Liepājas Valsts tehnikums

**“SystemShock” videospēles**

Kvalifikācijas eksāmena praktiskās daļas dokumentācija

Personālā kvalifikācija: Programmēšanas tehniķis

Darba autors: Kārlis Melveris

Liepāja 2024

# Saturs

[Saturs 2](#_Toc162075722)

[Ievads 5](#_Toc162075723)

[1. Uzdevuma formulējums 6](#_Toc162075724)

[2. Programmatūras prasību specifikācija 7](#_Toc162075725)

[2.1. Produkta perspektīva 7](#_Toc162075726)

[2.2. Sistēmas funkcionālās prasības 8](#_Toc162075727)

[2.2.1. Izvēlēties līmeni poga 8](#_Toc162075728)

[2.2.2. Iestatījuma izvēles atvēršanas poga 8](#_Toc162075729)

[2.2.3. Iziet poga 8](#_Toc162075730)

[2.2.4. Izvēlēties grūtības iestatījumu logs 9](#_Toc162075731)

[2.2.5. Izvēlēties grūtības iestatījumu 9](#_Toc162075732)

[2.2.6. Sākt spēli poga. 9](#_Toc162075733)

[2.2.7. Pauzēšana 10](#_Toc162075734)

[2.2.8. Atsākt spēli poga 10](#_Toc162075735)

[2.2.9. Iziet no līmeņa 10](#_Toc162075736)

[2.2.10. Lasīt līmeņa datus priekš spēles sākšanas 11](#_Toc162075737)

[2.2.11. Pārskatīt visu līmeņu datus 11](#_Toc162075738)

[2.2.12. Iel ekrāns 11](#_Toc162075739)

[2.2.13. Spēles laikā nospiesta kāda no pozīcijas pogām 12](#_Toc162075740)

[2.2.14. Spēles noteikto būltu ielādēšana 13](#_Toc162075741)

[2.2.15. Īsā kliška būltiņa 13](#_Toc162075742)

[2.2.16. garu spiedienu būltiņa 14](#_Toc162075743)

[2.3. Sistēmas nefunkcionālās prasības 14](#_Toc162075744)

[2.3.1. Veikt spējas prasības 14](#_Toc162075745)

[2.3.2. Vizuāls izskats 14](#_Toc162075746)

[2.3.3. Skaņu dizains 14](#_Toc162075747)

[2.3.4. Modificējama 15](#_Toc162075748)

[2.4. Gala lietotāja raksturiezīmes 15](#_Toc162075749)

[3. Izstrādes līdzekļu, rīku apraksts un izvēles pamatojums 16](#_Toc162075750)

[3.1. Izvēlēto risinājuma līdzekļu un valodu apraksts 16](#_Toc162075751)

[3.1.1 Izstrādes līdzekļi 16](#_Toc162075752)

[3.1.2 Dizaina rīki 16](#_Toc162075753)

[3.1.3 Datu glabāšanas formatēšanā 17](#_Toc162075754)

[3.2. Iespējamo risinājuma līdzekļu un valodu apraksts 17](#_Toc162075755)

[3.2.1 Izstrādes līdzekļi 17](#_Toc162075756)

[3.2.2 Dizaina rīki 17](#_Toc162075757)

[3.2.3 Datu glabāšanas formatēšanā 17](#_Toc162075758)

[3.2.4 Datu glabāšanas modifikācija 18](#_Toc162075759)

[4. Sistēmas modelēšana un projektēšana 19](#_Toc162075760)

[4.1. Klašu diagramma 19](#_Toc162075761)

[1. attēls. Klašu diagramma 19](#_Toc162075762)

[4.2. ER diagramma 20](#_Toc162075763)

[4.4. Funkcionālais un dinamiskais sistēmas modelis 21](#_Toc162075764)

[4.5. Aktivitāšu diagramma 21](#_Toc162075765)

[4.6. Lietojumgadījumu diagramma 21](#_Toc162075766)

[4.7. Sistēmas moduļu apraksts un algoritmu shēmas 21](#_Toc162075767)

[5. Lietotāju ceļvedis 22](#_Toc162075768)

[6. Testēšanas dokumentācija 23](#_Toc162075769)

[6.1. Izvēlētās testēšanas metodes, rīku apraksts un pamatojums 23](#_Toc162075770)

[6.2. Testpiemēru kopa 23](#_Toc162075771)

[6.3. Testēšanas žurnāls 23](#_Toc162075772)

[7. Individuālais ieguldījums 24](#_Toc162075773)

[8. Secinājumi 25](#_Toc162075774)

[9. Lietoto terminu un saīsinājumu skaidrojumi 26](#_Toc162075775)

[10. Literatūras un informācijas avotu saraksts 27](#_Toc162075776)

# Ievads

Šis ievads ir domāts, lai iedotu vienkāršu aprakstu par “SystemShock” dokumentāciju, lai gan lielākā daļa informācijas par videospēli būs aprakstīta uzdevuma formulējuma sadaļā.

Specifikācijas nodaļa ir domāta, lai izveidotu labāku saprašanu par šo projektu, spēles funkcionalitāti un spēles vizuālo dizainu. Lai to dokumentētu tas tiks izskaidros funkcionālā, nefunkcionāla daļā un kopējas perspektīvs par šo projektu.

Izstrādes līdzekļu nodaļā tiks izskaidrots kādus instrumentus un rīkus es lietošu, lai izveidotu šo videospēli, kā piemēram kuru video spēļu dzini, 3d modelēšanas rīkus, 2d modelēšanas un jebkurus papildinājumus kurus es lietoju klāt pie tiem.

Sistēmas modelēšanā un projektēšanā tiks izskaidrots manējās spēles datu bāzes uzbūvi un struktūru kura būs demonstrēta izmantojot ER diagrammu. Lai demonstrētu videospēles funkcijas un kontroles tiks izmantots funkcionālais un dinamiskais sistēmas modelis. Aktivitāšu diagramma tiks izmantota, lai parādītu funkcionāli solim pa solim kā videospēle tiks organizēta. Lietojum gadījumu diagramma (Use Case) tiks izmantota, lai reprezentētu kā spēlētājs un modifikators mainīs ar videospēli. Sistēmas moduļu apraksts un algoritmu shēmas sadaļā tiks izskaidrots dziļāk kā strādā funkcijas kuras tika pieminētas iepriekšējās diagrammās.

Lietotāja ceļvedis, tiek rakstīts, lai dotu instrukcijas kā pārvietoties cauri šīs videospēles iestatījumiem un līmeņiem. Kā arī kā modificēt to priekš tekstūras, mūzikas un jaunu līmeņu ievienošanai.

Projekta veidošanas laikā tiek dokumentēti visas kļūdas un problēmas kas izveidojas projekta taisīšanas laikā, kā arī tiks dokumentēts kādi instrumenti kuri tiek izmantoti, lai atrastu, labotu un dokumentētu šīs kļūdas.

# 1. Uzdevuma formulējums

Šī projekta mērķis ir, lai izveidotu ritma videospēli cyberpunk stilā ar 3d neon krāsas vizuāliem, dinamiski mainās un atskaņo kopā ar mūziku, lai veidotu unikālu spēles sajūtu.

Galvenais mērķis ir izveidot ritmu spēli kura saprot kas padara citas ritmu spēles jautras un interesantas, lai gan vienlaicīgi, kas ir viņu vājākās daļas un uzlabot šobrīd pieņemto normu. Pat ja mērķis ir pārveidot šo žanru, šīs spēles mērķis nav to pārtaisīt uz kaut ko jaunu, un uzturēt ritmu spēles jautrību, interesi un iemīļoto sajūtu.

SystemShock programmatūras arhitektūrā prioritāte ir modularitāte, lai veicinātu trešās puses iesaistīšanos, izmantojot modifikācijas. Aizmugursistēmas dizaina vienkāršības mērķis ir nodrošināt optimālu veiktspēju, koncentrējoties uz preventīvu dizainu, lai izvairītos no iespējamām kļūdām. Kā piemēru, ja lietotājs nomaina līmeņus, izdzēš, vai modificē tās asētus programmai ir jāsaprot tas un jāturpina strādāt.

Izstrādes process sākas ar cyberpunk stilu un tēmu, kā mūzikas stilu konceptuālu izpēti un piemēru atrašanu. Tālāk seko prototipu izstrāde, lai pārbaudītu spēles galveno mehānismu un iekļautu lietotāju atsauksmes, tiks izveidota funkcionālā versija ar minimāliem vizuāliem, bet galveno spēļu ritmu. Vizuālo efektu versiju kurā ir vizuālie aspekti, bez vairākiem funkcionāliem aspektiem, lai pārbaudītu dizainu un stilu. Mākslinieciskā attīstības vienkāršums un skaņas dizains, ārpus mūzikas, ir jāizveido, lai darbojas vienlaikus, lai izveidotu cyberpunk gaisotni. Lai gan veidojot šādu gaisotni aizņem daudz eksperimentāciju. Pēdējās fāzēs tiek uzsvērta kļūdu labošana, optimizācija un stratēģiskā izlaišana ar pastāvīgu atbalstu, izmantojot atjauninājumus un kopienas iesaisti.

Spēles SystemShock tehnoloģiskā infrastruktūra ir pielāgota pašreizējiem veiktspējas standartiem, bet arī piemērota nākotnei un vecākām ierīcēm, lai nodrošinātu saderību ar jaunām un vecām platformām. Tas ietver apsvērumus par pielāgojamību mainīgajām un nozares standartu ievērošanu, nodrošinot, ka spēle joprojām ir pieejama un atbilstoša tehnoloģijai progresējot. Lietotāja vieglā modificējamība ir grūti pievienojama pie vairākām ierīcēm, kā piemēram Apple telefoniem un konsolēm, lai gan šī dokumentācija ir izveidota, lai demonstrētu datora versiju, tādēļ konsoles pieejamība netiks pieminēta tālāk.

Kopumā šī spēle ir līdzīga citām ritmu spēlēm, lai gan dēļ unikālā stila, jaunām kontrolēm, plāniem izveidot pieejamu visām ierīcēm un 3d vizuāliem. Šis projekta mērķis ir liels, bet dēļ šī unikāluma ir liela cerība ka ritmu spēles fani un lietotāji kuri nekad nav spēlējuši šāda stila spēli varēs to izbaudīt.2. Programmatūras prasību specifikācija

Šī specifikācija ir domāta, lai izskaidrotu gala mērķi un domu gājienu kā funkcionēs, un izskatīsies šī spēle. Šī nodaļa ir sadalīta 4 daļās. Produkta perspektīvs, kurā ir aprakstīta spēles funkcijas kopumā. Sistēmas funkcionālās prasības tiks izskaidrota spēles funkcijas un spēlētāja ievade. Nefunkcionālās prasībās tiks izskaidrota katra daļa no projekta, kas nav funkcija, kā piemēram izskats, programmēšanas valoda, utt.

## 2.1. Produkta perspektīva

No sākuma es izskaidrošu šī projekta spēli. Šī ir ritmu spēle, tādēļ seko ierasto ritmu spēles formātu, kā spēlētājs tavs uzdevums ir nospiest un turēt pogas ritmiski ar spēles mūziku, ar vairākiem līmeņiem un grūtības iestatījumiem. Šie līmeņi izmaina kura mūzika tiek spēlēta, un vizuāli kā izskatās spēle. Kamēr grūtības iestatījumi nomaina cik daudz pogas un cik ātri ir jāspiež ritmā ar mūziku.

Sāksim ar UI un ģenerālu izkārtojumu pirms sākas paša spēle. No spēles sākumlapas vajadzētu būt ļoti vienkāršām 3 pogām. Pogu, lai sāktu spēli, pogu, lai nomainītu iestatījumus un, lai izietu no programmas. Iestatījumu poga vienkārši atvērs līdzīgu apskatu ar iestatījuma pogām un slīdņiem. Lai gan “sākt spēli” poga rādīs visus iespējamos līmeņus, kuros varēs redzēt mūzikas nosaukumu un albūma foto vai mākslas darbu. Nospiežot uz šī līmeņa parādīs izvēli nomainīt grūtumu un spēju spēlēt šo līmeni.

Pēc līmeņa izvēles un spēles sākšanas atveras īss sagatavošanās periods kur var redzēt pašu spēli, bet vel nesākas mūzika un līmenis. Šī gaidīšana ir domāta, lai sagatavotu spēlētāju un lēnām iesākt līmeni. Šīs spēles visi līmeņi izskatās līdzīgi, ar 3 ceļiem centrā un 3d vizuāliem aspektiem apkārt. Spēles gaitā, ārpus no spēlētāja redzes lauka nokrīt instrukcionālas bultiņas kuras korespondē ar kurām pogām ir lietotājam jāspiež. Kad šīs instrukcionālas bultiņas nokrīt pie ekrāna apakšas, kur ir atzīmēta līnija. Kad šis nokrīt ir jābūt muzikāla instrumenta skaņa vai vienkārši jābūt ritmā ar mūziku. Tad lietotāja objektīvs ir nospiest, vai turēt šīs korespondējošās pogas. Ja pārāk daudzas bultiņas pāriet pāri atzīmētai līnijai, tad samazinās dzīvība spēle beidzas kad vai nu mūzika beidzas vai lietotāja dzīvība beidzas. Jebkurā laikā var pauzēt un mainīt iestatījumus vai iziet no līmeņa. Lai gan tavs progress līmenī netiks saglabāts.

## 2.2. Sistēmas funkcionālās prasības

### 2.2.1. Izvēlēties līmeni poga

Mērķis:

Funkcija atver jaunu izvēlni ar visiem līmeņiem.

Ievaddati:

1. Peles kursors tiek nospiests uz šīs pogas.
2. līmeņa nosaukumi
3. līmeņa grūtuma līmeņi

Nosacījumi:

1. Poga ir redzama tikai sākumlapā.

Apstrāde:

Pēc nospiešanas lietotājs tiek pārnests uz spēles līmeņa izvēlnes.

### 2.2.2. Iestatījuma izvēles atvēršanas poga

Mērķis:

Funkcija atver jaunu izvēlni ar visiem iestatījumiem.

#### Ievaddati:

1. Peles kursors tiek nospiests uz šīs pogas.

Nosacījumi:

1. Poga ir redzama tikai sākumlapā un pauzēšanas sadaļa.

Apstrāde:

Pēc nospiešanas lietotājs tiek pārnests uz Iestatījumu izvēlnes.

### 2.2.3. Iziet poga

#### Mērķis:

Funkcija lai aizvērtu aplikāciju.

#### Ievaddati:

1. Peles kursors tiek nospiests uz šīs pogas.

#### Nosacījumi:

1. Poga ir redzama tikai sākumlapā.

#### Apstrāde:

Pēc nospiešanas aplikācija tiek apstādināta.

### 2.2.4. Izvēlēties grūtības iestatījumu logs

#### Mērķis:

Pēc nospiešanas uz konkrēta līmeņa tiek parādīta grūtības izvēle, pop-up logā.

#### Ievaddati:

1. Peles kursors nospiež uz jebkuru līmeni.
2. Konkrētā līmeņa noteiktie dati JSON faila.

#### Nosacījumi:

1. Tiek parādīti tikai grūtības līmeņi kuri ir pieejami konkrētam līmenim.
2. Paradās tikai spēles izvēlnes sadaļā.

#### Apstrāde:

Pēc nospiešanas paradās visa svarīgā informācija par izvēlēto līmeni, kā arī izvēle grūtības līmeņiem.

### 2.2.5. Izvēlēties grūtības iestatījumu

#### Mērķis:

Pēc nospiešanas uz konkrēta līmeņa tiek parādīta grūtības izvēle, popup logā.

#### Ievaddati:

1. Peles kursors nospiež uz kādu no grūtības iestatījumiem.

#### Nosacījumi:

1. Nav iespējams noklikšķināt uz grūtības līmeni kas nefunkcionē.
2. Paradās tikai spēles grūtības iestatījuma sadaļā.

#### Apstrāde:

Pēc nospiešanas sākt spēli poga tiek atvērta un var nospiest. Kā arī ir redzams kurš līmenis tika izvēlēts.

### 2.2.6. Sākt spēli poga.

#### Mērķis:

Atvērt spēles nodaļu, ar visiem datiem kas ir vajadzīgi, lai spēlētu iepriekš noteikto līmeni un grūtību.

#### Ievaddati:

1. Peles kursors nospiež uz sākt spēli pogu kas atrodas grūtības iestatījumu logā.
2. Dati par kurš grūtības līmenis tiek izvēlēts.
3. Iestatījumi par spēles noteikumiem un grafiskajiem iestatījumiem.

#### Nosacījumi:

1. Nav iespējams nospiest, ja lietotājs nav izvēlējiem grūtības līmeni.
2. Ir redzama tikai grūtības izvēles logā.
3. Poga nosaka, vai visi dati ir pieejami un neļauj spēlei sākt ja ir kļūda ar līmeni.

#### Apstrāde:

Pēc nospiešanas lietotājs tiek pārnests uz spēles sadaļu.

### 2.2.7. Pauzēšana

#### Mērķis:

Atvēta spēles logu kurā ir iestatījuma poga, atsākt spēli poga un iziet no spēles poga.

#### Ievaddati:

1. Klaviatūras “esc” poga.

#### Nosacījumi:

1. Tikai paradās, ja lietotājs atrodas spēles sadaļā.

#### Apstrāde:

Pēc nospiešanas spēle tiek apstādināta tajā momentā, lai parādītu izvēlnes logu.

### 2.2.8. Atsākt spēli poga

#### Mērķis:

Pēc pauzēšanas tikt atpakaļ pie spēles progresa.

#### Ievaddati:

1. Peles kursors nospiež uz pogu.
2. Klaviatūras “esc” poga.

#### Nosacījumi:

1. Ir redzams tikai spēles sadaļā un tikai, ja ir atvērts pauzēšanas logs.
2. Tikai aizver logu ja tas ir jau atvērts.

#### Apstrāde:

Pēc atsākšanas tiek uzsākts 2 sekunžu taimers un spēle atsākas iepriekšējā pozīcijā kurā lietotājs pauzēja.

### 2.2.9. Iziet no līmeņa

#### Mērķis:

1. Pēc pauzēšanas pārvietot lietotāju uz spēles izvēlnes sadaļu.

#### Ievaddati:

1. Peles kursors nospiež uz iziet no līmeņa pogas.

Nosacījumi:

1. Ir redzams tikai spēles sadaļā un tikai, ja ir atvērts pauzēšanas logs.

Apstrāde:

1. Spēles progress un dati netiek saglabāti un lietotājs tiek pārvietots uz izvēlēties līmeņa sadaļu.

### 2.2.10. Lasīt līmeņa datus priekš spēles sākšanas

Mērķis:

1. Izlasīt līmeņa failu datus un saglabāt viegli lasāmā formātā, priekš lēmeņa

Ievaddati:

1. Izvēlētā limeņa faila nosaukums
2. līmeņa izvēlētais grūtums

Nosacījumi:

1. Līmeņa failiem ir jābūt noteiktā direktoriā
2. Izsauc tikai vienu reizi pēs spēles sākšanas

Apstrāde:

1. Dati tiek saglabāti lokāli, lai varētu būt viegli izlasāmi līmeņa vidū.
2. Pārbauda vai dati ir korekti un strādā
3. Mazas problēmas ar nepietiekamiem datiem ir jābūt piepildītām automātiski

### 2.2.11. Pārskatīt visu līmeņu datus

Mērķis:

Izlasīt visu līmeņa failus, lai var parādīt līmeņu izvēlē (2.2.1.)

Ievaddati:

1. līmeņa faila directorija

Nosacījumi:

1. Ir jābūt kaut vienam līmenim, iekšā direktorijai

Apstrāde:

1. Izlasa un izvada katra līmeņa nosaukums un grūtības līmeņus
2. Ja ir specifizēts parāda PNG failu un maina līmeņa apmales krāsu specifizētam līmenim.

### 2.2.12. Lādēšanās ekrāns

Mērķis:

Parādīt lietotājam kā programma procesē datus, lai lietotājs zina ka spēle nav apstājusies.

#### Ievaddati:

1. Apstiprinājums ka nākošā sadaļa ir gatava
2. Paziņojums ka jauna sadaļa tiek atvērta

#### Nosacījumi:

1. Radās tikai ja aizņem ilgāk par 0.5 sekundes, lai ielādētos.

#### Apstrāde:

1. Pēc katra procesa tiek izdots paziņojums ka progress ir pabeidzies, un tiek noņemts lādēšanās ekrāns

### 2.2.13. Spēles laikā nospiesta kāda no pozīcijas pogām

#### Mērķis:

Kad lietotājs nospiež kādu no četrām kustības pogām (w, a, d, vai space) spēlei ir jāreaģē, vai tas bija pareizā laikā, vai nē.

#### Ievaddati:

1. Spēles noteiktās bultas pozīcija
2. Spēles noteiktās bultas tipu
3. Lietotāja nospiestās pogas vērtību un nospiešanas ilgumu

#### Nosacījumi:

1. Tiaki tiek procesēta, ja lietotājs ir jebkurā līmenī
2. Tikai skaita punktus un dzīvību pēc sākuma 2 sekundes gaidīšanas periodu
3. Netiek procesēta pēc līmeņa pabeigšanas

#### Apstrāde:

1. Kad lietotājs nospiež jebkuru no 4 pogām jebkura no šīm iespējām var notikt:

**1.tabula**

**Funkcijas apstrāde**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Īsā kliška būlta gadijumā | garu spiedienu ūltas gadijumā | Neviena būlta laukā |
| Poga tiek nospiesta tieši laikā | Bonus punktus un dzīvību iedod | pusi iespējamiem punktiem | neskaita |
| Poga tiek Nospiesta mazliet par ātru/lēnu | Punkti iedoti, bet ne dzīvību | ceturtdaļu iespējamiem punktiem | neskaita |
| Poga tiek nospiesta pārāk vēlu/ātru | Nav punkti iedoti, dzīvība samazinās | Nav punkti iedoti, dzīvība samazinās | neskaita |
| Poga netiek nospiesta | Nav punkti iedoti, dzīvība samazinās | Nav punkti iedoti, dzīvība samazinās | neskaita |
| Nepareizā poga tika nosipesta | Nav punkti iedoti, dzīvība samazinās | Nav punkti iedoti, dzīvība samazinās | neskaita |
| Poga nospiesta laikā un atlaista perfekti | Bonus punktus un dzīvību iedod | Bonus punktus un dzīvību iedod | neskaita |
| Poga nospiesta laikā, bet atlaista par vēlu/ātru | Bonus punktus un dzīvību iedod | Punkti iedoti, bet ne dzīvību | neskaita |
| Poga tiek nospiesta par vēlu, bet atlaista perfekti | Punkti iedoti, bet ne dzīvību | Punkti iedoti, bet ne dzīvību | neskaita |

### 2.2.14. Spēles noteikto bultu ielādēšana

#### Mērķis:

Norādīt lietotājam kuras pogas, cik ilgi un kurā laikā ir jāspiež, lai progresētu līmenī.

#### Ievaddati:

1. Līmeņa dati
2. Līmeņa modifikācijas
3. Šobrīdējais laiks līmenī

#### Nosacījumi:

1. Būltas vienmēr ir ritmiski kopā ar mūziku

#### Apstrāde:

1. Kad līmenis ir sācies un ir līdz galam izlasīts, programma ievieto bultiņu spēles aukšā, tieši kādu pozīciju, tipa un kad līmeņa fails nosaka.

### 2.2.15. Īsā kliška būltiņa

#### Mērķis:

Ja “Spēles noteikto būltu ielādēšana” ielādē īsa kliška būltiņu, šis nosaka visu kas notiks ar to.

#### Ievaddati:

1. Būltas ātrums
2. Vai ir lietotājs nospiedis pogu laikā

#### Nosacījumi:

1. Būltas tiek izdzēstas ja tās ir ārpus lietotāja skatam.

#### Apstrāde:

1. Pēc ielādēšanās, būltas spraits kustās uz leju noteiktā ātrum
2. Ja būltas sasniedz atzīmēto vietu tās dati tiek pārsūtīti uz “Spēles laikā nospiesta kāda no pozīcijas pogām” funkciju

### 2.2.16. garu spiedienu būltiņa

#### Mērķis:

Ja “Spēles noteikto būltu ielādēšana” ielādē garu spiediena būltiņu, šis nosaka visu kas notiks ar to.

#### Ievaddati:

1. Būltas ātrums
2. Būltas garums
3. Vai ir lietotājs nospiedis pogu laikā

#### Nosacījumi:

1. Būltas tiek izdzēstas ja tās ir ārpus lietotāja skatam.

#### Apstrāde:

1. Pēc ielādēšanās, būltas spraits kustās uz leju noteiktā ātrumā
2. Būlatai ir jābūt noteiktā garumā.
3. Ja būltas sasniedz atzīmēto vietu tās dati tiek pārsūtīti uz “Spēles laikā nospiesta kāda no pozīcijas pogām” funkcij

## 2.3. Sistēmas nefunkcionālās prasības

### 2.3.1. Veikt spējas prasības

Spēlei ir jāstrādā ar jebkuru datoru ar Gtx 1050 grafikas kartes un i7-6800K CPU ekvivalentiem. Lai gan ar ir jābūt grafisku intensitātes un partikulu iestatījumiem, lai pārliecinātos ka vecāki, vai lētāki datori var spēlēt.

### 2.3.2. Vizuāls izskats

Visai spēlei ir jābūt krāsainai, lai uzturētu “Cyberpunk” vai “neon” mākslas stilu. Kā arī visām pogām un slaidajiem ir jābūt viegli identificējamiem. Ja pogu nevar nospiest, tam ir jābūt viegli identificējamam.

### 2.3.3. Skaņu dizains

Pēc katras pogas vai slīdņa kustības ir jābūt kaut kādai identificējami skaņai, tas arī ieskaita, spēles ritma gaitā.

### 2.3.4. Modificējama

Jābūt relatīvi vienkārši izveidot jaunu līmeni un pievienot to spēlē, pat bez Unity projekta pieejas. Kā arī vizuālie elementi, katrā līmenī ir jābūt viegli modificējamiem. Kā piemēru priekš katras dziesmas, tiek mainīts ceļu un bultiņas tekstūru.

## 2.4. Gala lietotāja raksturiezīmes

Veidojot ritmu spēli kura ir modificējama, veidos divu veidu lietotājus. Vienkāršus lietotājus, kuri instalēts trešās parijas veidotus līmeņus, dizaina elementus un mūziku. Kā arī modifikatori, kuri veidos trešās partijas līmeņus, dizaina elementus un mūziku, kurus varēs publicēt un pievienot pie Cybershock.

Lietotāji neizzinās neko saistītu ar iekšējo datu struktūru, vai kā šī spēle strādā. Tādēļ jāpārliecinās ka visa datu struktūra ir viegli atrodama, labojama un viegli saprotama. Kā piemēru ja lietotājs instalē trešās partijas līmeni, kurai ir savi dizaina elementi un mūzika, ir jābūt vieglam instalācijas procesam.

Modifikatori ir lietotāji, kuri veidos jaunus līmeņus un mainīs failus, lai pārliecinātos ka modifikatori var viegli veidot savus līmeņus. Tādēļ ir jābūt vai nu dokumentācijai, vai instrumenti, lai veidotu korespondējošos formātus. Kā arī ir jābūt log fails, vai veidu kā informēt lietotāju ja ir problēmas ar programmatūru.

# 3. Izstrādes līdzekļu, rīku apraksts un izvēles pamatojums

Šeit tiks aprakstīti visi rīki un instrumenti kuri tiks izmantoti, lai izveidotu programmatūru pēc dokumentācijas iepriekš noteiktam prasībām. Pirmajā nodaļā tiks noteiktas visas programmatūras un līdzekļus kuri tiks izmantoti izstrādes laikā. Otrā sekcija vairāk aprakstīs visus resursus un līdzekļus kuri netiks obligāti lietoti, lai gan var būt noderīgi programmatūras izstrādes laikā.

## 3.1. Izvēlēto risinājuma līdzekļu un valodu apraksts

Šī nodaļa saturēs visas programmatūras, programmēšanas valodas un līdzekļus kuri tiks lietoti izstrādes procesā. Šī nodaļa tiks sadalīta 3 daļās, līdzekļi, Dizaina rīki, datu glabāšana. Kur tālāk tiks izskaidrots katrs resurss, kā tiks izmantots un kādēļ tieši šo konkrēto līdzekli.

### 3.1.1 Izstrādes līdzekļi

* Unity 2019.4.f3.1

Galvenā sadaļa tiek taisīta Unity spēles dzini, kurš tiks izmantots, lai izveidotu spēles funkcionalitāti, failu glabāšanu, renderēšanu un citu programmatūru integrāciju. Lai atbalstītu iepriekšējo pieredzi tiks izmantota Unity versiju 2019.4.f3.1 ar kuru ir iepriekšējā pieredze. Var būt problēmas ar šo versiju, viss īpaši ar partikulu un efektu sadaļu. Tādēļ pāriešana uz citu versiju var notikt.

* Studio Code Editor

Lai izmainītu Unity programmatūru ir jāizmanto C# programmēšanas valodu, lai to valodu rediģētu es izmantošu rekomendēto IDE. Pie Studio Code Editor ir ispēja pievienot

### 3.1.2 Dizaina rīki

* Photoshop

Lai veidotu lielāko daļu 2D dizainus tiks izmantota jaunākā versija ar Photoshop. Photoshop ir noderīgs dēļ krāsas regulācijas iestatījumiem kas atļauj izveidot detalizētus emisijas un svara kartes.

* Blender

Blender programma tiks izmantota lai veidotu 3D objektus un UV kartes. Šī programma tika izvēlēta, dēļ tās vairākām iebūvētām funkcijām kas atļauj izmantot tikai vienu programmu veidojot visas daļas kas ir saistītas ar 3D modeļiem.

### 3.1.3 Datu glabāšanas formatēšanā

* Json

Json faila formāts ir ļoti viegli modificējams un pieejams caur lielāka daļa valodām tas izveido daudz vieglāku datu modifikāciju un veidošanu. Kā arī tas atļauj pievienot jaunus pievienojumus un modifikācijas nemainot kopējo datu struktūru.

## 3.2. Iespējamo risinājuma līdzekļu un valodu apraksts

Iespējamo risinājumu līdzekļu nodaļa ir domāta, lai rādītu alternatīvas risinājumus visām iepriekš pieminētām programmatūrām un risinājumiem. Lai gan šie risinājumi nav plānoti, tie ir noderīgi gadījumos kad ir kādas problēmas ar iepriekš minētām programmatūrām.

### 3.2.1 Izstrādes līdzekļi

* Unity 2022.3.6f1

Pat ja šī versija ir jaunāka un modernizētāka un ar ļoti līdzīgām un vairākām jaunām funkcijām. Tas iemesls kādēļ šī versija netiek izmantota ir dēļ tā ka šī versija nav ierasta un nav pieredze izmantot. Viena no lietām kas bija izmainīta ir GUI un ģenerālu dizainu, tas var ļoti ietekmēt projekta taisīšanas procesu.

* Visual studio

Visual studio ir parasti rekomendēta priekš Unity C# galvenokārt dēļ visām trešās partijas bibliotēkām un modifikācijām kuras ir domātas C# un konkrētu Unity. Pat ja lielākā daļa Unity programmētāji izmanto šo IDE tai ir problēmas ar optimizāciju, un parasti aizņem pārāk daudz RAM kas var ļoti ietekmēt testēšanu.

### 3.2.2 Dizaina rīki

* Inkscape

Šī programma ir domāta, lai veidotu vektora grafikus kuri ir daudz labāki priekš videospēles grafikiem, lai gan Inkscape ir krāsas imitācijas kas izveido problēmas ar modifikāciju, svara kartes veidošanu un emisijas kartes veidošanu.

### 3.2.3 Datu glabāšanas formatēšanā

* Īpašumtiesības formatēts fails

Alternatīvi JSON formātam ir iespēja veido īpašumtiesības formātu. Paštaisīts formāts var aizņemt mazāk atmiņu, ātrāk lasāms un viss svarīgāk var aizsargāt pret pirātismu vai trešās partijas modifikāciju. Vairākas kompānijas izmanto paštaisītu, vai īpašumtiesības formātu dēļ iepriekš minētiem iemesliem. Lai gan, priekš indi ražošanas šī drošība un minimāli uzlabojumi nav pietiekami lieli, lai veidotu pilnībā jaunu formatējumu. Jauns formatējums nozīmē ka jāveido jauns lasītājs, re-formatētājs un rakstītājs, visi var aizņemt pārāk daudz laiku šādam projektam.

### 3.2.4 Datu glabāšanas modifikācija

* Paštaisīta json kartes veidošanas aplikācija

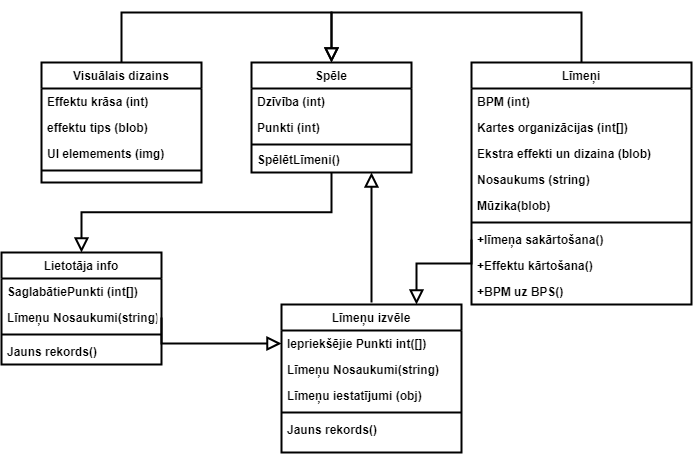
Lai veidotu līmeņus ātrāk ir iespēja veidot alternatīvu programmu kas izmantojot JSON formātu atļaus spēlēt korespondējošo mūziku un demonstrācijas skatu no videospēles skatupunkta. Šādi veidota programmatūra ir specifiska un tikai izmantojama ar cybershock JSON kārtojuma formātu.

# 4. Sistēmas modelēšana un projektēšana

## 4.1. Sistēmas struktūras modelis

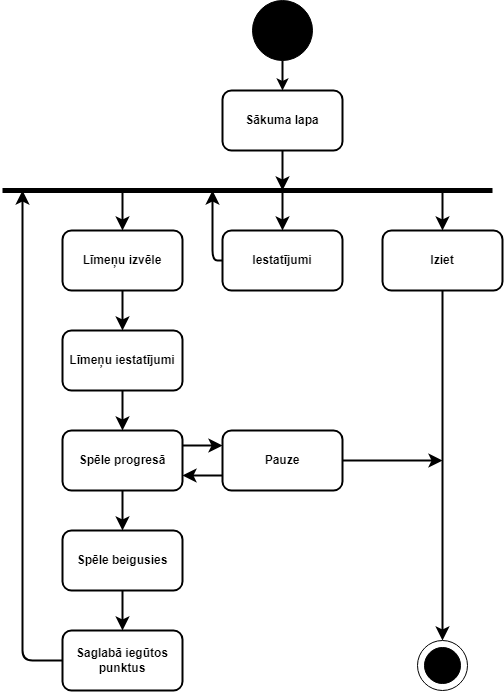
1. attēls Sistēmas struktūras modelis

## 4.2. Klašu diagramma

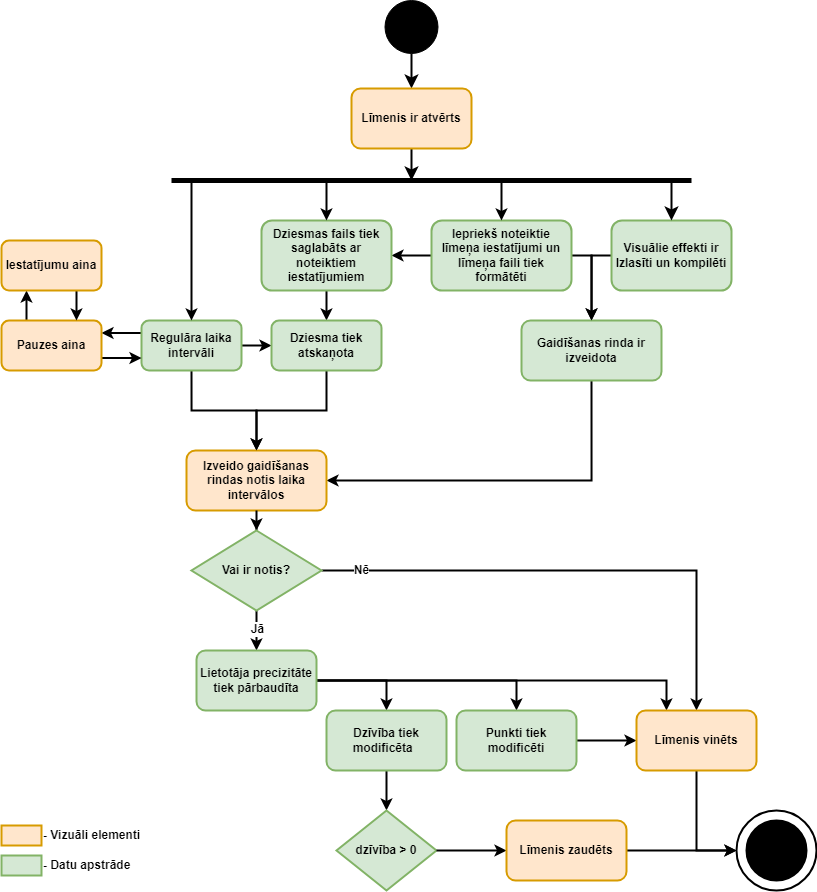
Šī diagramma reprezentē datus kuri tiks izmantoti projektā. Katra sadaļa ir individuālā failā vai faila direktorijā. 

2. attēls. Klašu diagramma

## 4.3. State diagramma

Šī diagramma reprezentē tikai navigācijas un loģistikas funkcijas. Šī diagramma nav veidota, lai demonstrētu internālas un datu pārraides funkcijas.

**3. attēls State diagramma**

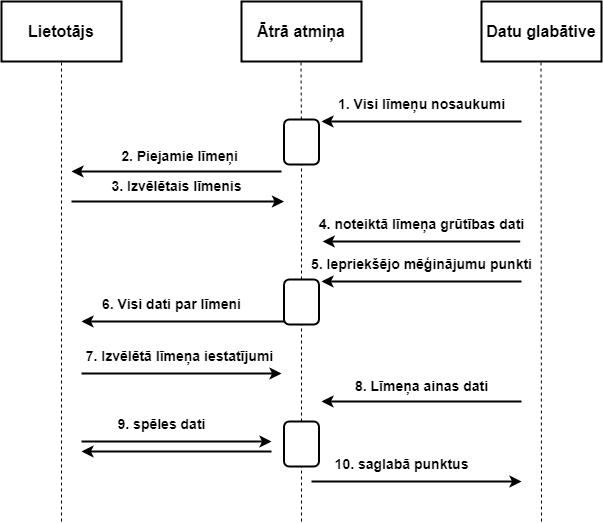
Šī diagramma ir domāta, lai reprezentētu “spēle progresā” nodaļu un labāk izskaidrotu kā lietotāji pārvietosies cauri spēles elementiem un kā tie mainīsies relatīvi padotajiem datiem un lietotāju reakciju.

**4. attēls State diagramma**

## A diagram of a flowchart Description automatically generated4.4. Algoritma diagramma

# 5. attēls Algoritma diagramma

## 4.5. Scenāriju diagramma

Lai reprezentētu datu plūsmas laikus un ieguves vietu šis dinamiskais sistēmas modelis ir domāts, lai atrastu iespējamus konfliktus, vai neloģiskas informācijas pārraides gadījumus.

# 6. attēls Scenāriju diagramma

# 5. Lietotāju ceļvedis

# 6. Testēšanas dokumentācija

## 6.1. Izvēlētās testēšanas metodes, rīku apraksts un pamatojums

## 6.2. Testpiemēru kopa

## 6.3. Testēšanas žurnāls

# 7. Individuālais ieguldījums

# 8. Secinājumi

# 9. Lietoto terminu un saīsinājumu skaidrojumi

# 10. Literatūras un informācijas avotu saraksts