādes komandas komplektācijas un projekta lieluma [1, 2]. Projektā var būt arī viens mākslinieks. Lielos projektos, kuros iespējams piesaistīt vairākus mākslinieku vai veselu mākslinieku komandu, tos var sadalīt pēc to prasmju novirzieniem un veikt darba dalīšanu starp tiem.

Spēles mākslinieciskās daļas veidošana ietver saistītās mēdijas izveidi, paraugskatus, mākslinieciskus ekrānšāviņus, kā arī vienkārši sagatavotus parauga ekrānšāviņus, kas ievērojami ietekmē potenciālos spēles lietotājus, jo šādi mākslinieku darba rezultāti var tikt novērtēti vēl līdz spēle tiek izlaista, kamēr pats spēles process novērtēts vēl nevar būt.

Mākslinieki strādā ciešā sadarbībā ar spēles dizaineriem un projektētājiem pie tā, kas ir nepieciešams spēlei no mākslinieciskā viedokļa.

Savā darbā spēles projekta ietvaros mākslinieki var izmantot kā papīru un zīmuli, tā arī profesionālus programmatūras rīkus darbam ar 2D vai 3D grafiku.

1.2.5. Spēles audio daļas veidošana

Spēles audio var tikt iedalīts trīs daļās [1] – skaņas efekti, muzikālais pavadījums un balss ieraksti, kas atbilst personāžiem vai sižeta pasniedzējam.

Skaņu efektus var ierakstīt, piemēram, simulējot īstu priekšmetu mijiedarbības rezultāta radošos skaņu. Piemēram, lai ierakstīt mašīnas dzinēja startēšanas skaņu, var startēt kādu mašīnu, ierakstot rodošos skaņu vai, lai ierakstīt zobenu sitienu pa vairogu, var iesist ar zobenu pa vairogu un ierakstīt skaņu.

Skaņu efekti ir ļoti nozīmīga komponente spēles uztveršanā no spēlētāja viedokļa, tādēļ tiem nepieciešams sniegt atbilstošu uzmanību spēles izstrādes procesā.

Muzikālo pavadījumu var sintezēt vai veikt tā ierakstu studijā. Ir vairāki veidi, kā mūzika tiek reprezentēta spēlē [1]:

- mūzika var būt atbilstoša apkārtējai videi, jo īpaši nesteidzīgos spēles etapos, kad mūzika pastiprinās estētisko spēles uztveršanu un radīs papildus atmosfēru;
- mūziku var iniciēt notiekošais spēlē, piemēram, stratēģijas spēlēs sākoties cīņai sāk skanēt atbilstoša karadarbībai mūzika;
- mūziku var izsaukt notikumi spēlē, piemēram, veicot kādu darbību ar nelielu muzikālu pavadījumu var apzīmēt tā veiksmīguma līmeni;
- muzikālais pavadījums ir noderīgs arī izvēlnes logā vai spēles titros.

Spēle, kuras pilna izspēlēšana varētu aizņemt apmēram 20 stundas, var saturēt apmēram 60 minūtes muzikālā pavadījuma [1].

Balss ieraksti, kas atbilst noteiktiem personāžiem padara spēli interaktīvu. Balss ierakstus parasti ierunā aktieri. Šādi arī tiek pastiprināta spēles atmosfēra un personāžiem tiek piešķirta individualitāte.

1.2.6. Testēšana

Spēles izstrādes projekta beigu daļā kvalitātes pārvaldība un kontrole spēlē īpaši nozīmīgu lomu. Testētāji sāk strādāt pie testēšanas tik līdz parādās kaut kas, ko var spēlēt [7].

Tā var būt spēles sākotnējā versija ar vienu spēlējamu līmeni vai neliels spēles fragments, kurā var veikt kādas apzinātas darbības. Testētāji var strādāt pie vairākām spēlēm vienlaicīgi.

Kad izstrāde tuvojas beigām bieži tiek piesaistīts liels daudzums testētāju. To uzdevums ir pārbaudīt jaunākās funkcionalitātes darbspēju, kā arī regresīvi testēt iepriekš veidoto funkcionalitāti. Testēšana ir jo svarīgāka, jo sarežģītāka ir spēle, jo sarežģītā spēlē pat nelielas izmaiņas var radīt graujošas sekas [7].

Projekta izstrādes noslēguma daļā ir lielākais visa projekta laikā testējamo lietu skaits. Ir izstrādāts liels daudzums jaunas funkcionalitātes, taču testētājiem ir ļoti svarīgi regresīvi notestēt arī agrāk radīto funkcionalitāti [7], jo to varēja ietekmēt pēdējās izmaiņas. Nereti tiek izdalīts liels daudzums testētāju uzmanības jaunākajām izstrādātajām lietām, bet iepriekš izstrādātā funkcionalitāte netiek pienācīgi notestēta [7].

Tādu un citu testēšanas nepilnību rezultātā ļoti bieži pie lietotājiem nonāk spēle ar kļūdām, kuras vēlāk, uzturēšanas etapā, tiek labotas, izlaižot spēles atjauninājumus. Pārāk liels atstāto kļūdu skaits var izraisīt neapmierinātību spēles lietotāju vidū.

1.3. Uzturēšanas etaps

Pēc tam, kad spēles izstrāde ir pabeigta un tā tiek izlaista tirgū, iestājas spēles uzturēšanas un atbalsta sniegšanas periods.

1.3.1. Spēles uzturēšana

Ja runa ir par parastu viena lietotāja spēli, tad uzturēšanas etaps principā nozīmē laiku, kad tiek izlaisti atjauninājumi, kas labo testēšanas laikā neatklātās kļūdas. Īpaši aktuāls tas ir spēlēm, kas tiek ražotas personālajiem datoriem, jo to konfigurāciju variantu daudzums neļauj pārbaidīt spēles pilnu savietojamību ar katru no tām. Vienkāršāk ir ar spēļu konsolēm, kur konfigurācijas dažādība ir stingri ierobežota [1].

Ja runa ir par daudzspēlētāju spēlēm (MMO), tad atbalsta perioda jēdziens ir apskatāms citos toņos. Tas ietver ne tikai kļūdu labošanu bet arī nepieciešamību pēc patstāvīgas attīstības un izmaiņām spēlē [1], jo šāda veida spēļu realizācijas specifika pieprasa patstāvīgu attīstību, citādi spēlētāju skaits kritīsies un spēle nenesīs ekonomisku labumu. Šādām spēlē, uzturēšanas etaps var būt garāks un darbietilpīgāks par izstrādes etapu [1] un tajā var ietilpt atsevišķu spēles saturisku papildinājumu izstrādes projekti.

Lai izlaist atjauninājumu, kas labo atrastās kļūdas, programmētāji sakrāj lielu daudzumu paziņojumu par kļūdām, izlabo tās visas un izlaiž vienotā atjauninājuma pakotnē [1]. Dažkārt atjauninājuma pakotnes var iekļaut jaunu funkcionalitāti, saturu vai ietekmēt spēles procesu [1].

2. IZSTRĀDĀJAMĀS SPĒLES PLĀNOŠANAS UN PROJEKTĒŠANAS ETAPA REZULTĀTI

2.1. Augsta līmeņa koncepcija

Kā apskatīts nodaļā 1.1.1., augsta līmeņa koncepcija pēc būtības atbild uz jautājumu: "Par ko ir šī spēle?" un ļauj īsi formulēt spēles koncepciju ļoti augstā, nedetalizētā līmenī.

Maģistra dara ietvaros izstrādājamai spēlei augsta līmeņa koncepcijas apgalvojums tiek definēts sekojoši:

"Izstrādājamā spēle ir stratēģijas spēle, kur spēlētājam jāizvēlas viena no divām karojošām frakcijām un jāuzvar karā no vairākām pieaugošām pēc grūtības misijām otra frakcija. Darbība notiek izdomātā pasaulē, kurā karo orki un cilvēki."

2.2. Spēles priekšlikums

Tā kā izstrādājamā spēle ir individuāla izstrādātāja projekts un tai nav ekonomiskā pamatojuma, bet ir akadēmiskais pamatojums, tad zemāk aprakstītais spēles priekšlikums nav tipisks spēļu projektiem, bet ir piemērots pēc jēgas konkrētajam izstrādes projektam:

"Spēles izstrādes jēga nav ekonomiska labuma gūšana, bet ir pētījuma veikšana, kurā spēles izstrādes process ir viens no centrālajiem pētījuma objektiem. Spēles izstrādes nepieciešamība tiek pamatota ar nepieciešamību iepazīt un izpētīt spēles izstrādes procesa visus posmus, iepazīt un izpētīt uz praktiska izmantojuma pamatā LibGDX ietvaru, kā arī apskatīt mazāk dziļā līmenī citus spēļu radīšanas rīkus un gala rezultātā iegūt zināšanas par spēļu izstrādi, kā arī apkopot šīs zināšanas maģistra darba ietvaros, veikt to analīzi, salīdzinājumu ar teorētiskajām pieejamajām zināšanām šajā jautājumā un veikt attiecīgus secinājumus."

2.3. Spēles kopējā koncepcija

2.3.1. Augsta līmeņa koncepcija

Augsta līmeņa koncepcija definēta nodaļā 2.1,

2.3.2. Spēles žanrs

Stratēģija reālajā laikā (RTS).

2.3.3. Spēles procesa apraksts.

Spēlētājs vada pakļautībā esošas izvēlētās frakcijas dzīvās vienības skatā no augšas. Galvenais kontroles rīks ir pele, ar kuru var pārvietoties pa karti novedot to pie ekrāna malām, kā arī izvēlēties pakļautībā esošās vienības un nodot tām komandas. Pele ļauj arī iegūt informāciju par citiem objektiem, izvēloties tos kā arī pārvaldīt vadības paneļus.

Otrs vadības rīks ir klaviatūras bultiņu taustiņi un to funkcionalitāti dublējošie WASD taustiņi, kuri ļauj pārvietoties pa karti, kas dublē peles pievešanu pie kādas no malām. Klaviatūras ciparu taustiņiem var piesaistīt vienību formēšanu un ātru izsaukšanu.

Spēlētājam ir pieejams vadības un informācijas paneļi, kā arī minikarte vienā no ekrāna stūriem. Informācijas panelī atrodas informācija par spēlētāja rīcībā esošajiem līdzekļiem. Vadības panelis ļauj izvēlēties komandas izvēlētajai vienībai kā arī demonstrē tās sastāvu.

Galvenais spēlētāja uzdevums katrā no kartēm ir vadīt savā rīcībā esošās frakcijas vienības ar mērķi veikt ekspansiju kartē un pretinieka datorspēlētāja vai cita spēlētāja vadītās frakcijas likvidēšanu.

Spēlētājam ir pieejami resursi, tādi kā zelts, mežs, akmens un citi. Resursi nepieciešami ēku celšanai, kurās savukārt var veidot jaunas vienības vai veikt uzlabojumus. Citi ēku tipi ir aizsargēkas, kas veic uzbrukumu pretiniekiem no vietas.

Spēlētāja rīcībā ir karojošas vienības – dzīvie spēki, tehnikas vienības un resursu ieguves vienības.

2.3.4. Spēles konfigurācija

Spēle tiek pozicionēta kā “vecā tipa” reālā laika stratēģijas spēle mūsdienīgā realizācijā.

Spēlē tiek pārstāvētas divas pieejamas spēlētājam frakcijas, kas ir konflikta puses. Viena no tām cilvēki, otra orki. Abas frakcijas apveltītas ar paritējošām vienībām un tehnoloģijām, taču tām ir savas īpatnības.

Abu frakciju sastāvā ir dzīvās vienības un ēkas. Abu frakciju sastāvā ir resursu iegūšanas un ēku celšanas vienība, tuvcīņas karotāja vienība, tālcīņas karotāja vienība, īpaši tālas darbības cīņas vienība, maģiska vienība, trīs tipu gaisa vienības, četru tipu ūdens vienības. Detalizētāka konfigurācija tik izstrādāta un noformēta dizaindokumentā.

Spēles saturs sastāv no vienspēlētāja kampaņas par jebkuru no frakcijām. Spēlētāja uzdevums ķēdē no desmit secīgām misijām uzvarēt pretinieka frakciju. Katrai frakcijai ir sava unikāla kampaņa.

2.3.5. Spēles stāsts

Spēles sižeta pamatā ir karš starp savā starpā naidīgi noskaņotām orku un cilvēku rasēm. Priekšstāsts nosaka karadarbības ģeogrāfiju un mērķus katras puses kampaņai.

Ir pagājuši 500 gadi kopš cilvēku karaļvalstīs pēdējo reizi redzēja orkus. Pirms 500 gadiem tie zaudēja apvienotajiem cilvēku spēkiem un tika izdzīti pāri necaurejamajiem kalniem tālu uz ziemeļiem. Pēc šīs uzvaras cilvēku karaļvalstis norobežoja no ziemeļiem maģiska barjera, par kuras uzturēšanu atbildēja magu ordenis. Taču nezināmu iemeslu dēļ magu ordeņis pēkšņi pazuda komā ar maģisko barjeru.

Orku sirojumi cilvēku zemēs kļūst arvien biežāki un izlūki ziņo par iebrukuma gatavošanos.

2.3.6. Spēles mērķauditorija

Spēles mērķauditorija sastāv no divām potenciālo lietotāju grupām.

Pirmā grupa ir iepriekšējās paaudzes spēlētāji, kuri ir noilgojušies pēc "vecā tipa" spēlēm, bet agrāka ražojuma spēles nevēlas spēlēt, jo tās jau ir izspēlētas vai nav iespējas tās iegūt. Izstrādājamā spēle būs kā kaut kas līdzīgs jau zināmajam, bet jaunā realizācijā un ar jaunu saturu.

Otrā grupa ir jaunie spēlētāji, kurus var piesaistīt klasiska stratēģijas spēle uz mobilās platformas. Android ierīcēm nav pieejams liels skaits šāda tipa spēļu, mobilo spēļu tirgus attīstās citā virzienā, taču eksistē spēlētāju grupas, kas labprāt spēlētu klasisku "vecā tipa" reālā laika stratēģiju uz mobilās iekārtas, savukārt šī žanra spēles ir vāji pārstāvētas mobilo iekārtu spēļu tirgū.

2.3.7. Atbalstāmās platformas

Spēli paredzēts darbināt uz Windows un Linux darbstacijām, kā arī mobilajām platformām – viedtālruņiem un planšetdatoriem ar Android operētājsistēmas vadību.

Spēlei jādarbojas uz Windows 10 OS, Linux Mint 18.1, Ubuntu 16.04 LTS, Android 4.0.3 un jaunākām versijām.

2.3.8. Paredzētais realizācijas laiks

Spēlei atvēlētais realizācijas laiks sākas 2017. gada 1. februārī un beidzas 2017. gada 17. novembrī. Tādējādi uz realizāciju tiek atvēlēti astoņi ar pusi mēneši. Šā termiņa laikā nepieciešams izstrādāt dizaindokumentu, kurš izstrādes laikā tiks aktualizēts, kā arī izstrādā spēles kodu, grafiku, skaņas, kartes, kampaņas un to saturu.

Izstrādi plānots veikt individuāli, iteratīvi, dinamiski.

Maģistra darba ietvaros paredzēts apskatīt spēles projektēšanas un plānošanas etapa rezultātus kā arī spēles projektējumu un agrīnās prototipēšanas etapa rezultātus, kuri tiks izstrādāti un veikti balstoties un ņemot vērā vairākplatformu atbalsta problemātikas risinājumu ietekmi un libGDX sniegtās iespējas izstrādes procesa veikšanā.

2.3.9. Tirgus analīze

Personālo datoru spēļu tirgū ir liels daudzums dažādos gados izlaistu līdzīgu pēc būtības spēļu. Izstrādājamā spēle var būt interesanta ar jaunu sižetu, modificētu mehāniku, mūsdienīgu realizāciju.

Mobilo iekārtu spēļu tirgū praktiski nav analoģisku spēļu, kas, iespējams, pamatojams ar pārāk sarežģītu mobilajām iekārtām lietotāja saskarnes un spēles vadības sistēmu kā arī ar šāda žanra un tipa spēles neatbilstību mobilo iekārtu auditorijas vajadzībām.

Neskatoties uz komerciālu nepamatotību, spēle tiks izstrādāta, jo tās izstrādes galvenais virzītājspēks ir nevis ekonomiska rakstura, bet pētnieciska un akadēmiska rakstura.

Viens no maģistra darba izaicinājumiem ir noskaidrot, vai izmantojot libGDX individuālam izstrādātājam izdosies izveidot pēc kvalitātes un spēlējamības līdzvērtīgu spēli tirgū esošajām spēlēm.

2.3.10. Izstrādes komanda

Spēli paredzēts izstrādāt individuāli, detalizēti izpētot programmētāja, mākslinieka, skaņas inženiera, projektētāja un testētāja lomas un veicot visu šo lomu pienākumus projekta realizācijas ietvaros.

2.4. Spēles dizaindokumenta izstrāde

Spēles dizaindokumenta izstrāde ir īpaši laikietilpīgs un sarežģīts process, tādēļ tā izstrādes procesu plānots optimizēt spēles izstrādes ietvaros ar mērķi padarīt to maksimāli efektīvu un atbalstošu izstrādes laikā un novērst lieku apgrūtinājumu, ko varētu izraisīt nepieciešamība pēc tā formālas izstrādes pirms izstrādes sākuma.

Spēles izstrādi plānots veikt uz kopējās koncepcijas un prototipēšanas rezultātu pamata, paralēli veicot padarītā dokumentēšanu dizaindokumenta veidā, analīzi, uzlabošanu un dokumentētā aktualizāciju, respektīvi dizaindokumenta aktualizāciju, tādējādi šī nelielā pēc pieejamajiem izstrādes resursiem, individuālā projekta laikā tiks gan izveidots kvalitatīvs, atbilstošs spēles stāvoklim dizaindokuments, kas ļaus pārvaldīt un apzināties spēli visas izstrādes laikā, gan kalpos par pamatu spēles konfigurācijas uzlabošanai nākotnē.

Spēles dizaindokumenta galējo versiju plānots izveidot līdz spēles izstrādes beigām un pievienot maģistra darbam kā pielikumu.

2.5. Spēles prototipu izstrāde

Spēles prototipu izstrāde agrīnajā izstrādes stadijā notiek fiziska spēles ekrānu prototipēšana papīra skiču formā, kas ļauj iztēloties, kādi būs spēlē pieejamie ekrāni un kā potenciāli izskatīsies lietotāja saskarne.

Nākotnē tiek plānots izmantot protitipēšanu karšu radīšanā, funkcionalitātes izmēģināšanā iekš spēles, u.t.t. Maģistra darba nodaļā 5.3 ir apskatāmi agrīnās prototipēšanas etapa rezultāti.

3. VAIRĀKPLATFORMU SPĒLES IZSTRĀDES IETVARI

Šīs nodaļas ietvaros tiek definētas galvenās prasības pret ietvaru, kurš tiks izmantots izstrādājamās spēles radīšanā. Tiek veikts vairāku ietvaru apskats un pamatota LibGDX ietvara izvēle, kā arī analizēta alternatīvu iespēja.

3.1. Prasības pret apskatāmajiem ietvariem

Ietvaram, kurš tiks izmantots 2. nodalā aprakstītās spēles izstrādei ir jāatbilst vairākām nozīmīgām prasībām, kuras uzskaitītas un definētas zemāk:

1) Vairākplatformu atbalsts. Nepieciešamās platofrmas: personālās darbstacijas, mobilās iekārtas – planšetdatori, mobilie telefoni. Atbalstāmās operētājsistēmas: Windows, tiks testēta versija Windows 10, Linux, tiks izstrādāts un testēts distributīviem Linux Mint 18.1 un Ubuntu 16.04 LTS, Android mobilo iekārtu OS.
2) Ērta izstrādes vide, kas ļauj koncentrēties uz spēles izstrādi vairākām platformām bez nepieciešamības veikt smagu konfigurēšanu.
3) Ērta koda pārnesamība starp platformām, bez nepieciešamības pielāgot vai mainīt kodu kādai no platformām.
4) Viegli pieejama pēc iespējas pilna, kvalitatīva dokumentācija, kā arī lietojumgadījumu demonstrējumi un instrukcijas.
5) 2D grafikas atbalsts, pēc iespējas ērtāka iespēja strādāt ar dalāmu sektoros(rūtiņās) lokācijas karti un objektiem tajā. Savietojams ar ietvaru gatavs šāda veida karšu redaktors.
6) 2D grafikas tekstūru, 2D objektu un animācijas veidošanas pēc iespējas ērts atbalsts.
7) Ietvara licencei jābūt bezmaksas vismaz nepieciešamās funkcionalitātes ietvaros, kā arī jāļauj bezmaksas publicēt izstrādājamo spēli komercnolūkos.

3.2. Ietvaru pārskats

Autors ir izvēlējies sarakstu ar ietvariem, kas pēc pirmās apskates šķiet piemēroti definētajam uzdevumam. Šīs nodaļas ietvaros notiks izvēlēto ietvaru pārskats ar mērķi noteikt to atbilstību prasībām, kas definētas nodaļā 3.2., kā arī lai noteikt ietvaru stiprās un vājās puses, kas arī var ietekmēt rīka izvēli uzdevuma veikšanai.

3.2.1. LibGDX Java ietvars

LibGDX ir Java valodā spēļu izstrādes ietvars, kurš nodrošina vienotu API darbam ar visām atbalstāmajām platformām [8].

Izstrādājot projektu neatkarīgi no mērķplatformām, izstrādes procesā produkta testēšanu var viegli un ātri veikt uz izstrādes darbstacijas [8].

LibGDX filozofija neuzspiež programmatūras projekta struktūru vai dizainu. Izstrādātājiem ir izstrādes brīvība šajos jautājumos, kā arī iespējas izvēlēties izmantojamās komponentes [8].

3.2.1.1. Platformu un operētājsistēmu atbalsts

Ietvara sniegtais platformu atbalsts ir maksimāli plašs [8]:

- Windows
- Linux
- Mac OS X
- Android (2.2+)
- BlackBerry
- iOS
- Java Applet
- Javascript/WebGL

3.2.1.2. Izstrādes vide

Darba procesā ar libGDX ietvaru iespējams veiksmīgi izmantot Eclipse, Intellij IDEA, NetBeans izstrādes vides [8]. Jāpiebilst, ka, ja ir vēlme izmantot Intellij IDEA vidi, tad tās bezmaksas “Community” versijā nebūs iespējams sagatavot projektu darbam Javascript/ WebGL vidē, jo šīs vides bezmaksas versijā nav GWT atbalsta [9].

3.2.1.3. Koda pārnesamība

Koda pārnesamība ir maksimāli optimāla. Lai darbināt kodu jebkurā no atbalstāmajām platformām, kodā nav nepieciešams veikt nekādas izmaiņas [8].

3.2.1.4. Dokumentācija

Apskatāmajam ietvaram pieejama pilna dokumentācija angļu valodā [8], daudzi izstrādes piemēri, labi dokumentēta projekta izveides un pabeigšanas procedūra. Ietvaram ir plašs lietotāju loks un daudzi lietujumpiemēri angļu un krievu valodās.

3.2.1.5. Rūtiņu kartes un 2D grafikas atbalsts

Apskatāmais ietvars ir savietojams ar TMX tile map [8], kuras iespējams radīt atbilstošu karšu redaktorā Tiled.

LibGDX izmanto OpenGL funkcionalitāti un ļauj darboties ar to kā zemā tā augstā līmeni, izmantojot iebūvētās klases darbam ar ortogrāfisko kameru, tekstūru atlasiem, bitmap fontiem, 2D daļiņu sistēmu, 2D saskarnes bibliotēku, u.c. [8].

3.2.1.6. Licence

LibGDX ir atvērtā pirmkoda projekts ar Apache 2.0 linceci [8]. LibGDX drīkst brīvi un bezmaksas izmantot gan nekomerciālos, gan komerciālos projektos, gan jebkādos citos projektos [8].

3.2.2. Godot Engine

Spēļu izstrādes rīks, kurā programmēšana notiek C++ un GDScript valodās, kur pēdējā ir speciāli izstrādāta šim ietvaram [10].

3.2.1.1. Platformu un operētājsistēmu atbalsts

Ietvara sniegtais platformu atbalsts ir maksimāli plašs [10]:

- mobilās platformas: iOS, Android, BlackBerry OS;
- darbstacijas ar sekojošu OS saimēm: Windows, OS X, Linux, *BSD, Haiku
- tīmekļa platforma: HTML5 (Emscripten)

3.2.1.2. Izstrādes vide

Godot Engine izmanto specializētu, domātu šim izstrādes rīkam izstrādes vidi. Izstrādes vide darbināma Windows, Linux, OS X sistēmās [10].

Izstrādes vidē ir iebūvēts vizuāls satura redaktors, kurš var tikt lietots testēšanas laikā saglabājot izmaiņas reālajā kodā bez nepieciešamības pārstartēt testēšanas vidi [10].

3.2.1.3. Koda pārnesamība

Koda pārnesamība ir maksimāli optimāla. Izstrādājamais kods ir darbināms jebkurā no atbalstāmajām platformām bez izmaiņu veikšanas tajā vai atsevišķas pielāgošanas [10].

3.2.1.4. Dokumentācija

Ietvaram ir laba dokumentācija angļu valodā ar lietošanas piemēriem [11], kā arī plašs lietotāju loks, kas nodrošina iespēju operatīvāk veikt problēmsituāciju risinājumu meklēšanu.

3.2.1.5. Rūtiņu kartes un 2D grafikas atbalsts

Rīka izstrādes vidē ir iebūvēts tile map redaktors. Ietvars atbalsta 2D grafiku, animācijas, ir 2D sadursmju dzinis, kurš strādā pikseļu koordinātu plaknē [11].

3.2.1.6. Licence

Godot Engine ir atvērtā pirmkoda projekts ar MIT licenci [10]. To var brīvi un bezmaksas izmantot jebkāda veida projektiem – gan komerciāliem, gan nekomerciāliem. Tiesības uz izstrādāto spēli ir pilnībā spēles izstrādātāju rīcībā [10].

3.2.3. Polycode

Polycode ir C++ un Lua valodā ietvars, kas paredzēts lai vaidot interaktīvas apliācijas, ieskaitot spēles [12]. Tas ir bezmaksas, atvērtā pirmkoda un vairākplatformu [12].

3.2.1.1. Platformu un operētājsistēmu atbalsts

Polycode kodols ir programmēts valodā C++ un programmatūra, kas tiek izstrādāta izmantojot šo ietvaru ir darbināma uz personālajiem datoriem ar Windows, Mac vai Linux OS [12]. Mobilo platformu atbalsts ar Android un iOS operētājsistēmām uz doto brīdi nav, bet tiek plānots to realizēt [12].

Respektīvi tiek atbalstītas tikai personālās darbstacijas ar operētājsistēmām Windows, OS X vai Linux.

3.2.1.2. Izstrādes vide

Polycode ir izveidota speciāla izstrādes vide ar iebūvētu vizuālo redaktoru un koda rediģēšanas rīku, taču Polycode ir izmantojams arī bez tā, kā bibliotēka un to var lietot, strādājot jebkurā ērtā C++ vai Lua vidē [12].

3.2.1.3. Koda pārnesamība

Visa platformu specifiskā funkcionalitāte ir abstragēta atsevišķi no lietotāja un nav nepieciešams veikt jebkādas izmaiņas radītajā programmas kodā, lai to kompilēt uz jebkuras no atbalstāmajām platformām [12].

3.2.1.4. Dokumentācija

Polycode ir pieejama laba un pilnīga dokumentācija angļu valodā [13]. Ietvaram ir plašs lietotāju skaits un problēmsituācijas ir risināmas dinamiskā ceļā, izmantojot dokumentāciju un risinājumus internet vidē.

3.2.1.5. Rūtiņu kartes un 2D grafikas atbalsts

Ir iespējams strādāt ar Tiled rīkā radītajām kartēm, bet mazāk augsta līmeņa gatavas funkcionalitātes salīdzinājumā ar citiem potenciālajiem ietvariem. 2D un 3D grafikas un animāciju atbalsts [13], izstrādes vidē pieejams vizuāls redaktors grafiskiem skatiem [12, 13].

3.2.1.6. Licence

Polycode ir atvērtā pirmkoda brīvi un bezmaksas izmantojams jebkāda veida projektiem [12]. Polycode izmanto MIT licenci savā izplatīšanā [12].

3.2.4. LÖVE

LÖVE ir atvērtā pirmkoda 2D spēļu izstrādes ietvars valodā Lua, tas ir bezmaksas, brīvi lietojams jebkāda veida projektos [14].

3.2.1.1. Platformu un operētājsistēmu atbalsts

Tiek atbalstītas sekojošas platformas [14]:

- Windows;

- Mac OS X;
- Linux;
- Android;
- iOS;

3.2.1.2. Izstrādes vide

LÖVE piedāvā strādāt speciāli sagatavotā izstrādes vidē, kas darbojas Windows XP vai jaunākā vidē, Ubuntu Linux 14.04 līdz 16.10 vidē, Mac OS X 10.7 vai jaunākā vidē [14].

3.2.1.3. Koda pārnesamība

Rīkā veidoto kodu nav nepieciešams mainīt atkarībā no izvēlētās platformas, viens kods tiek kompilēts visām atbalstāmajām platformām [14].

3.2.1.4. Dokumentācija

Atšķirībā no iepriekš apskatītajiem ietvariem, LÖVE ir vājāka dokumentācija [15]. Tās apskate uzrādīja vairākas lapas, kurām nepieciešams papildinājums kā arī funkcionalitātes apraksts ir nabadzīgāks [15]. Lietotāju kopiena ir mazāka nekā iepriekš apskatītajiem ietvariem, kas samazina iespējas uz efektīvu atbalstu kopienas līmenī problēmsituāciju risināšanā.

3.2.1.5. Rūtiņu kartes un 2D grafikas atbalsts

LÖVE ir viss nepieciešamais darbam ar 2D grafiku [14, 15], bet nav izdevies noskaidrot, cik veiksmīgs ir rūtiņu karšu atbalsts un kāda ir savietojamība ar Tiled rīkā veidotajām kartēm, spriežot pēc dokumentācijas nekādas papildus funkcionalitātes šajā virzienā nav sagaidāmas.

3.2.1.6. Licence

LÖVE ir zlib/libpng licence [14], kas nozīmē, ka tas ir bezmaksas lietošanā un to drīkst brīvi un bezmaksas lietot jebkādiem – gan komerciāliem, gan nekomerciāliem projektiem [14].

3.3. Pārskata kopsavilkums, ietvara izvēle un tās tehniskais pamatojums

Kad ir apskatīti iespējamie rīki spēles izveidei un ir nodefinēts uzdevums, ir nepieciešams veikt izvēli – kura rīka izmantošana būs piemērotāka risināmajam uzdevumam. Lai to izdarīt, šīs nodaļas ietvaros tiks veikts neliels kopsavilkums, kā arī tiks pamatota ietvara izvēle.

3.3.1. Ietvaru pārskata analīze un kopsavilkums

Lielākā daļa apskatīto ietvaru atbalsta visas nepieciešamās platformas – gan Windows un Linux darbstacijas, gan mobilās iekārtas ar Android operētājsistēmu. Tas nav pārsteidzoši, jo rīki tika meklēti un izvēlēti kā ietvari vairākplatformu izstrādei.

Izņēmums ir Polycode, kurā vēl nav realizēts mobilo platformu atbalsts, bet tas ir plānos. Protams, tā kā izstrāde notiks jau tagad, tad gaidīt nav laika un izvēlei jānotiek starp pārējiem rīkiem.

Daļai no rīkiem, piemēram Godot Engire ir pieejama speciāla izstrādes vide ar spēcīgu vizuālo redaktoru, kas ir vilinošu, kā arī šajā pašā rīkā tiek izmantots speciāli spēļu radīšanai ar šo rīku domāta programmēšanas valoda. Tā ir augsta līmeņa un tai ir līdzība ar Python valodu.

Visi apskatītie rīki atbalsta darbu ar 2D grafikas tehnoloģijām, lielākā daļa sniedz iespēju strādāt arī ar 3D grafiku, to skaitā libGDX, Godot un citi. Visi rīki ļauj darbodies ar rūtiņu kartēm, pārsvarā ir iespējams strādāt ar Tiled rīkā radītajām kartēm vai arī ir iebūvēts redaktors.

Lielākajai daļai rīku ir laba dokumentācija, taču citiem ir pieejams lielāks kopienas atbalsts, piemēram libGDX, kā arī Godot. Rīks libGDX atšķiras ar diezgan plašu lietojumpiemēru skaitu, kas brīvi pieejami tīmeklī un ir vērtīgs palīgmateriāls darbā ar šo izstrādes rīku.

3.3.2. Ietvara izvēle un izvēles tehniskais pamatojums

Jāatzīst, ka autora izvēle bija zināma vēl līdz maģistra darba izstrādes sākumam, un, lai arī pētījuma veikšanas laikā daži no rīkiem ļoti ieinteresēja autoru, piemēram, Godot Engine, kurš izskatās kā ļoti ērts izstrādes rīks no pirmās apskates pozīcijas, taču, ņemot vērā vairākus

apsvērumus, autora izvēle nav mainījusies, un, par izstrādes rīku izstrādājamajai spēlei, autors izvēlas LibGDX.

Tas ir pamatojams ar to, ka autors ir pārliecināts par šī rīka pilnu atbilstību visām prasībām, jo pēc tā dokumentācijas pētījumiem ir skaidrs, ka viss iecerētais ir realizējams izmantojot šo rīku. Tam ir viss nepieciešamais darbam ar audio, 2D grafiku, tajā ir iespējas izmantojot iebūvēto funkcionalitāti radīt vairāklietotāju režīmu, kurš ir ļoti vēlams izstrādājamās spēles žanra spēlēs, jo kampaņa agri vai vēlu tiks izieta, bet vēlāk var izklaidēties sacenšoties ar citiem spēlētājiem atsevišķās kartēs, līdzīgi kā tas notiek citās analoģiskās spēlēs.

LibGDX ne tikai ļauj sasniegt visus izvirzītos mērķus, bet ir arī ērts izmantošanā izstrādes procesā. To iespējams visā pilnībā testēt izstrādes laikā bez nepieciešamības izmantot ārējas iekārtas vai virtuālās mašīnas, kas paātrina katras izstrādes iterācijas veikšanu. LibGDX nav vienīgais rīks, kas piedāvā ērtu testēšanas vidi.

Viens no svarīgākajiem argumentiem par labu LibGDX izvēlei ir autora iepriekšējā minimāla, bet tomēr pieredze darbā ar šo ietvaru, kas nozīmē nelielu, bet izstrādes laika optimizāciju uz tā rēķina, ka nepieciešams apgūt mazāk nepazīstamu tehnoloģiju, analoģiski var izteikties par valodu Java. Tehnoloģiju pazīstamība ir nopietns arguments kāda rīka izvēlē ne tikai spēļu izstrādē, bet arī daudzos citu veidu projektos.

4. LIBGDX IETVARS

LibGDX ir JAVA spēļu izstrādes ietvars, kurš nodrošina vienotu lietojumprogrammas saskarni, kura strādā ar visām atbalstāmajām platformām. Tas nozīmē, ka veidojot vienu programmatūras kodu, to iespējams veiksmīgi izpildīt jebkurā no atbalstāmajām mērķa platformām.

Ietvars nodrošina vidi ātrai prototipēšanai un ātrām iterācijām. Tā vietā, lai veikt pilnu izstrādātā koda izpildi un testēšanu uz paredzētajām platformām pēc katru izmaiņu veikšanas, iespējams, ar pārliecību, ka visās atbalstāmajās platformās izstrādātā funkcionalitāte strādās analoģiski, veikt izmaiņu testēšanu izstrādājamajā vidē.

LibGDX koncepcija ir neuzspiest izstrādātājiem savu struktūru un funkcionalitāte, ietvars ļauj dinamiski un pēc nepieciešamības pieslēgt un izmantot piedāvājamo funkcionalitāti pēc modularitātes principiem.

Šīs nodaļas ietvaros tiek apskatītas visas LibGDX ietvara piedāvātās iespējas un risinājumi, ko iespējams efektīvi izmantot kopā ar LibGDX ietvaru.

4.1. Vairākplatformu atbalsts

Vienota API saskarne darbam ar sekojošām platformām [8]:

- Windows, Linux, Mac OS X darbstacijas;
- Android iekārtas sākot ar versiju 2.2;
- BlackBerry;
- iOS;
- Java sīklietotne;
- Javascript/WebGL (Chrome, Safari, Firefox, IE).

4.2. Integrācija ar ārējiem rīkiem

LibGDX ietvaru iespējams integrēt mijiedarbībai ar sekojošiem trešo pušu rīkiem[8]:

- Spine 2D animācijas rīks;
- Nextpeer – daudzspēlētāju režīma implementācija;
- Saikoa – ProGuard un DexGuard spēles aizsardzības pret atgriezenisko inženieriju un citām uzlaušanas un pirātisma metodēm;

4.3. Audio

LibGDX ietvars nodrošina iespēju darbam ar mūzikālo un skaņas pavadījumu, kā arī skaņas efektu izmantošanu. Iespējams izmantot tiešo piekļuvi iekārtas skaņas ierakstīšanas iekārtai un veikt skaņas ierakstīšanu, taču jāņem vērā, ka šī funkcionalitāte netiek atbalstīta tīmekļa Javascript platformā, kur nav iespējams iegūt tiešu piekļuvi skaņas ierakstīšanas iekārtai.

Darbā ar skaņas datnēm tiek atbalstīti sekojoši datu formāti: WAV, MP3 un OGG [8].

4.4. Matemātika un fizika

LibGDX piedāvā izmantot matemātiskām un ar spēles fizikas aprēķiniem saistītām operācijām sekojošus rīkus un predefinētas klases [8]:

- klases darbam ar matricām, vektoriem un kvaretnioniem. Matricu un vektoru operācijas ir pēc iespējas paātrinātas izmantojot C valodā rakstītu kodu;
- klases darbam ar ģeometriskām formām, tādām, aplis, nošķelta piramīda, virsma, brīvformas līkne, poligons, stars, četrstūris, sfēra;
- klase darbam ar Katmula-Roma līknēm;
- klase darbam ar biežāk lietojamajām interpolācijām;
- klase darbam ar ieliekta daudzstūri un tā triangulāciju;
- klase, kas piedāvā virkni ar metodēm krustošanās un pārklāšanās pārbaudei starp dažādiem ģeometriskiem objektiem;
- 2D objektu fizika: JNI (Java Native Interface) apvalks darbam ar Box2D fizikas dzini, kurš programmēts valodā C++;
- 3D objektu fizika: JNI apvalks darbam ar 3D objektu fizikas dzini Bullet Physics, kurš programmēts valodās C un C++;

4.5. Kontrole un ievadīšana

LibGDX ietver abstrakcijas darbam ar klaviatūras, peles un skārienjūtīgās virsmas ievadi, kā arī iekārtas akselerometru un kompasu. LibGDX žestu noteikšanas implementācija ļauj noteikt sekojošus žestus, ievadītus, izmantojot skārienjūtīgās virsmas ievades ierīci [16]:

- pieskāriens (touchDown metode) – lietotājs pieskaras ekrānam;
- ilgs pieskāriens (longPress metode) – lietotājs tur piespiestu pirkstu ekrānam ilgāku laiku;
- viegls sitiens (tap metode) – lietotājs viegli uzsit pa ekrānu ar pirkstu, uzreiz to paceļot no ekrāna, pie tam, pirksts nedrīkst veikt kustību ārpus sākotnējā pieskāriena laukuma, vairāki secīgi pieskārieni tiks reģistrēti, ja lietotājs tos veiks noteiktā laika intervālā pēc kārtas;
- vilkšana (pan metode) – lietotājs velk ar pirkstu pāri ekrānam, tiks noteiktas aktuālās pieskāriena koordinātas un atšķirība starp esošajām un iepriekšējām koordinātām; noderīgi, piemēram, kameras grozīšanas realizācijai;
- vilkšanas beigas (panStop metode) – nostrādā, kad tiek pārtraukta pirksta pārvietošana pa ekrānu;
- pirksta vilkšana ar atraušanu no ekrāna (fling metode) – lietotājs velk pirkstu pāri ekrānam un pēc apstāšanās pirksts tiek atrauts no ekrāna;
- tuvināšana/tālināšana (zoom metode) – lietotājs novieto divus pirkstus uz ekrāna un satuvina vai attālina tos, tiks noteikts sākotnējais un aktuālais attālums starp pirkstiem; noderīgi veicot kameras pietuvināšanu vai attālināšanu;
- tuvnāšana/tālināšana un rotācija (pinch metode) – darbojas analoģiski zoom metodei, bet tiks iegūti nevis attālumi starp pirkstu pieskāriena vietām, bet to konkrētas pozīcijas; pielietojams kā zoom metode, bet iespējams izmantot arī sarežģītāku žestu, piemēram, rotācijas noteikšanai;

4.6. Operācijas ar datnēm un datu glabāšana

Darbs ar datnēm katrā no LibGDX atbalstāmajām platformām notiek atšķirīgi. Šajā nodaļā plānots apskatīt iespējas darbā ar datnēm kā arī atšķirības, kuras jāņem vērā izstrādājot programmatūru atšķirīgām platformām, izmantojot libGDX.

LibGDX modulis, kurš atbild par darbu ar datnēm nodrošina sekojošu funkcionalitāti [16]:

- lasīšana no datnes;
- ierakstīšana datnē;
- datnes dublēšana;
- datnes pārvietošana;

- datnes dzēšana;
- datņu un direktoriju saraksta apskate;
- pārbaude vai datne vai direktorija eksistē.

4.6.1. Atbalstāmo platformu failu sistēmu īpatnības

Šajā nodaļā tiek apskatītas LibGDX atbalstāmo platformu failu sistēmu paradigmas un darbības īpatnības.

4.6.1.1. Darbstacijas (Windows, Linux, Mac OS X)

Darbstacijās, kas darbojas ar atbalstāmajām operētājsistēmām, failu sistēma ir viens liels atmiņas apgabals. Datnes var tikt sasniegtas izmantojot relatīvo attiecībā pret tekošo direktoriju ceļu vai izmantojot absolūtās adreses. Datnes un direktorijas parasti iespējams lasīt un rakstīt jebkurai lietotnei, izņemot, protams, gadījumus, ja darbība tiek liegta izmantojot tiesību ierobežojumus.

4.6.1.2. Android / iOS

Android iekārtās situācija ir nedaudz sarežģītāka. Datnes iespējams glabāt iekš lietotnes APK pakotnes kā lietotnes resursus. Šīs datnes ir iespējams tikai lasīt. Dotās datnes, protams, ka ir pieejamas tikai lietotnei, kurai tās pieder.

Datnes iespējams glabāt arī iekšējā atmiņā, kur tās var gan lasīt, gan rakstīt. Lietotnei Android sistēmās tiek izdalīta atsevišķa direktorija, kurai piekļuve ir tikai dotajai lietotnei. Šo direktoriju drīkst uztvert par lietotnes privāto darba telpu atmiņā.

Visbeidzot, datnes iespējams glabāt ārējā atmiņas iekārtā, piemēram, SD atmiņas kartē. Jāņem vērā, ka šāda atmiņa var ne vienmēr būt pieejama, piemēram, lietotājs to var izņemt no iekārtas. Datnes, kuras tiek glabātas šādā atmiņas iekārtā, nepieciešams lietotnē uztvert kā potenciāli gaistošas, jeb tādas, kas var pazust. Kā arī nedrīkst aizmirst pievienot tiesību piekļūt arējai atmiņas iekārtai pieprasījumu projekta AndroidManifest.xml konfigurācijas datnē.

4.6.1.3. Javascript / WebGL

Javascript / WebGL lietotnei pēc noklusējuma nepiemīt failu sistēma klasiskā izpratnē. Tā vietā lietotnes resursi tiek sasniegti izmantojot vienoto resursu vietrādi un fiziski atrodas

uz kāda servera. Jaunākās pārlūkprogrammu versijas piedāvā lokālās glabātuves funkcionalitātes atbalstu, kas pēc izmantojamības ir līdzīga tradicionāli pieejamajai failu sistēmai, taču problemātika darbā ar šo lokālo glabātuvi ir tāda, ka pēc noklusējuma tā ir diezgan neliela izmēra un tā nav standartizēta, respektīvi, dažādās pārlūkprogrammās tā var būt dažāda izmēra un nav stingra veida, kā noteikt tās izmērus.

LibGDX ietvaros ir pieejama realizācija, kura ļauj strādāt Javascript / WebGL lietotņu izstrādes ietvaros ar abstrakciju, kas sniedz iespēju strādāt kā ar tikai lasīšanai paredzētu failu sistēmu [16].

4.7. Grafika

Renderēšana visās atbalstāmajās platformās notiek, izmantojot OpenGL ES 2.0. Šīs nodaļas ietvaros paredzēts apskatīt pieejamās iespējas darbā ar spēles grafisko attēlošanu [8].

4.7.1. Zema līmeņa OpenGL palīgi

LibGDX ietver palīgklases darbam ar OpenGL zemā līmenī. Klases sniedz abstrakcijas līmeni, kurš ļauj darboties ar izstrādi vairākām platformām paralēli, izstrādājot kodu vienu reizi visām sistēmām, kā arī padara darbošanos ērtāku [8].

Šajā nodaļā paredzēts apskatīt lielāko daļu no šāda veida klasēm.

4.7.1.1 Virsotņu masīvi un virsotņu buferu objekti

LibGDX piedāvā palīgklasi darbam ar OpenGL virsotņu masīviem. Klase nav savietojama ar OpenGL 3+ pamata profiliem. Savietojamībai, nepieciešams izmantot virsotņu bufera objekta klasi [17].

4.7.1.2. Režģi un tekstūras

Režģi satur virsotnes kas sastādītas no atribūtiem, kas specificēti virsotnes atribūtu klases instancē. Virsotnes tiek glabātas grafiskās kartes operatīvajā atmiņā virsotņu bufera objektu formā vai arī operatīvajā atmiņā virsotņu masīva formā. Pirmajam glabāšanas veidam ir priekšroka un tas tiek pielietots, ja vien iekārta to atbalsta [18].

Režģi tiek pārvaldīti automātiski. Ja OpenGL konteksts tiek zaudēts, visi virsotņu bufera objekti kļūst nederīgi un tos nepieciešams pārlādēt, kad konteksts tiek atjaunots. Šādi

var notikt tikai Android iekārtā, ja lietotājs pārslēdzas uz citu lietotni vai arī saņem ienākošo zvanu. Režģi tiek pārlādēti automātiski un nav nepieciešams to veikt manuāli [18].

Režģis sastāv no virsotnēm. Katra virsotne sastāv no atribūtiem, tādiem, kā pozīcija, krāsa vai tekstūru koordināta. No šiem atribūtiem obligāts ir tikai pozīcija. Katram atribūtam ir aizstājvārds, kurš tiek izmantots renderējot ar OpenGL ES 2.0. Aizstājvārds ir paredzēts, lai sasaistīt specifisku virsotņu atribūtu ar ēnotāja atribūtu. Ēnotājam un atribūta aizstājvārdam ir precīzi jāsakrīt [18].

Tekstūru palīgklase (Texture) ietīsta standarta OpenGL ES tekstūru. Tekstūru nepieciešams pārvaldīt. Ja OpenGL konteksts tiek pazaudēts, kas var notikt, ja lietotājs pārslēdzas uz citu lietotni, vai arī, ja tiek saņemts ienākošais zvans, tad tekstūra kļūst nederīga un to nepieciešams pārlādēt. Pārvaldītās tekstūras tiek pārlādētas automātiskā režīmā [19].

Tekstūru nepieciešams saistīt ar ģeometriju, izmantojot *GLTexture.bind()* metodi. Tekstūru nepieciešams izvietot no atmiņas, kad tā vairs nav nepieciešama [19].

4.7.1.3. Pikseļu ēnotāji

Pikseļu ēnotāji iekapsulē virsotni un ēnotāja fragmentu kā pārus, no kuriem tiks veidota ēnotāju programma (*ShaderProgram*). Pēc savas izveides, ēnotāju programma var tikt pielietota, lai zīmēt režģi. Lai piespiestu grafiskajam procesoram lietot noteiktu ēnotāja programmu, iespējams izmantot programmas *begin()* metodi, kas piesaistīs noteikto programmu [20].

Kad ēnotāja programma ir saistīta, iespējams uzstādīt parametrus un atribūtus, izmantojot atbilstošās metodes. Ēnotāja programmu iespējas atsaistīt izmantojot metodi *end()*, bet izvietot no atmiņas, izmantojot metodi *dispose()* [20].

Ēnotāja programmas tiek pārvaldītas. Gadījumā, ja tiek zaudēts OpenGL konteksts, tad visi ēnotāji kļūst nederīgi un tos nepieciešams pārlādēt. Šādi var notikt tikai Android iekārtā, ja lietotājs pārslēdzas uz citu lietotni vai arī saņem ienākošo zvanu. Ēnotāja programmas tiek pārlādēti automātiski, kad konteksts tiek atgriezts, un nav nepieciešams to veikt manuāli [20].

4.7.1.4. Tūlītēja režīma renderēšanas emulācija

Tūlītēja režīma renderēšanas palīgklase GLES 2.0, kura ļauj reālā laikā specificēt virsotnes un nodrošina noklusējuma ēnotāju neierobežotai tūlītējai renderēšana [21].

4.7.1.5. Vienkāršu formu renderēšana

Renderē punktus, līnijas, figūras – aizpildītas un kontūras [22].

Pēc noklusējuma 2D ortogrāfiska projekcija ar sākumpunktu kreisajā apakšējā stūrī, laukuma vienības tiek specificētas ekrāna pikseļos. To iespējams konfigurēt izmantojot konfigurāciju matricu. Konfigurāciju matricu nepieciešams atjaunināt, ja ekrāna izšķirtspēja mainās.

4.7.1.6 ETC1 atbalsts

LibGDX klase, kas paredzēta ETC1 kompresētu attēlu dekodēšanai. Nodrošina arī metodes PKM galvenes pievienošanai.

4.7.2. Augsta līmeņa 2D API saskarne

LibGDX ietver klases darbam ar 2D grafiku augstā līmenī. Klases sniedz abstrakcijas līmeni, kurš ļauj darboties ar izstrādi vairākām platformām paralēli, izstrādājot kodu vienu reizi visām sistēmām, kā arī padara darbošanos efektīvu un funkcionālu.

Šajā nodaļā paredzēts apskatīt lielāko daļu no šāda veida klasēm un rīkiem, kas ar tām saistīti.

4.7.2.1. Ortogrāfiska kamera

Kamera ar ortogrāfisku projekciju. Klases objektam iespējams pielietot sekojošas metodes:

- rotēšana ap virziena vektoru pa noteiktu leņķi;
- kameras objekta piesaistīšana ortogrāfiskajai projekcijai, izmantojot apskates punktu, kurš tiek ievietots ekrāna laukā, centrējot to vidū un izmantojot y-asi kā vērsuma virzienu;
- kameras objekta piesaistīšana ortogrāfiskajai projekcijai, centrējot to vidū un izmantojot y-asi kā vērsuma virzienu;
- kameras pārvietošana par noteiktu attālumu pa asīm metodes argumenti – pārvietojums pa x un y asi;
- kameras pārvietošana pa vektoru – metodes arguments vektors;
- projekcijas un kameras redzamības matricas atjaunināšana.

4.7.2.2. Augstas veiktspējas gariņu paketēšana un kešdarbe

Tiek veikta 2D attēlu zīmēšana, optimizējot tos ģeometrijai, kura nemainās. Gariņi un tekstūras tiek iekešotas un tām tiek piešķirts unikāls identifikators, kurš vēlāk var tik izmantots zīmēšanas operācijās. Izmērs, krāsa un tekstūru apgabals katram kešotajam attēlam nevar tikt mainīts. Šī informācija tiek glabāta video atmiņā un to nav nepieciešams sūtīt grafiskajai skaitļošanas iekārtai katru reizi, kad nepieciešams veikt attēlošanu uz ekrāna.

Gariņu un tekstūru kešošanas inicializācija notiek izmantojot *beginCache()* metodi, pēc kuras izsaukšanas notiek attēlu definēšana izsaucot atbilstošas pievienošanas metodes un visbeidzot process tiek noslēgts pielietojot *endCache()* metodi, kura atgriež kešoto datu identifikatoru [22].

Lai attēlot kēšotos datus, tiek pielietota klases *SpriteCache* metode *begin(),* tālāk tiek izsauktas zīmēšanas metodes, padodot tām kešoto datu identifikatorus un visbeidzot attēlošanas process tiek noslēgts izsaucot *end()* metodi [22].

Pēc noklusējuma klase SpriteCache izmanto ekrāna koordinātas lai noteikt vietu, kur veikt attēlošanu. Par sākuma punktu tiek izmantots ekrāna kreisais apakšējais stūris un skaitļošana notiek pa x un y asīm kā attiecīgi horizontālo un vertikālo virzienu. Iespējams veikt noklusējuma transformāciju un projekciju matricu konfigurācijas izmaiņas, kuras nepieciešams atjaunot pēc ekrāna izšķirtspējas izmaiņām [22].

Gariņu kešdarbe tiek pārvaldīta. Ja tiek zaudēts OpenGL konteksts un pēcāk tas tiek atjaunots, visi OpenGL resursi, kurus gariņu kešdarbe izmanto iekšēji tiek automātiski atjaunoti.

Gariņu kešdarbes instance ir lielu atmiņas daudzumu aizņemošs objekts un parasti katrai lietotnei piemīt viens gariņu kešdarbes objekts. Gariņu kešdarbe izmanto savu ēnotāju realizāciju attēlošanai. Gariņu kešdarbes objektu nepieciešams izvietot no atmiņas, kad tas vairs nav nepieciešams [22].

4.7.2.3. Tekstūru tīkli

Tekstūru tīkls jeb tekstūru atlass ir efektīvs veids, kā glabāt un izmantot tekstūru kopas lietotnes grafiskās reprezentācijas vajadzībām. *TextureAtlas* klases objekts nodrošina darbību ar tekstūru pakošanas rīkā izveidotajiem tekstūru tīkliem [23].

Dotās klases objektu nepieciešams izvietot no atmiņas, ja tas vairs nav nepieciešams, lai atbrīvot tā patērētos resursus.

4.7.2.4. Bitmap fonti

LibGDX atbalsta bitmap fontu izmantošanu lietotnēs. Fonts sastāv no diviem failiem – attēla faila vai tekstūru reģiona, kurš satur fonta simbolus kā arī no datnes *AngleCode BMFont,* kura teksta formātā apraksta kur atrodas katrs simbols tekstūru reģionā vai attēla datnē [24].

Teksts tiek attēlots izmantojot klasi *Batch*, kura tiek pielietota, lai attēlot 2D četrstūrus bāzējoties uz tekstūru reģionu. Teksts var tikt ievietots bitmap fontu kešdarbes klasē *BitmapFontCache* ar mērķi palielināt statiska teksta renderēšanas veiktspēju bez nepieciešamības katru reizi kalkulēt katra simbola atrašanās vietu [24].

Tekstūra, kas tiek ielādēta no attēla datnes tiek pārvaldīta, kā arī to nepieciešams izvietot no atmiņas, pielietojot *dispose()* metodi, kad tā vairs nav nepieciešama un nekur netiek izmantota [24].

4.7.2.5. TMX rūtiņu kartes atbalsts

LibGDX ietvars atbalsta iespēju izmantot Tiled rīkā radītās rūtiņu kartes, t.sk. darbu ar lauku atribūtiem un īpašībām [8].

4.7.2.6. Scene 2D un Scene 2D UI saskarnes bibliotēka

Scene 2D ir 2D skatu grafs, kas paredzēts veidot lietotnes un saskarnes izmantojot aktieru hierarhiju. Ir pieejama saskarnes komponentu bibliotēka *Scene 2D UI* [25].

Scene 2D nodrošina sekojošu funkcionalitāti [25]:

- rotācija un mērogs tiek pielietoti grupai un tās apakšelementiem, apakšelementi darbojas savā koordinātu sistēmā, vecāku transformācijas piemēro caurskatāmi;
- vienkāršota 2D zīmēšana, izmantojot gariņu pakotnes, katrs aktieris attēlo savā personīgajā koordinātu sistēmā, kur sākumpunkts ir aktiera kreisais apakšējais stūris;
- ievades un citu notikumu maršrutēšana līdz atbilstošajiem aktieriem, notikumu sistēma ir elastīga, vecākiem ir iespēja ieturēt notikumu līdz vai pēc tam kad to ir veikuši apakšelementi;
- metožu sistēma manipulācijām ar aktieriem laikā, metodes var tikt kombinētas vai slēgtas virknē ar mērķi panākt sarežģītākus efektus.

Scene 2D ir labi nokomplektēts izklājuma veidošanai, skatu attēlošanai, ievades un notikumu tveršanai spēles izvēlnē, HUD pārklājumos, rīkos un citos saskarnes elementos. *Scene 2D UI* pakotne satur daudzus aktierus jeb elementus un citus speciālus rīkus saskarnes veidošanai [26].

Scene 2D sastāv no trim galvenajām klasēm [25]:

1) *Actor* klase, kas ir grafa mezgls, kuram piemīt pozīcija, izmērs, oriģināls izmērs un mērogs, rotācija un krāsa.
2) *Group* klase, kas ir pēc būtības aktieris, kuram var būt apakšelementi.
3) *Stage* klase, kuras objektam piemīt kamera, gariņu pakotne, un saknes grupa, kura ietver gan aktieru attēlošanu gan notikumu tveršanu un izplatīšanu līdz aktieriem.

4.7.3. Augsta līmeņa 3D API saskarne

LibGDX ietver klases darbam ar 3D grafiku augstā līmenī. Klases sniedz abstrakcijas līmeni, kurš ļauj darboties ar izstrādi vairākām platformām paralēli, izstrādājot kodu vienu reizi visām sistēmām, kā arī padara darbošanos efektīvu un funkcionālu [8].

Šajā nodaļā paredzēts apskatīt lielāko daļu no šāda veida klasēm.

4.7.3.1. Perspektīva kamera

LibGDX klase darbam ar perspektīvu kameru. Klases konstruktors rada klases PerspectiveCamera objektu izmantojot trīs parametrus – redzamības leņķi grādos, redzamības lauka platumu un augstumu. Redzamības leņķis izteikts attiecībā pret redzamības lauka augstumu, redzamības leņķis platumā tiek izskaitļots izejot no redzamības lauka proporcijām [25].

4.7.3.2. Dekāļu paketēšana

Dekālis ir gariņa reprezentācija trīs dimensiju telpā. Tipiskas 3D transformācijas ir pieejamas līdzīgi ka citiem objektiem 3D telpā, piemēram, translācija, rotācija, mērogošana. Pozicionēšana iekļauj z ass komponenti. Korekts izklājums tiek garantēts izmantojot dziļuma buferi [26].

4.7.3.3. 3D renderēšana ar materiālu un apgaismojuma sistēmas atbalstu FBX modeļu ielādei

LibGDX sniedz iespēju veikt 3D renderēšanu ar materiālu un apgaismojuma sistēmas atbalstu un FBX formāta 3D modeļu importēšanas un izmantošanas iespējām [8].

4.8. Integrēti ietvarā pakalpojumi

LibGDX ietvars sniedz funkcionalitāti darbam ar datu kolekcijām, dažādas palīgfunkcijas, kā arī metodes JSON un XML datņu emitēšanai un importam [8].

4.9. Rīki

Ar LibGDX ietvaru ir savietojami ārēji rīki dažādu apakšuzdevumu spēles izstrādes procesa ietvaros risināšanai [8]:

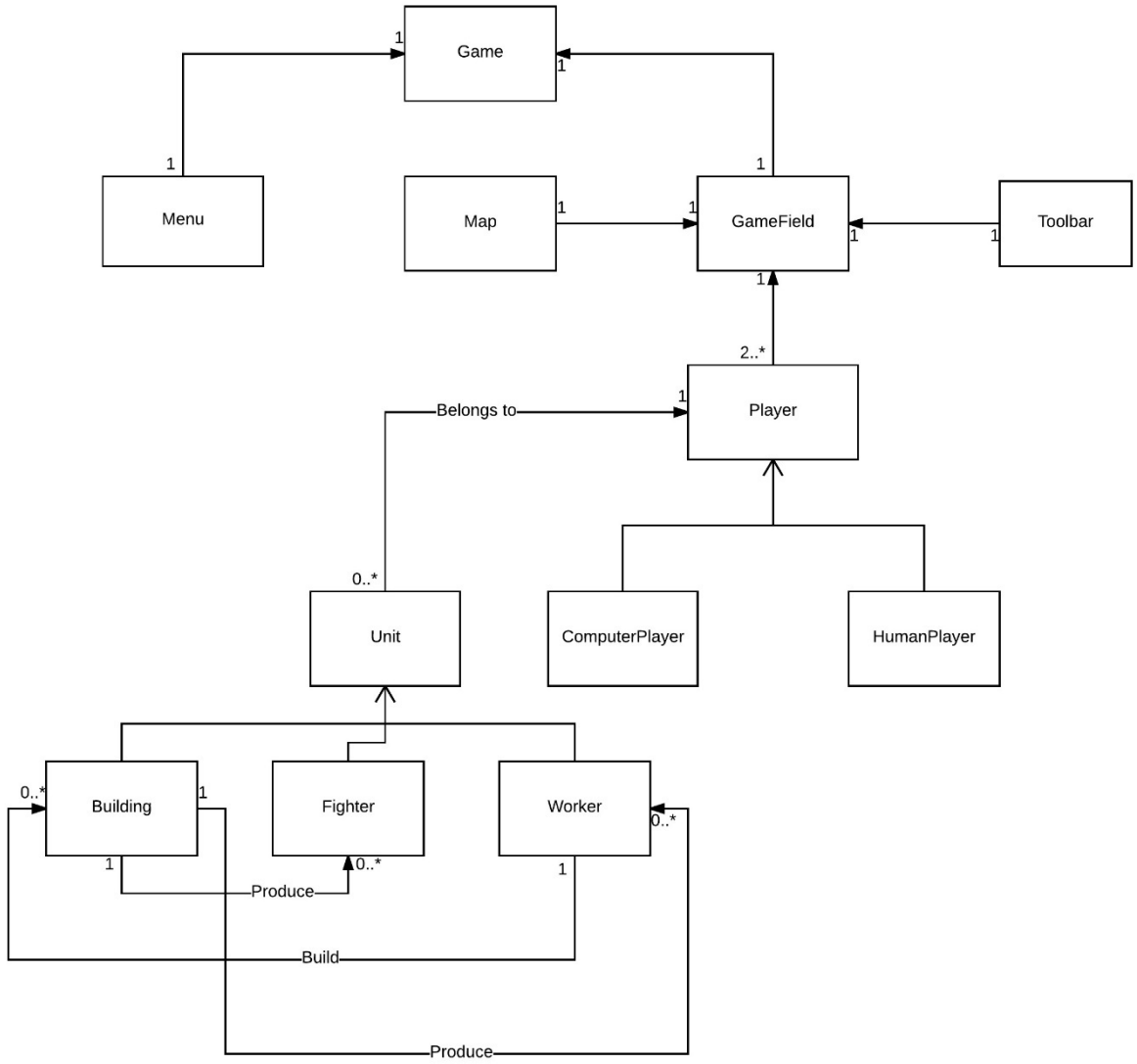
- 2D un 3D partikulu redaktori, kuri ļauj veidot daļiņu efektus animācijai 2D vai 3D telpā, kurus vēlāk var izmantot LibGDX lietotnē; jāņem vērā, ka 3D daļiņu redaktorā radītie efekti nav pielietojami darbā ar 2D klasēm;
- tekstūru pakotājs, kurš ļauj efektīvi veidot tekstūru atlasus izmantošanai LibGDX lietotnē, kas ir ievērojami efektīvāk nekā lietot atsevišķus tekstūru elementus, jo tekstūras piesaistīšana ir resursietilpīga operācija;
- bitmap fontu ģenerators, kurš pārveido atbalstāmajā lietotnē formātā nepieciešamos fontus.

5. SPĒLES PROJEKTĒJUMS

Šīs nodaļas ietvaros paredzēts apskatīt izstrādājamās spēles projektējuma un agrīnās prototipēšanas izstrādes etapa rezultātus.

5.1. UML konceptuālā klašu diagramma

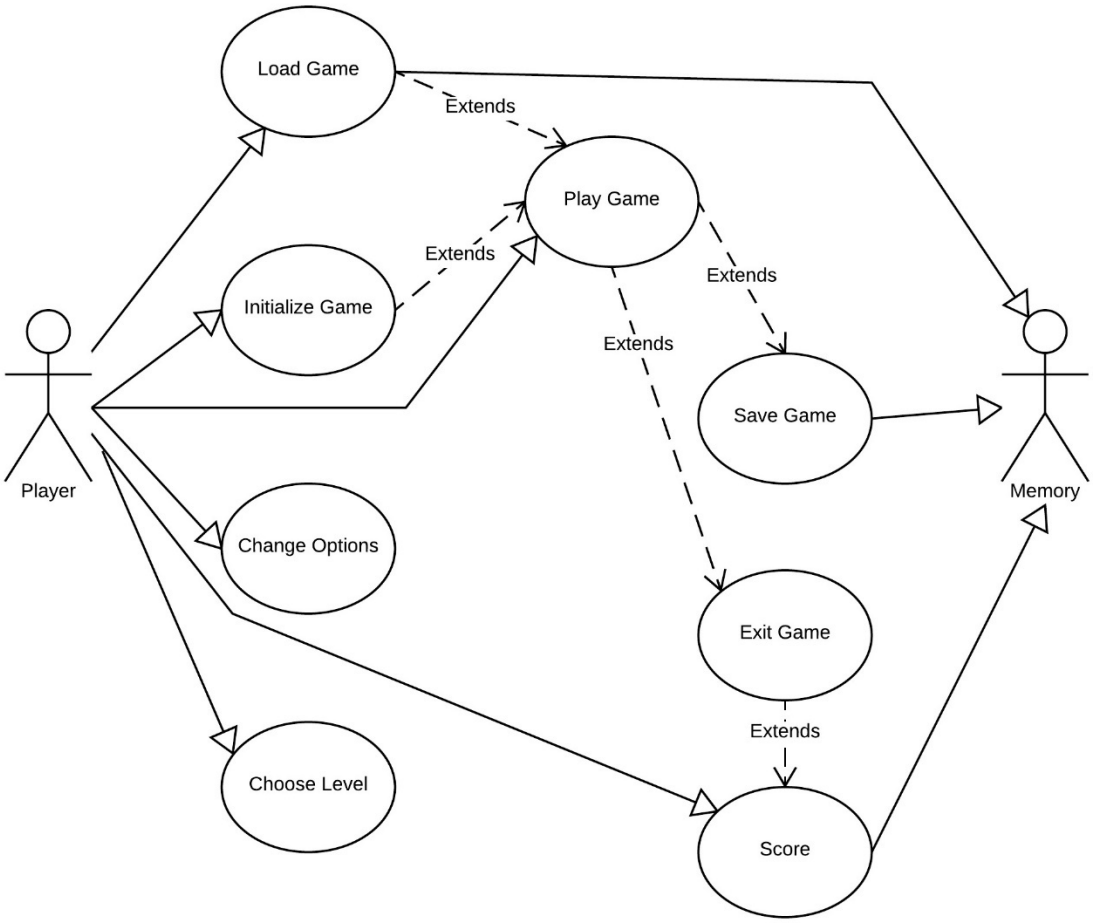
Attēlā 5.1. redzama spēles konceptuālā klašu diagramma.



5.1. att. **Spēles konceptuālā klašu diagramma**

5.2. Lietojumgadījumu diagramma

Attēlā 5.2. redzama spēles lietojumgadījumu diagramma.



5.2. att. **Spēles lietojumgadījumu diagramma**

Kā redzams no lietojumgadījumu diagrammas, lietotājam pēc lietotnes palaišanas pieejamas iespējas izvēlēties līmeni, no jau atklātajiem spēles līmeņiem, ielādēt saglabātu spēli, sākt jaunu spēli vai arī apskatīt sasniegtos rezultātus.

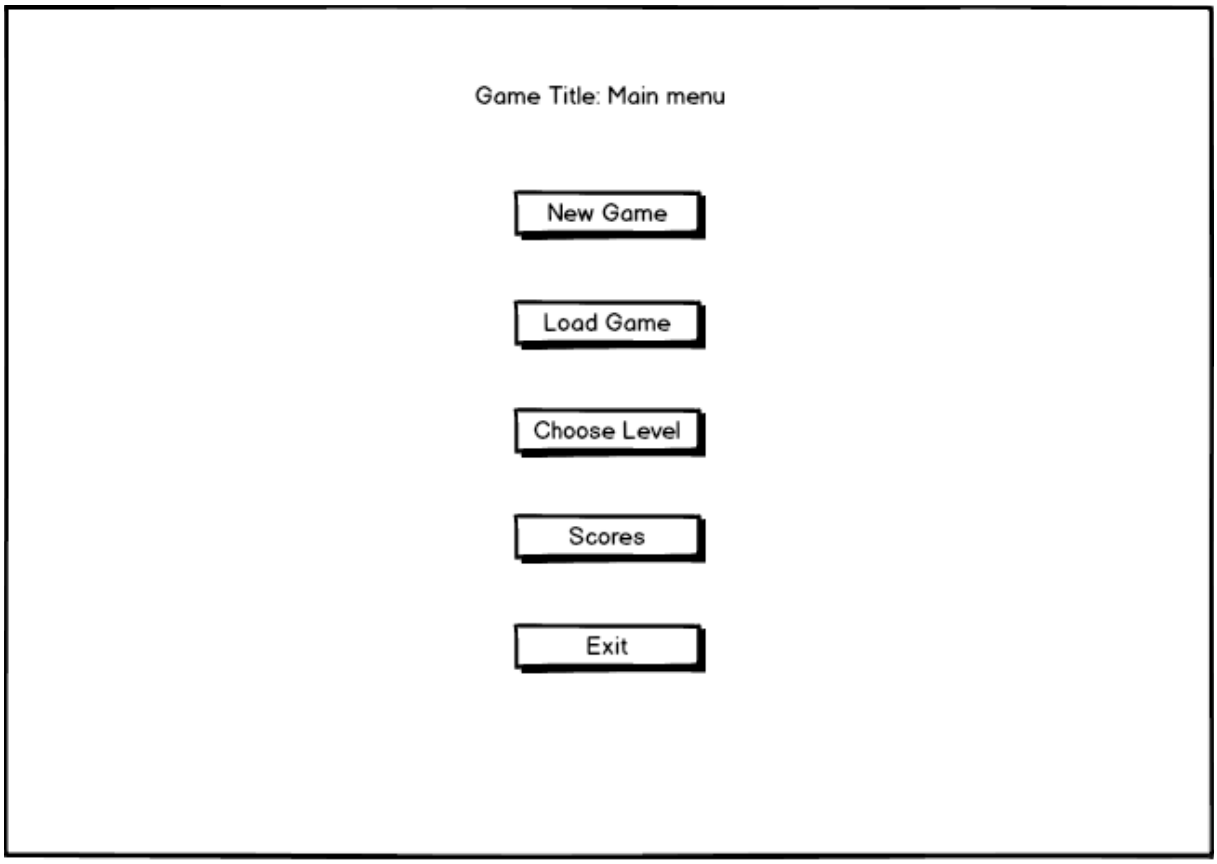
Noslēpts šajā abstrakcijas līmenī ir pats spēles process, taču redzams, kā spēles lietotne mijiedarbosies ar pieejamo ilglaicīgo atmiņu, lai glabāt tajā spēlētāja sasniegumus.

5.3. Prototipēšana

Dotajā nodaļā tiek apskatīti spēles skatu prototipi, kas ļauj izprast un iztēloties vizuāli, kā izskatīsies projektējamā spēle.

5.3.1. Izvēlnes saskarne

Attēlā 5.3. redzama spēles galvenās izvēlnes skats.

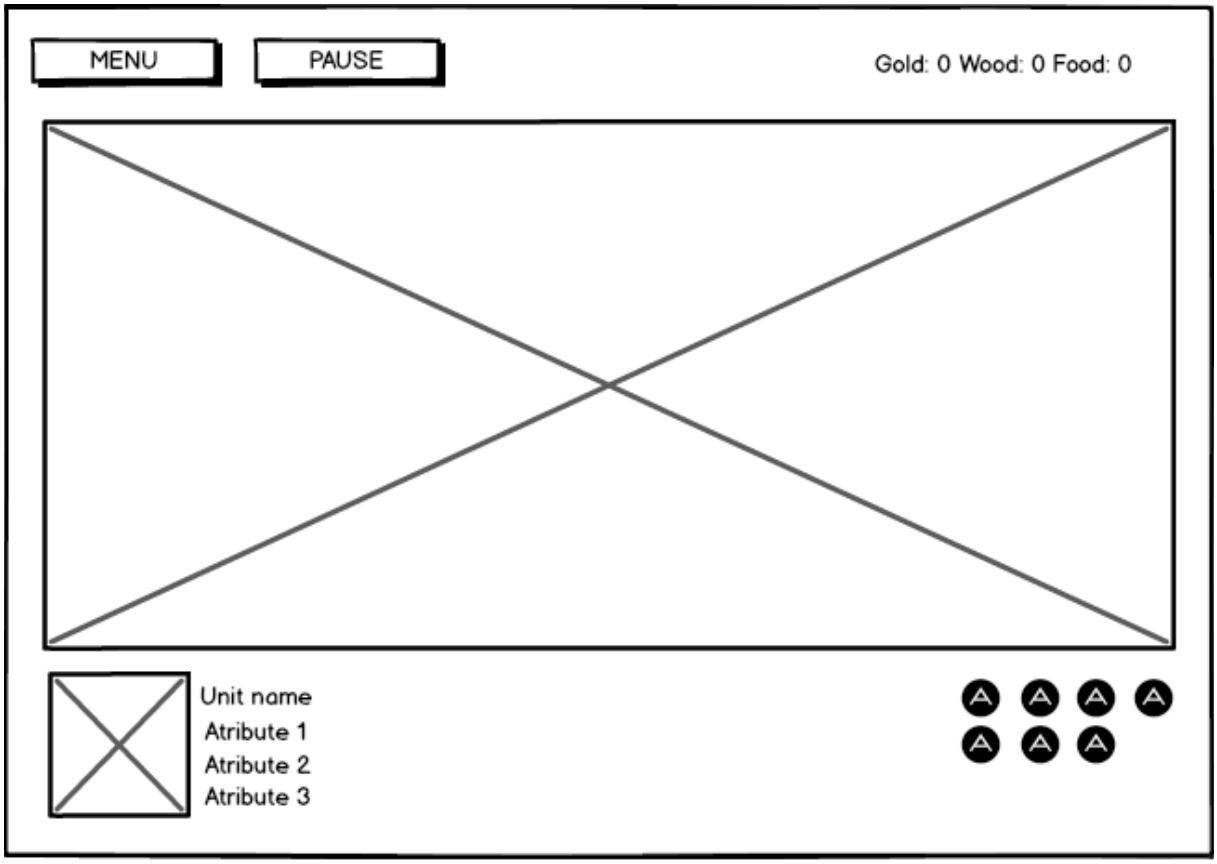


5.3. att. **Spēles galvenās izvēlnes skats**

Redzamais skats ir pirmais skats, ko redz spēlētājs pēc lietotnes palaišanas. No šī skata ir pieejamas visas galvenās darbības, tādas kā spēles uzsākšana, ielādēšana, rezultātu apskatīšana un līmeņa izvēle.

5.3.2. Spēles skata saskarne

Attēlā 5.4. redzama spēles galvenais skats – lietotāja saskarne aktīvās spēles laikā.



5.4. att. **Spēles skats spēles režīmā**

Attēlā 5.4. redzamajā skatā ir attēlots spēles laukums, pieejamā rīkjosla, kas ļauj atvērt lietotāja izvēlni, kā arī spēlētāja rīcībā esošo resursu uzskaitījums. Centrālo daļu aizņem galvenā daļa – spēles laukums, bet apakšējo ekrāna daļu aizņem pieejamo darbību klāsts un informācijas laiks par izvēlēto spēles vienību.

SECINĀJUMI

Darba izstrādes gaitā autors ieguva un noformēja zināšanas par spēles izstrādes procesu visā tā pilnībā. Šīs zināšanas tiek atspoguļotas darba pirmajā nodaļā. Uz šo zināšanu pamata var izdarīt secinājumus, ka spēles izstrādes process līdzīgi kā citu veidu programmatūras izstrādes procesi, sastāv no projektēšanas, izstrādes un uzturēšanas daļām, taču atšķirībā no citu veidu programmatūras izstrādes procesiem, spēles izstrādes process nevar eksistēt bez ļoti dažādām pēc būtības darbības jomām, kā audio māksla, vizuālā māksla un grafikas apstrāde, scenārija un dialogu veidošanas māksla, programmēšana, projektēšana. Spēles izstrādes process ir sarežģīts ar nepieciešamību savienot vienā projektā dažādu veidu speciālistus. Tas ir sarežģīti gan no izstrādes, gan projektēšanas viedokļa.

Darbā tika arī noformulēti spēles izstrādes procesa projektēšanas un plānošanas etapa rezultāti. Tie redzami darba otrajā un piektajā nodaļā un tie ir vērtīgi kā tiešs piemērs aprakstītajam pirmajā nodaļā no teorijas un projektēšanas viedokļa spēles izstrādes procesam.

Pēc darba trešajā nodaļā veiktā alternatīvu tehnoloģisko risinājuma apskata un salīdzinājuma, tiek secināts, ka tirgū eksistē liels daudzums līdzīgu pēc nolūka un iespējām rīku vairākplatformu lietotņu un spēļu radīšanā. Rīki pamatā ir līdzvērtīgi un izvēle par labu kādam no tiem ir lielā mērā gaumes jautājums un mazākā mērā objektīva kāda rīka pārākuma jautājums. No funkcionālā viedokļa tie pamatā ir līdzvērtīgi, kā arī sniedz līdzvērtīgas iespējas efektīvā vairākplatformu izstrādē.

Darba ceturtajā nodaļā ir detalizēta libGDX ietvara sniegto iespēju spēles izstrādē apskate, kā arī to iespēju, kurus sniedz libGDX vairākplatformu izstrādes problēmu risināšanā apskate, kuras rezultātā autors var izdarīt secinājumu, ka dotais ietvars pilnībā atrisina vairākplatofrmu izstrādes problēmas no koda izstrādes viedokļa, taču paliek aktuāli veiktspējas jautājumi kā arī lietotāja saskarnes jautājumi, jo, kā zināms, darbstaciju veiktspēja parasti ir ievērojami lielāka par mobilo iekārtu veiktspēju, kā arī ir acīmredzama liela atšķirība starp lietotāja saskarni, kas balstās uz klasiskajiem ievades mehānismiem un raksturīgajiem mobilajām iekārtām – skārienjūtīgajiem un žestu ievades mehānismiem. Dotie jautājumi nevar tikt atrisināti ietvara līmenī, tie jārisina lietotnes projektēšanas līmenī.

Maģistra darba ietvaros veiktais pētījums var tikt attīstīts tālāk, veicot veiktspējas atšķirības noteikšanu un ietekmi uz izstrādi, kā arī veicot pētījumus un piemeklējot labākos risinājumus saskarnes projektēšanā, kas sniegs vienlīdzīgu lietotāja pieredzi dažādu platformu lietotājiem.

IZMANTOTĀ LITERATŪRA

1. Video game development [tiešsaiste] – [atsauce: 25.01.2017.]. Pieejams: https://en.wikipedia.org/wiki/Video_game_development
2. Разработка компьютерных игр [tiešsaiste] – [atsuce: 25.01.2017.] Pieejams: https://ru.wikipedia.org/wiki/Разработка_компьютерных_игр
3. Дизайн-документ [tiešsaiste] – [atsauce: 25.01.2017.]. Pieejams: https://ru.wikipedia.org/wiki/Дизайн-документ
4. Game programming [tiešsaiste] – [atsuace: 25.01.2017.]. Pieejams: https://en.wikipedia.org/wiki/Game_programming
5. Level design [tiešsaiste] – [atsauce: 25.01.2017.]. Pieejams: https://en.wikipedia.org/wiki/Level_design
6. Warcraft III World Editor [tiešsaiste] – [atsauce: 25.01.2017.]. Pieejams: https://ru.wikipedia.org/wiki/Warcraft_III_World_Editor
7. Game testing [tiešsaiste] – [atsauce: 20.02.2017.]. Pieejams: https://en.wikipedia.org/wiki/Game_testing
8. libGDX [tiešsaiste] – [atsauce: 14.03.2017.]. Pieejams: https://libgdx.badlogicgames.com/features.html
9. IntelliJ IDEA editions comparision matrix [tiešsaiste] – [atsauce: 14.03.2017.]. Pieejams: https://www.jetbrains.com/idea/features/editions_comparison_matrix.html
10. Godot Engine features [tiešsaiste] – [atsauce: 15.03.2017.]. Pieejams: https://godotengine.org/features
11. Godot Engine documentation [tiešsaiste] – [atsauce: 15.03.2017.]. Pieejams: http://docs.godotengine.org/en/stable/
12. Polycode features [tiešsaiste] – [atsauce: 15.03.2017.]. Pieejams: http://polycode.org/features/
13. Polycode docs [tiešsaiste] – [atsauce: 15.03.2017.]. Pieejams: http://polycode.org/learn/
14. LOVE [tiešsaiste] – [atsauce: 15.03.2017.]. Pieejams: https://love2d.org
15. LOVE wiki [tiešsaiste] – [atsauce 15.03.2017.]. Pieejams: https://love2d.org/wiki/Main_Page

16. LibGDX File handling [tiešsaiste] – [atsauce:20.05.2017.]. Pieejams: https://github.com/libgdx/libgdx/wiki/File-handling
17. LibGDX Vertex Array [tiešsaiste] – [atsauce: 20.05.2017.]. – Pieejams: http://libgdx.badlogicgames.com/nightlies/docs/api/com/badlogic/gdx/graphics/glutils/VertexArray.html
18. LibGDX Mesh [tiešsaiste] – [atsauce: 20.05.2017.]. – Pieejams: http://libgdx.badlogicgames.com/nightlies/docs/api/com/badlogic/gdx/graphics/Mesh.html
19. LibGDX Texture [tiešsaiste] – [atsauce: 20.05.2017.]. – Pieejams: http://libgdx.badlogicgames.com/nightlies/docs/api/com/badlogic/gdx/graphics/Texture.html
20. LibGDX Shader Program [tiešsaiste] – [atsauce: 20.05.2017.]. – Pieejams: http://libgdx.badlogicgames.com/nightlies/docs/api/com/badlogic/gdx/graphics/glutils/ShaderProgram.html
21. LibGDX Immediate Model Renderer [tiešsaiste] – [atsauce: 20.05.2017.]. – Pieejams: http://libgdx.badlogicgames.com/nightlies/docs/api/com/badlogic/gdx/graphics/glutils/ImmediateModeRenderer.html
22. LibGDX Shape Renderer [tiešsaiste] – [atsauce: 20.05.2017.]. – Pieejams: http://libgdx.badlogicgames.com/nightlies/docs/api/com/badlogic/gdx/graphics/glutils/ShapeRenderer.html
23. LibGDX Texture Atlas [tiešsaiste] – [atsauce: 21.05.2017.]. – Pieejams: http://libgdx.badlogicgames.com/nightlies/docs/api/com/badlogic/gdx/graphics/g2d/TextureAtlas.html
24. LibGDX Bitmap font [tiešsaiste] – [atsauce: 21.05.2017.]. – Pieejams: http://libgdx.badlogicgames.com/nightlies/docs/api/com/badlogic/gdx/graphics/g2d/BitmapFont.html
25. LibGDX Perspective camera [tiešsaiste] – [atsauce: 21.05.2017.]. – Pieejams: http://libgdx.badlogicgames.com/nightlies/docs/api/com/badlogic/gdx/graphics/PerspectiveCamera.html
26. LibGDX Decal Batch [tiešsaiste] – [atsauce: 21.05.2017.]. – Pieejams: http://libgdx.badlogicgames.com/nightlies/docs/api/com/badlogic/gdx/graphics/g3d/decals/DecalBatch.html

DOKUMENTĀRĀ LAPA

Maģistra darbs "VAIRĀKPLATFORMU SPĒLES IZSTRĀDES PROCESS, IZMANTOJOT LIBGDX IETVARU, UN TĀ SALĪDZINĀJUMS AR LĪDZĪGIEM RĪKIEM" izstrādāts LU Datorikas fakultātē.

Darba teksta galīgā versija izgatavota 22.05.2017.

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

Autors: _____

(Autora paraksts un datums)

Ar savu parakstu apliecinu, ka esmu lasījis augstāk minēto maģistra darbu un atzīstu to par **p i e m ē r o t u / n e p i e m ē r o t u** (nevajadzīgo svītrot) aizstāvēšanai Latvijas Universitātes datorzinātņu maģistrantūrā.

Darba vadītājs: _____

(Vadītāja paraksts un datums)

Darbs iesniegts **maģistratūras sekretariātā** _____.

(Iesniegšanas datums)

Ar šo es apliecinu, ka darba elektroniskā versija ir augšupielādēta LU informatīvajā sistēmā.

Studiju metodiķe: _____.

(Metodiķes paraksts)

Recenzents: Profesors, Dr.dat. Uldis Straujums

(Akad.amats, zin.grāds, vārds, uzvārds)

Darbs aizstāvēts maģistra gala pārbaudījuma komisijas sēdē

_____ prot. Nr. _____

(Darba aizstāvēšanas datums)