Liepājas Valsts tehnikums

**“SystemShock” videospēles**

Kvalifikācijas eksāmena praktiskās daļas dokumentācija

Personālā kvalifikācija: Programmēšanas tehniķis

Darba autors: Kārlis Melveris

Liepāja 2024

# Saturs

[Saturs 2](#_Toc168066442)

[Ievads 1](#_Toc168066443)

[1. Uzdevuma formulējums 2](#_Toc168066444)

[2.1. Produkta perspektīva 3](#_Toc168066445)

[2.2. Sistēmas funkcionālās prasības 4](#_Toc168066446)

[2.2.1. Izvēlēties līmeni poga 4](#_Toc168066447)

[2.2.2. Iestatījuma izvēles atvēršanas poga 4](#_Toc168066448)

[2.2.3. Iziet poga 4](#_Toc168066449)

[2.2.4. Izvēlēties grūtības iestatījumu logs 5](#_Toc168066450)

[2.2.5. Izvēlēties grūtības iestatījumu 5](#_Toc168066451)

[2.2.6. Sākt spēli poga. 5](#_Toc168066452)

[2.2.7. Pauzēšana 6](#_Toc168066453)

[2.2.8. Atsākt spēli poga 6](#_Toc168066454)

[2.2.9. Iziet no līmeņa 6](#_Toc168066455)

[2.2.10. Lasīt līmeņa datus priekš spēles sākšanas 7](#_Toc168066456)

[2.2.11. Pārskatīt visu līmeņu datus 7](#_Toc168066457)

[2.2.12. Lādēšanās ekrāns 7](#_Toc168066458)

[2.2.13. Spēles laikā nospiesta kāda no pozīcijas pogām 8](#_Toc168066459)

[2.2.14. Spēles noteikto bultu ielādēšana 9](#_Toc168066460)

[2.2.15. Īsā klikšķa bultiņa 9](#_Toc168066461)

[2.2.16. Garu spiedienu bultiņa 10](#_Toc168066462)

[2.2.17. Fotogrāfijas un tekstūras apstrāde mainīšanas gadījumā. 10](#_Toc168066463)

[2.2.18. Sākumlapas kursora sekojos aizmugures lauks 10](#_Toc168066464)

[2.3. Sistēmas nefunkcionālās prasības 11](#_Toc168066465)

[2.3.1. Veikt spējas prasības 11](#_Toc168066466)

[2.3.2. Vizuāls izskats 11](#_Toc168066467)

[2.3.3. Skaņu dizains 11](#_Toc168066468)

[2.3.4. Modificējama 11](#_Toc168066469)

[2.4. Gala lietotāja raksturiezīmes 11](#_Toc168066470)

[3. Izstrādes līdzekļu, rīku apraksts un izvēles pamatojums 12](#_Toc168066471)

[3.1. Izvēlēto risinājuma līdzekļu un valodu apraksts 12](#_Toc168066472)

[3.1.1 Izstrādes līdzekļi 12](#_Toc168066473)

[3.1.2 Dizaina rīki 13](#_Toc168066474)

[3.1.3 Datu glabāšanas formatēšanā 13](#_Toc168066475)

[3.2. Iespējamo risinājuma līdzekļu un valodu apraksts 13](#_Toc168066476)

[3.2.1 Izstrādes līdzekļi 13](#_Toc168066477)

[3.2.2 Dizaina rīki 14](#_Toc168066478)

[3.2.3 Datu glabāšanas formatēšanā 14](#_Toc168066479)

[3.2.4 Datu glabāšanas modifikācija 14](#_Toc168066480)

[4. Sistēmas modelēšana un projektēšana 15](#_Toc168066481)

[4.1. Sistēmas struktūras modelis 15](#_Toc168066482)

[1. attēls Sistēmas struktūras modelis 15](#_Toc168066483)

[4.2. Klašu diagramma 16](#_Toc168066484)

[2. attēls. Klašu diagramma 16](#_Toc168066485)

[4.3. State diagramma 17](#_Toc168066486)

[4.4. Algoritma diagramma 19](#_Toc168066487)

[5. attēls Algoritma diagramma 19](#_Toc168066488)

[4.5. Scenāriju diagramma 20](#_Toc168066489)

[6. attēls Scenāriju diagramma 20](#_Toc168066490)

[5. Lietotāju ceļvedis 21](#_Toc168066491)

[5.1 Sākumlapa 21](#_Toc168066492)

[5.2 Iestatījumi 22](#_Toc168066493)

[5.3 līmeņu izvēle 24](#_Toc168066494)

[6. Testēšanas dokumentācija 25](#_Toc168066495)

[6.1. Izvēlētās testēšanas metodes, rīku apraksts un pamatojums 25](#_Toc168066496)

[6.2. Testpiemēru kopa 25](#_Toc168066497)

[6.3. Testēšanas žurnāls 25](#_Toc168066498)

[7. Individuālais ieguldījums 26](#_Toc168066499)

[7.1. 2D mākslas pieredze 26](#_Toc168066500)

[7.2 3D mākslas pieredze 26](#_Toc168066501)

[7.3 Unity pieredze 26](#_Toc168066502)

[7.4 C# pieredze 26](#_Toc168066503)

[7.5 Vispārējā pieredze 26](#_Toc168066504)

[8. Secinājumi 27](#_Toc168066505)

[9. Lietoto terminu un saīsinājumu skaidrojumi 28](#_Toc168066506)

[Literatūras un informācijas avotu saraksts 29](#_Toc168066507)

[Mākslinieki 29](#_Toc168066508)

# 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.

## 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. Līmeņa dati
2. 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 līmeņa

Ievaddati:

1. Izvēlētā līmeņa faila nosaukums
2. līmeņa izvēlētais grūtums

Nosacījumi:

1. Līmeņa failiem ir jābūt noteiktā direktorija
2. Izsauc tikai vienu reizi pēc 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 direktorija

Nosacījumi:

1. Ir jābūt kaut vienam līmenim, iekšā direktorijai

Apstrāde:

1. Izlasa un izvada katra līmeņa nosaukums un grūtības līmeņus
2. Ja ir specificēts parāda PNG failu un maina līmeņa apmales krāsu specificētam 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. Tikai tiek rēķināta, ja lietotājs ir jebkurā līmenī
2. Tikai skaita punktus un dzīvību pēc sākuma 2 sekundes gaidīšanas periodu
3. Netiek rēķināta pēc līmeņa pabeigšanas

#### Apstrāde:

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

**1.tabula**

**Funkcijas apstrāde**

|  |  |  |  |
| --- | --- | --- | --- |
|  | Īsā klikšķa bulta gadījumā | garu spiedienu bultas gadījumā | Neviena bulta laukā |
| Poga tiek nospiesta tieši laikā | bonuss 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 nospiesta | Nav punkti iedoti, dzīvība samazinās | Nav punkti iedoti, dzīvība samazinās | neskaita |
| Poga nospiesta laikā un atlaista perfekti | Bonuss punktus un dzīvību iedod | Bonuss punktus un dzīvību iedod | neskaita |
| Poga nospiesta laikā, bet atlaista par vēlu/ātru | Bonuss 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. Bultas vienmēr ir ritmiski kopā ar mūziku

#### Apstrāde:

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

### 2.2.15. Īsā klikšķa bultiņa

#### Mērķis:

Ja “Spēles noteikto bultu ielādēšana” ielādē īsa klikšķa bultiņu, šis nosaka visu kas notiks ar to.

#### Ievaddati:

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

#### Nosacījumi:

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

#### Apstrāde:

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

### 2.2.16. Garu spiedienu bultiņa

#### Mērķis:

Ja kāda no spēles tekstūrām ir modificētas “Resources”

#### Ievaddati:

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

#### Nosacījumi:

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

#### Apstrāde:

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

### 2.2.17. Fotogrāfijas un tekstūras apstrāde mainīšanas gadījumā.

#### Mērķis:

Izveidot funkciju kas reaģēs ja lietotājs ir mainījis tekstūras failus, vienalga vai mainot vai izdzēšot.

#### Ievaddati:

1. Failu direktorija/faili.

#### Nosacījumi:

1. Jaunas tekstūras tiek izmantotas spēlē.
2. Ja faili ir izdzēsti, tie tiek aizvietoti izmantojot nemodificējamus failus spēles datu direktorijā.

#### Apstrāde:

1. Pēc modifikācijas tekstūras mainās.

### 2.2.18. Sākumlapas kursora sekojos aizmugures lauks

#### Mērķis:

Izveidot interesantu izskatu sākumlapā, kas seko peles kursoram un maina aizmugures lauka izskatu.

#### Ievaddati:

1. Peles kursora atrašanās vieta

#### Nosacījumi:

1. Peles kursors netiek ietekmēts
2. Aizmugures lauks kustās minimāli
3. Nevar redzēt fotogrāfijas galus

#### Apstrāde:

1) peles koordinātes nosaka kameras atrašanās vietu un rotāciju

## 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.

* Inkscape

Šī ir vel vienā programma kuru var izmantot, lai veidotu 2D dizainus, viņa ir atvērta programma ar vairākām funkcijām kas ir vairāk noderīga precizitātei un izmēriem.

### 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.

* Unreal Engine

Šis ir cits dzinis kas ir populāri lietots. Tam ir vairāki uzlabojumi kas nav unity, kā piemēram labāka ēnotāja kontrole, apgaismojuma sistēma un ģenerāli labāk optimizēta. Lai gan es regulāri to neizmantoju un nav daudz pieredze. Unreal ir arī vienkāršojusi vairākas sadaļas kas nozīmē būs mazāk jāraksta skripti.

* Godot

Godut ir alternatīvs Unity spēles dzinis, šis ir daudz mazāk populārs nekā Unity vai Unreal. Godut izmanto 4 valodas, GDScript, C#, Via un C++. No tām man ir tikai pieredze ar C# un C++. Godot, ir iepriekš lietots, bet nav pietiekami daudz pieredze, vai mentori.

* 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

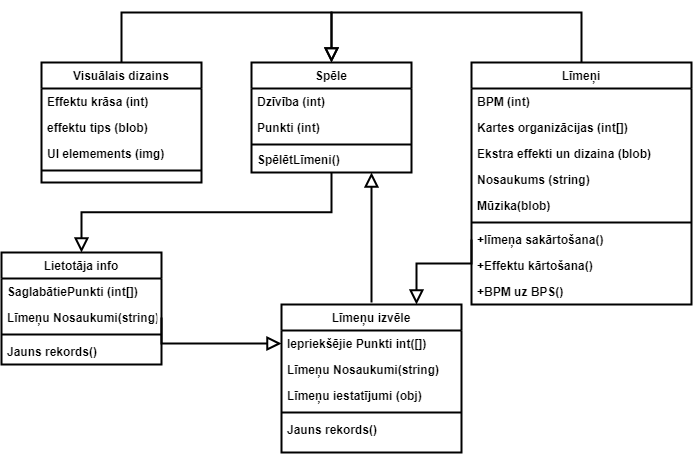
Lai veidotu līmeņus ātrāk ir iespēja veidot alternatīvu programmu kas izmantojot JSON formātu atļaus spēlēt korespondējošo mūziku un demonstrācijas skatu no videospēles skatupunkta. Šādi veidota programmatūra ir specifiska un tikai izmantojama ar cybershock JSON kārtojuma formātu.

# 4. Sistēmas modelēšana un projektēšana

## 4.1. Sistēmas struktūras modelis

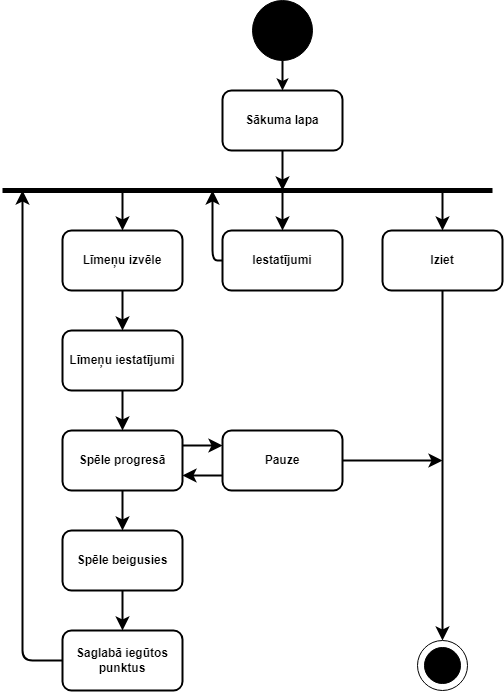
1. attēls Sistēmas struktūras modelis

## 4.2. Klašu diagramma

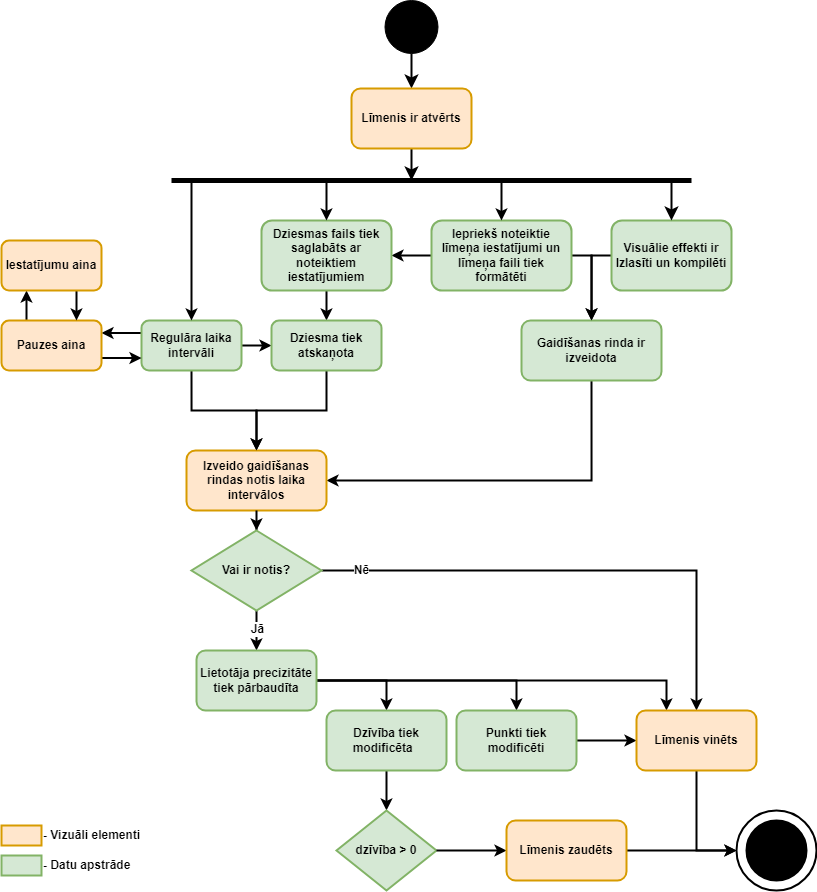
Šī diagramma reprezentē datus kuri tiks izmantoti projektā. Katra sadaļa ir individuālā failā vai faila direktorijā. 

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

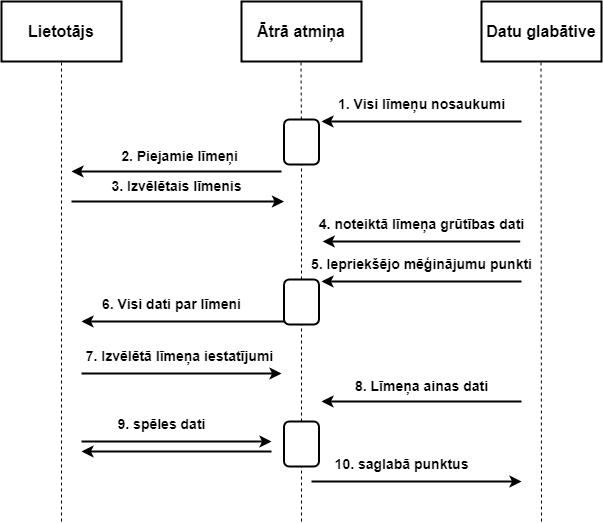
Šī diagramma ir domāta, lai reprezentētu “spēle progresā” nodaļu un labāk izskaidrotu kā lietotāji pārvietosies cauri spēles elementiem un kā tie mainīsies relatīvi padotajiem datiem un lietotāju reakciju.

**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

Šī nodaļa ir domāta, lai izskaidrotu lietotāju navigāciju un kā spēlēt spēli. Pogu atrašanās vieta paliks vienāda, bet ir iespējams ka grafiski dizaina elementi var mainīties, vai dēļ jaunas versijas, vai pēc modifikatorā izvēles.

## 5.1 Sākumlapa



**7. attēls Sākumlapa**

Play pārvietos tevi uz 5.3 nodaļu kurā varēs izvēlēties kuru līmeni vēlies spēlēt.

Settings pārvietos tevi uz 5.2 nodaļu kurā varēs mainīt iestatījumus.

Quit aizver aplikāciju.

## Iestatījumi

**8. attēls Iestatījumi**

Resolution izmaina izšķirtspēju kurā spēle tiks vadīta, pa kreisi ir mazāka un pa labi palielina izšķirtspēju.

Screen mode maina kā spēle tiks vadīta caur windows, pa kreisi ir “Windowd” kas nozīmē ka izskatīsies tā kā jebkura cita aplikācija, ja ir “Borderless window” tad tas aizņems visu ekrānu, bet varēs iziet ārā no programmas un izmantot citu ekrānu neaizverot programmu. “Full screen” aizņems visu ekrānu un ja nospiedīs uz jebkuru citu aplikāciju programma minimizēs sevi.

Quality iestatījumi maina cik detalizēti izskatīsies spēle. Konkrētas iespējas ir izskaidrotas tabulā, kur kreisā puse norāda zemākas kvalitātes iestatījumus kas ir pieejami un pa labi rāda visas augstākas kvalitātes iestatījumus.

**2. tabula**

**Kvalitātes iestatījumi**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| “Very low” | “Low” | “Medium” | “High” | “Very high” |
| Daļiņas ir ļoti samazinātas, fotogrāfijas rezolūcija ir samazināta un gaismas efekti it izslēgti. | Daļiņas ir limitētas bet ir vairāk nekā “Very low”. Tekstūrām izšķirtspēja ir mazliet samazināta. Daži gaismas efekti ir redzami | Daļiņas nav limitētas un foto izšķirtspēja ir orģinālā lielumā. Gaismas efekti ir mazliet limitēti un vienkārši aprēķināti | Daļiņu efekti ir ieslēgti, apgaismojumi ir optimāli veidoti.  Apgaismojuma efekti ir vairāk un atspoguļojas pret citiem objektiem | Ir pieliktas vel daļiņas, efekti un foto izšķirtspēja tiek palielināta. |

Volume vada kopējo aplikācijas skaļumu, kā piemēru ja mūzikas skaļums ir 50% un volume ir arī 50% tad kopējais mūzikas skaļums būs 25%.

SFX ir efektu skaļums, tās ir skaņas kas tiek atskaņotas ja nospiež pogas, vai spēlēšanas laikā efektu skaņas.

Music maina mūzikas skaļumu, tas ir gan spēles laikā, gan ceļvedī.

Back pārvieto tevi atpakaļ uz 5.1 nodaļu.

Apply saglabā iestatītos iestatījumus.

## 5.3 līmeņu izvēle

**9. attēls līmeņu izvēle**

Līmeņa izvēles nodaļa – šis saraksts satur visas spēles līmeņus, ja pāri tiem pāriet ar pelīti tā palielinās un ja tu uzklikšķini tās fotogrāfija un nosaukums paradās kreisajā pusē un maza daļa no mūzikas tiek atskaņota. Lai ietu cauri sarakstam var izmantot peles ritenīti vai turot kreiso taustiņu lejā un kustinot peli, līdzīgi kā saraksti telefonos.

Kreisā nodaļa sastāv ar izvēlēto dziesmas foto un nosaukumu. Ja neviena nav izvēlēta, kā piemēram ja ir tikko atvērts, spēle izvēlas kādu nejauši.

Zem tā ir grūtības iestatījumu saraksts. Katrai mūzikai var būt citi nosaukumi, krāsas un grūtības iestatījumu skaits. Kad kāda grūtības izvēle ir nospiesta tā tiek atzīmēta un “play” poga ir pieejamā.

Back poga ignorēs kas tika nospiests un pāries uz sākumlapu.

Play poga ir tikai pieejama, ja ir izvēlēta grūtības izvēle. Šī poga pārvietos tevi uz pašu spēli ar iestatīto mūziku un grūtību.

# 6. Testēšanas dokumentācija

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

## 6.2. Testpiemēru kopa

## 6.3. Testēšanas žurnāls

# 7. Individuālais ieguldījums

Šī nodaļa ir domāta, lai demonstrētu iegūto pieredzi taisot šo programmu. Tā būs sadalīta vairākās daļās, katra daļa ir domāta citam aspektam kurā es uzlaboju savas zināšanas. Kā piemēram 3D un 2D māksla, Unity un C# pieredze.

## 7.1. 2D mākslas pieredze

Lielāka daļa māksla nāca no komisijām, viņu info ir redzami 10 sekcijā (Literatūras un informācijas avotu saraksts). Es ieguvu pieredzi pareizi prasīt, izskaidrot un iedot references foto priekš jauniem mākslas darbiem. Lai izmantotu šos foto vajadzēja pārveidot, lai to darītu man vajadzēja izmantot Photoshop un Inkscape.

## 7.2 3D mākslas pieredze

3D objekti ir ļoti vienkārši šajā programma, bet es joprojām ieguvu pieredzi definēt un sakārtot UV mapes, materiālu sakārtošanu un “mirrored editing” kas ļauj man veidot simetriskus objektus vienkāršāk.

## 7.3 Unity pieredze

Mani galvenie ieguldījumi ir izmantojot Unity, UI taisīšana, trigera izmantošana 3D vidē, laika apstrāde, ainas apstrāde un skriptu sadalīšana. Konkrēti saistībā ar Unity dzini, es vairāk iemācījos kā izmantot “Project Settings”, lai modificētu spēles sajūtu un kopējo izskatu.

## 7.4 C# pieredze

Pirms taisot šo projektu man bija ļoti minimāla pieredze ar jebkuru C versiju, tādēļ es ieguvu ļoti daudz pieredzi kā izmantot C# sintaksi un ģenerālu lietošanu Unity ekosistēmā.

## 7.5 Vispārējā pieredze

Manuprāt, es ieguvu pieredzi strādāt videospēles vidē un kā video spēļu industrija organizē projektus. Tas ir kopumā, gan kā māksla tiek organizēta, dokumentācijā tiek rakstīta izstrādes laikā un spēles dizains. Šīs prasmes ļoti palīdzēja manā praksē, kā arī varēju izmantot pieredzi no prakses lai izmantotu uz šī projekta.

# 8. Secinājumi

Galvenokārt, es Izvēlējos labu projekta ideju kurai man bija motivācija piestrādāt un uzlabot. Šī “rave” interese nav samazinājusies un vel joprojām mani interesē šī stila mūzika. Pie tam es vel joprojām izmantoju Unity, lai gan jaunāku versiju. Es izvēlējos 2.5D spēli kas ļoti izaicināja manu spēju integrēt abu veidu grafikus, kā arī kā tos modificēt.

Viss nebija perfekti, Unity versijā kuru es izmantoju ir ar dažām problēmām, vis īpaši ar daļiņu ģenerācijas sadaļu un UI izkārtojumu. Tie abi padarīja spēles izstrādi daudz grūtāku nekā bija plānots. Pat ja taisot 2.5D iedeva man pieredzi strādāt ar jaunām funkcijām un iedeva man daudz pieredzi, bet es nevarēju izmantot manu specializāciju, 3D mākslu. Ja es taisītu 3D spēli es arī varētu izrādīt vairāk savas prasmes rakstīt skriptus.

Kopumā es varēju izplānot labāku projekta ideju un vairāk ieskatīties maniem izstrādes instrumentiem, bet taisot šo projektu es iemācījos daudz vairāk kā izmantot Unity rīkus un 2D mākslas apstrādi. Kā arī man vel ir motivācija turpināt izstrādāt šo spēli ar vairāk funkcijām un jaunām tekstūrām.

# 9. Lietoto terminu un saīsinājumu skaidrojumi

1. 2.5D – Šis termins nozīmē ka spēle tiek rēķināta un taisīta kā 3d spēle, bet izmanto vairākus 2d elementus un visuāli izskatās gan 3D un kā 2D vienlaicīgi
2. Aizmugur lauks - Fotogrāfija vai grafiks kas ir aiz UI
3. Beat – Mūzikas termins kas norāda kad viena nots, vai pauze tiek spēlēta, tā kā šī ir ritma spēle, es izmantoju to pašu terminu, lai apzinātu laika fragmentu.
4. BPM – “Beats Per Minute” ir kā tek saglabāts mūzikas ātrums, jeb cik beat ir vienā minūtē, šajā dziesmā.
5. Bulta/nots – bulta vai nots tiks izmantoti kā sinonīmi. Tie būs objekti kuri reprezentēs katru beat.
6. Cyberpunk – māksliniecisks stils kas savieno kiber tehnoloģijas dizainus ar punk stiliem, cyberpunk ir vairākas subkultūras un stili.
7. Daļiņas – Daļiņas, jeb partikui ir vizuāli objekti, kā piemēram putekļi, uguns, dūmi vai efekti. Jebkurš objekts kas ir 3 dimensijunāls, bez 3D modeļiem.
8. Rave – Mūzikas koncerts/stils kas spēlē ātru elektronisko mūziku, pasti pārtaisītas populāras dziesmas kuras ir jauktas kopā un ar palielinātiem BPM
9. Spēles dzinis – Programma domāta lai veidotu videospēles
10. Spraits – Objekts ar foto kurš tiks izmantots spēles gaitā.
11. UI – "User Interface" ir interakcija ar 2d interaktīvām daļām kas izmanto peli.

# Literatūras un informācijas avotu saraksts

* <https://docs.unity3d.com/Packages/com.unity.textmeshpro@2.2/manual/index.html>
* <https://docs.unity3d.com/2019.4/Documentation/Manual/>
* <https://docs.unity3d.com/2019.4/Documentation/ScriptReference/index.html>

## Mākslinieki

* Raine, izveidoja nots dizainu, pogas un logo: <https://www.instagram.com/togetherlast?igsh=MTRmNmkxMDFzbnYyMA==>
* Alexie, izveidoja mapes dizainu un dzīvību:

<https://youtube.com/@alaxiety?si=NMLhPJn8c_cnDkHW>