

„Liepājas Valsts tehnikums”

**Datoru spēle "Escape Protocol"**

Kvalifikācijas eksāmena praktiskās daļas tehniskā dokumentācija

|  |  |
| --- | --- |
| Izglītības programma | **33484011 Programmēšana** |
| Profesionālā kvalifikācija | **Programmēšanas tehniķis** |

|  |  |
| --- | --- |
| Darba autors: | Džonatans Svilis-Sudints |

Eksāmena datums 2025.gada \_\_.jūnijs

Liepāja 2025

**Saturs**

[Ievads 3](#__RefHeading___Toc4114_288123008)

[1. Uzdevuma formulējums 4](#__RefHeading___Toc4116_288123008)

[2. Programmatūras prasību specifikācija 5](#__RefHeading___Toc4124_288123008)

[2.1. Produkta perspektīva 5](#__RefHeading___Toc4126_288123008)

[2.2. Sistēmas funkcionālās prasības 5](#__RefHeading___Toc4128_288123008)

[2.3. Sistēmas nefunkcionālās prasības 19](#__RefHeading___Toc4130_288123008)

[2.4. Gala lietotāja raksturiezīmes 19](#__RefHeading___Toc4132_288123008)

[3. Izstrādes līdzekļu, rīku apraksts un izvēles pamatojum 21](#__RefHeading___Toc4134_288123008)

[3.1. Izvēlēto risinājuma līdzekļu un valodu apraksts 21](#__RefHeading___Toc4136_288123008)

[4. Sistēmas modelēšana un projektēšana 22](#__RefHeading___Toc4138_288123008)

[4.1. Sistēmas struktūras modelis 22](#__RefHeading___Toc4140_288123008)

[4.1.1. Izvietojuma diagramma 22](#__RefHeading___Toc4142_288123008)

[4.1.2. komponentu diagramma 22](#__RefHeading___Toc4144_288123008)

[4.2. Funkcionālais un dinamiskais sistēmas modelis 22](#__RefHeading___Toc4146_288123008)

[4.2.1. Aktivitāšu diagramma 22](#__RefHeading___Toc4148_288123008)

[4.2.2. Lietojumgadījumu diagramma 22](#__RefHeading___Toc4150_288123008)

[4.2.3. Stāvokļu diagramma 22](#__RefHeading___Toc4152_288123008)

[4.3. Datu struktūru apraksts 22](#__RefHeading___Toc4154_288123008)

[5. Lietotāju ceļvedis 23](#__RefHeading___Toc4156_288123008)

[6. Testēšanas dokumentācija 24](#__RefHeading___Toc4158_288123008)

[6.1. Izvēlētās testēšanas metodes, rīku apraksts un pamatojums 24](#__RefHeading___Toc4160_288123008)

[6.2. Alternatīvās testēšanas metodes un rīki 24](#__RefHeading___Toc4162_288123008)

[6.3. Testpiemēru kopa 24](#__RefHeading___Toc4164_288123008)

[6.4. Testēšanas žurnāls 24](#__RefHeading___Toc4166_288123008)

[Secinājumi 25](#__RefHeading___Toc4168_288123008)

[7. Lietoto saīsinājumu un terminu skaidrojums 26](#__RefHeading___Toc4170_288123008)

[Literatūras un informācijas avoti 27](#__RefHeading___Toc4172_288123008)

[7.1. Iespējamo risinājuma līdzekļu un valodu apraksts 28](#__RefHeading___Toc4174_288123008)

# Ievads

Šī dokumentācija apraksta spēles ideju, kuras pamatā ir automatizācijas un izdzīvošanas mehāniku apvienošana divdimensiju spēļu pasaulē. Darba autors piedāvā spēles koncepciju, kurā galvenais varonis nokļūst uz svešas, procedurāli ģenerētas planētas, kas ir bagāta ar resursiem, piemēram, dzelzi, varu, akmeni un ogli. Spēles mērķis ir izdzīvot un izveidot raķeti, kas ļautu pamest planētu, izmantojot automatizāciju, resursu ieguvi un to pārstrādi. Spēles galvenais mērķis ir apvienot izdzīvošanas mehānikas ar tehnoloģisko progresu, veidojot sarežģītas ražošanas ķēdes un veicinot spēlētāja stratēģisko domāšanu.

Darba autors piedāvā apvienot automatizācijas brīvību ar izdzīvošanas izaicinājumiem, padarot katru spēles sesiju unikālu un stratēģiski daudzveidīgu. Automatizācija kļūst par neatņemamu izdzīvošanas sastāvdaļu, bet veiksmīgai spēlei nepieciešama rūpīga plānošana un resursu pārvaldība. Šis dokuments ir paredzēts spēles izstrādes pamatprincipu aprakstam un galveno mehāniku definēšanai.

Pieļaujams vairākas potenciālās mērķauditorijas grupas, kuras varētu interesēt šī spēle. Spēles mehāniku un tematikas kombinācija padara to pievilcīgu gan izdzīvošanas spēļu cienītājiem, gan tiem, kas aizraujas ar automatizāciju un stratēģisko plānošanu. Spēle piedāvā izaicinājumu ne tikai izdzīvot, bet arī optimizēt un uzlabot ražošanas procesus, kas var piesaistīt arī tehnoloģiju entuziastus, kuri meklē kaut ko vairāk par vienkāršu izdzīvošanu.

Darba autors izvēlējās šo tēmu, jo, pirmkārt, viņu ļoti interesē spēļu izstrāde, un tā ir joma, ar kuru viņš vēlētos nodarboties visu savu dzīvi. Viņš ne tikai aizraujas ar spēlēm, bet arī to struktūru un spēļu dzinējiem. Otrkārt, viņam ļoti patīk spēlēt spēles, kurās ir nepieciešams domāt, rēķināt un optimizēt. Tāpēc, iedvesmojoties no spēles Factorio, vēlas izveidot savu projektu. Papildus tam, šī spēles izstrāde ir mērķis izzināt, kā spēles un spēļu dzinēji darbojas no iekšienes, lai gūtu profesionālu pieredzi. Viņš arī vēlas popularizēt izvēlēto spēļu dzinēju, lai tam pievērstu lielāku uzmanību, un lai citi saprastu, ka, lai izstrādātu savu spēli, nav nepieciešams izvēlēties vislielāko un smagāko spēļu dzinēju.

Spēles izstrāde ir arī iespēja parādīt, ka ar radošumu un pareizu pieeju var radīt aizraujošu spēļu pieredzi, izmantojot resursus efektīvi un pieejami visiem. Šis projekts ne tikai piedāvā izaicinājumu spēlētājiem, bet arī kalpo kā iedvesma citiem izstrādātājiem, demonstrējot, ka inovācijas un kvalitāte var nākt no dažādiem spēļu dzinējiem un pieejām, ne tikai no visplašāk pazīstamajiem un izmantotajiem.

# Uzdevuma formulējums

Spēles izstrādes process tiks sadalīts vairākos secīgos posmos, lai nodrošinātu projekta strukturētu un mērķtiecīgu attīstību.

Pirmajā posmā paredzēts izstrādāt procedurālas pasaules ģenerēšanas algoritmus. Šis mehānisms nodrošinās, ka katra jaunā spēles sesija tiek veidota ar unikālu karti, kur resursu, objektu un reljefa izvietojums būs atšķirīgs. Šim nolūkam tiks izmantota Perlina trokšņa tehnoloģija (Perlin noise), kas ļauj radīt dabiskus un reālistiskus reljefus. Šāda pieeja veicina spēles atkārtotu spēlējamību un paplašina lietotāja pieredzi, piedāvājot dinamisku un neparedzamu vidi.

Otrajā posmā tiks realizēta spēles mehānikas izstrāde, kas iekļauj resursu ieguves, automatizācijas un būvniecības sistēmas ieviešanu. Šīs sistēmas spēlētājiem sniegs iespēju projektēt efektīvas ražošanas ķēdes un pārvaldīt pieejamos resursus. Papildus tiks ieviesta izdzīvošanas mehānika, kas spēlētājiem radīs nepieciešamību izstrādāt stratēģijas, lai veiksmīgi pārvarētu izaicinājumus nakts laikā. Šīs mehānikas kombinācija nodrošina spēles dinamiku un stratēģiskās domāšanas attīstību.

Trešajā posmā tiks veikta spēles testēšana un līdzsvarošana, lai pārliecinātos par tās funkcionalitāti un stabilitāti. Šajā posmā uzmanība tiks pievērsta kļūdu novēršanai un spēles grūtības pakāpes sabalansēšanai, lai nodrošinātu patīkamu un līdzsvarotu spēlēšanas pieredzi.

Lai konstatētu, ka mērķis ir sasniegts, tiks izmantoti šādi kritēriji:

1. Spēlei jābūt pilnībā pabeigtai un spēlējamai no sākuma līdz beigām, nenonākot pie tehniskām kļūdām vai spēles procesu traucējumiem.
2. Tiks sasniegts pietiekams kvalitātes līmenis, kas ļauj spēli teorētiski publicēt Steam vai citās digitālajās platformās.
3. Testēšanas procesā tiks apstiprināts, ka spēle ir interesanta, izaicinoša un tehniski stabila.

Programmatūras produkta nepieciešamība ir pamatota ar tādu mērķi, kā piedāvāt lietotājiem kvalitatīvu un interesantu brīvā laika pavadīšanas iespēju. Šī spēle apvieno izklaides elementus ar prāta attīstību. Tā rosina spēlētājus analizēt situācijas, izstrādāt efektīvas stratēģijas un optimizēt ražošanas procesus.

Papildus izklaides vērtībai spēle piedāvā arī izglītojošu dimensiju, jo spēlētāji mācīsies pārvaldīt ierobežotus resursus un plānot to efektīvu izmantošanu. Spēles nakts laikā paredzētie izaicinājumi liek spēlētājiem domāt taktiski un adaptēties mainīgiem apstākļiem. Šī kombinācija padara spēli par intelektuāli stimulējošu un emocionāli aizraujošu pieredzi, kas spēj piesaistīt plašu auditoriju.

# Programmatūras prasību specifikācija

Šajā sadaļā ir detalizēti aprakstīts, kā šis projekts atšķiras no līdzīgajiem, izceļot tā unikālās priekšrocības un funkcionalitāti, kā arī norādītas sistēmas funkcionālās un nefunkcionālās prasības. Tāpat šeit tiks apskatītas gala lietotāju raksturīgās īpašības.

## Produkta perspektīva

Šis projekts ir unikāls, jo apvieno izdzīvošanas spēļu žanru ar automatizācijas elementiem, radot jaunu pieredzi divdimensiju spēļu pasaulē. Atšķirībā no citām spēlēm, kas varētu fokusēties vai nu uz izdzīvošanu, vai uz automatizācijas sistēmām, mūsu spēle piedāvā procedurāli ģenerētu pasauli. Katra spēles sesija ir unikāla, pateicoties kartes ģenerācijai, kas izmanto Perlina trokšņa algoritmus, lai radītu dabiskus un mainīgus reljefus, resursu un briesmu izvietojumu.

Dziļa automatizācijas līmeņa ieviešana ļauj spēlētājiem būvēt sarežģītas ražošanas ķēdes, kur automatizācija nav tikai izvēle, bet nepieciešamība izdzīvošanai. Šāda pieeja prasa spēlētājiem ne tikai izdzīvot, bet arī domāt par ilgtermiņa stratēģijām un optimizāciju.

Izdzīvošanas izaicinājumi ar nakts režīmu pievieno papildu aspektu, kur spēlētājiem ir jāadaptējas un jāaizsargājas pret briesmām, kas ir atšķirīgs no daudzām automatizācijas spēlēm, kur fokuss ir tikai uz ražošanu.

Spēle mudina spēlētājus ne tikai būvēt un automatizēt, bet arī plānot, pārvaldīt resursus un pielāgoties mainīgiem apstākļiem, kas ir retāk sastopams automatizācijas žanra spēlēs.

Šī spēle var piesaistīt gan izdzīvošanas spēļu cienītājus, gan tos, kas aizraujas ar automatizāciju un stratēģisko plānošanu, paplašinot tās potenciālo lietotāju loku.

Šīs priekšrocības un atšķirības no citām spēlēm padara projektu izcilu savā žanrā, piešķirot spēlētājiem jaunas iespējas un izaicinājumus, kas apvieno divus populārus spēļu mehānismus vienā saskaņotā pieredzē.

## Sistēmas funkcionālās prasības

**PR.1. Galvenais izvēlnes ekrāns**

Mērķis:

* Nodrošināt spēlētājam piekļuvi spēles pamata funkcijām un iestatījumiem.

Ievaddati:

* Spēlētāja ierīces ieejas metodes (peles klikšķi, tastatūras taustiņi).
* Ievadītais spēles pasaules nosaukums.

Apstrāde:

* Izveidot procedurāli ģenerētu spēles pasauli
* Saglabāt jaunas spēles pasaules nosaukumu un sākotnējos datus spēles failā.

Izvaddati:

* Parādīt galvenā ekrāna fona attēlu un spēles logotipu.
* Izveidot un parādīt opcijas: "Jauna spēle", "Turpināt", "Iestatījumi", "Iziet".
* Pēc izvēles "Jauna spēle" un ievadītā nosaukuma, pārvietot spēlētāju uz jauno spēles pasauli.

**PR.2. Spēles turpināšana**

Mērķis:

* Ļaut spēlētājam atsākt iepriekšējo spēles sesiju.

Ievaddati:

* Spēlētāja ierīces ieejas metodes (peles klikšķi, tastatūras taustiņi).

Apstrāde:

* Izgūt un apstrādāt sarakstu ar saglabātajām spēles pasaulēm no failiem.
* Ielādēt izvēlēto spēles pasauli un tās datus, piemēram, spēlētāja stāvokli, resursus un pasauli.

Izvaddati:

* Parādīt sarakstu ar pieejamajiem saglabājumiem un ar papildu informāciju par katru (pēdējā spēlēšanas laiks, spēles nosaukums u.c.).
* Pēc izvēles konkrēta saglabājuma, pārvietot spēlētāju uz spēles ainu ar ielādēto saglabājumu.

**PR.3. Iestatījumi**

Mērķis:

* Ļaut spēlētājam pielāgot spēles iestatījumus, lai uzlabotu pieredzi.

evaddati:

* Spēlētāja ierīces ieejas metodes (peles klikšķi vai tastatūras taustiņi).

Apstrāde:

* Izgūt pašreizējos iestatījumus, piemēram, skaņas līmeņus no spēles konfigurācijas.
* Pielāgot skaņas līmeņus atbilstoši spēlētāja regulējumiem, atjauninot konfigurācijas failu.

Izvaddati:

* Parādīt ekrānu ar iestatījumiem, kurā ir redzams pašreizējais skaņas līmenis mūzikai un efektiem.
* Izvadīt slīdņus, lai spēlētājs varētu pielāgot skaņas līmeņus.

**PR.4. Iziet no spēles**

Mērķis:

* Nodrošināt drošu spēles aizvēršanu pēc spēlētāja pieprasījuma.

Ievaddati:

* Spēlētāja ierīces ieejas metodes (peles klikšķi vai tastatūras taustiņi).

Apstrāde:

* Parādīt apstiprināšanas dialogu, lai pārliecinātos, ka spēlētājs tiešām vēlas iziet no spēles.
* Gaidīt spēlētāja apstiprinājumu vai atcelšanu.

Izvaddati:

* Parādīt apstiprinājuma logu ar jautājumu "Vai tiešām vēlaties iziet no spēles?" un pogām "Jā" un "Nē".
* Pēc apstiprinājuma "Jā", aizvērt spēles logu un iziet no spēles.

**PR.5. Spēlētāja pārvietošanās**

Mērķis:

* Ļaut spēlētājam pārvietoties spēles pasaulē, izmantojot tastatūru.

Ievaddati:

* Spēlētāja tastatūras ievades (W, A, S, D taustiņi).

Apstrāde:

* Interpretēt taustiņu nospiešanu kā kustības komandas: W - uz priekšu, S - atpakaļ, A - pa kreisi, D - pa labi.
* Aprēķināt jauno spēlētāja pozīciju, ņemot vērā kustības virzienu, ātrumu un iespējamos šķēršļus
* Atjaunināt spēlētāja pozīciju spēles datu struktūrā.

Izvaddati:

* Atjaunināt spēles ainu, mainot spēlētāja avatara pozīciju uz ekrāna atbilstoši saņemtajām kustības komandām.

**PR.6. Resursu ieguve**

Mērķis:

* Ļaut spēlētājam iegūt resursus spēles pasaulē, izmantojot peli.

Ievaddati:

* Spēlētāja peles kreisā klikšķa ievade.
* Resursa veids (koks, ogle, dzelzs, varš, akmens) un tā atrašanās vieta spēles pasaulē.

Apstrāde:

* Noteikt, kurš resurss tiek iegūts, pamatojoties uz spēlētāja mērķa punktu.
* Sākt dēstīšanas laika atskaiti atbilstoši resursa veidam:
  + Koks - 2 sekundes.
  + Ogle, dzelzs, varš, akmens - 4 sekundes.
* Pēc atskaites beigām, pievienot iegūto resursu spēlētāja inventāram.

Izvaddati:

* Redzams vizuālais efekts, ka resurss tiek dēstīts (progresijas josla).
* Pēc dēstīšanas laika beigām, resurss parādās spēlētāja inventārā.
* Koks pazūd no spēles pasaules uzreiz pēc dēstīšanas sākuma.
* Pēc dēstīšanas laika beigām, resursi (ogle, dzelzs, varš) parādās spēlētāja inventārā, bet rūdas dzīsla paliek vietā.

**PR.7. Inventāra atvēršana**

Mērķis:

* Ļaut spēlētājam atvērt inventāra saskarni.

Ievaddati:

* Spēlētāja tastatūras ievade (E taustiņš).

Apstrāde:

* Atvērt inventāra saskarni.
* Attēlot 20 šūnas, kas paredzētas resursiem un komponentiem.

Izvaddati:

* Parādīt inventāra ekrānu ar tukšām šūnām, gatavu pieņemt resursus un komponentus.

**PR.8. Inventāra pārvaldība**

Mērķis:

* Ļaut spēlētājam apskatīt, pārvaldīt un izgatavot komponentus.

Ievaddati:

* Šobrīd esošie resursi un komponenti inventārā.
* Spēlētāja izvēle izgatavošanas saskarnē.

Apstrāde:

* Attēlot katrā šūnā esošos resursus vai komponentus, to daudzumu.
* Aktivizēt izgatavošanas saskarni, kur spēlētājs var izvēlēties receptes.
* Pārbaudīt, vai spēlētājam ir pietiekami daudz resursu izvēlētās receptes izpildei.
* Ja resursi ir pietiekami, izgatavot komponentus un pievienot tos inventārā.

Izvaddati:

* Atjaunināt inventāra ekrānu, parādot attiecīgos resursus un jaunizgatavotos komponentus.

**PR.9. Komponentu un būvju izgatavošana**

Mērķis:

* Ļaut spēlētājam izgatavot komponentus vai būves no inventārā esošajiem resursiem.

Ievaddati:

* Spēlētāja ievade (klikšķis uz vēlamā komponenta vai būves receptes inventārā).

Apstrāde:

* Pārbaudīt, vai spēlētājam ir pietiekami daudz nepieciešamo resursu izvēlētā komponenta vai būves izgatavošanai.
* Ja resursi ir pieejami, samazināt inventārā esošo resursu skaitu atbilstoši receptes prasībām.
* Izveidot izvēlēto komponentu vai būvi un pievienot to inventāram.
* Šis process notiek momentāni, bez papildu gaidīšanas laika.

Izvaddati:

* Ja komponents vai būve ir izgatavota, tas momentāni parādās inventārā.
* Resursi, kas tika izmantoti izgatavošanā, tiek noņemti no inventāra, un to skaits tiek atjaunināts, lai atspoguļotu izmaiņas.

**PR.10. Būvju izveidošana**

Mērķis:

* Ļaut spēlētājam izvietot būves spēles pasaulē pēc to izgatavošanas inventārā.

Ievaddati:

* Spēlētāja ievade (klikšķis uz būves inventārā).

Apstrāde:

* Atvērt inventāru, lai spēlētājs varētu izvēlēties būvi.
* Pēc būves izvēles un klikšķa uz tās, aizvērt inventāru.
* Parādīt būves projekciju spēles pasaulē, kas seko spēlētāja kursora, norādot, kur būve tiks izvietota.

Izvaddati:

* Inventārs aizveras pēc būves izvēles.
* Spēles pasaulē parādās būves projekcija, kas norāda potenciālo būvniecības vietu, kas mainās atbilstoši spēlētāja kursora kustībai.

**PR.11. Būvju rotācija**

Mērķis:

* Ļaut spēlētājam pagriezt būves projekciju pirms tās izvietošanas.

Ievaddati:

* Spēlētāja tastatūras ievade (Q taustiņš).

Apstrāde:

* Kad būves projekcija ir aktīva, reaģēt uz Q taustiņa nospiešanu.
* Pagriezt būves projekciju par 45 grādiem pulksteņrādītāja virzienā katru reizi, kad Q taustiņš tiek nospriegts.
* Atjaunināt projekcijas vizuālo attēlojumu, lai atspoguļotu pagriezienu.

Izvaddati:

* Parādīt pagrieztu būves projekciju spēles pasaulē, mainot tās orientāciju par 45 grādiem katru reizi, kad tiek nospriegts Q taustiņš.

PR.11. Būvju izvietošana

Mērķis:

* Ļaut spēlētājam beidzot izvietot būvi spēles pasaulē un saglabāt tās stāvokli un pozīciju.

Ievaddati:

* Spēlētāja peles kreisā klikšķa ievade.

Apstrāde:

* Pēc kreisā klikšķa, pārbaudīt, vai izvēlētā vieta atļauj būvniecību (piemēram, nav šķēršļu vai citu būvju).
* Ja būvniecība ir atļauta, izvietot būvi spēles pasaulē saskaņā ar projekcijas pozīciju un orientāciju.
* Atņemt izmantoto būvi no inventāra.
* Saglabāt būves koordinātas, orientāciju un citus nepieciešamos datus spēles saglabāšanas failā.

Izvaddati:

* Būve parādās spēles pasaulē tieši tur, kur tika izvēlēta ar projekciju.
* Inventārā vairs nav būves, kas tika izvietota.
* Saglabājuma sistēmā tiek fiksētas būves pozīcija un orientācija, lai tās būtu pieejamas nākamajā sesijā.

**PR.12. Būvju nojaukšana**

Mērķis:

* Ļaut spēlētājam nojaukt būves un atgūt tās atpakaļ inventārā.

Ievaddati:

* Spēlētāja peles labā klikšķa ievade un tā noturēšana virs būves.

Apstrāde:

* Kad spēlētājs tur labo peles taustiņu virs būves, sākt nojaukšanas procesa atskaiti (2 sekundes).
* Parādīt nojaukšanas progresu ar vizuālu indikatoru (progresijas josla).
* Pēc 2 sekundēm, nojaukt būvi un pievienot to atpakaļ spēlētāja inventārā.
* Atjaunināt saglabājuma datus, noņemot nojaukto būvi no spēles pasaules.

Izvaddati:

* Parādīt progresijas joslu, lai rādītu, cik ilgi vēl jātur peles taustiņš, lai nojauktu būvi.
* Pēc nojaukšanas, būve pazūd no spēles pasaules un parādās inventārā.

**PR.13. Interakcija ar būvēm**

Mērķis:

* Ļaut spēlētājam mijiedarboties ar būvēm, lai piegādātu resursus, iestatītu ražošanu vai iegūtu produktus.

Ievaddati:

* Spēlētāja ievade (klikšķis uz būves, lai atvērtu tās inventāru).

Apstrāde:

* Atvērt specifisku inventāra logu atbilstoši būves tipam (turrets, industriālais bloks.
* Turrets:
  + Ļaut spēlētājam ievietot munīciju (patronas).
* Industriālais bloks:
  + Ļaut spēlētājam piegādāt nepieciešamos resursus.
  + Iestatīt, kurš komponents vai produkts tiks ražots, izmantojot pieejamās receptes.
  + Parādīt ražošanas progresu.
  + Atļaut izņemt ražotos komponentus vai produktus, kad tie ir gatavi.
* Atjaunināt būves statusu un saglabāt izmaiņas spēles failā.

Izvaddati:

* Parādīt būves specifisko inventāru, kas ļauj pārvaldīt munīciju, resursus, ražošanas receptes.
* Ja runa ir par turrets, spēlētājs redz, cik munīcijas ir pieejams, un var papildināt krājumus.
* Industriālo bloku gadījumā, spēlētājs redz resursu slotus, ražošanas iespējas un gatavos produktus, ko var izņemt.
* Vizuāli norādīt ražošanas progresu, ja būve kaut ko ražo.

**PR.14. Konveijera integrācija ar būvēm**

Mērķis:

* Ļaut spēlētājam pievienot konveijerus pie turrets vai industriālo bloku, lai tie automātiski piegādātu vai izvadītu priekšmetus.

Ievaddati:

* Konveijera izvietošana un savienošana ar būvēm (turrets vai industriālajiem blokiem).
* Pieejamie priekšmeti uz konveijera vai būvēs.

Apstrāde:

* Kad konveijers ir savienots ar būvi, automātiski pārbaudīt, vai ir priekšmeti, kas jāpārvieto.
* Ja konveijers ir pievienots pie turrets, tas piegādā munīciju no konveijera uz turrets.
* Ja konveijers ir pievienots pie industriālā bloka, tas piegādā nepieciešamos resursus ražošanai vai izvada ražotos priekšmetus.
* Atjaunināt būves un konveijera statusu, lai atspoguļotu priekšmetu pārvietošanu.

Izvaddati:

* Parādīt vizuālo efektu vai animāciju, kas norāda, ka priekšmeti pārvietojas pa konveijeru uz vai no būves.
* Atjaunoties būves inventārs, lai parādītu, ka munīcija vai resursi ir piegādāti vai, otrādi, gatavie produkti ir izvadīti.

**PR.15. Pārvietošana priekšmetu pa konveijeru**

Mērķis:

* Ļaut priekšmetiem pārvietoties pa konveijeru sistēmu, neprasa konfigurāciju no spēlētāja, vienkārši pārvietojot priekšmetus pa līniju.

Ievaddati:

* Priekšmeti, kas ir novietoti uz konveijera sākuma vai pievienoti no blakus esošām būvēm.

Apstrāde:

* Automātiski pārvietot priekšmetus pa konveijera līniju virzienā, kādā konveijers ir novietots.
* Priekšmeti turpina pārvietoties, līdz sasniedz konveijera beigas vai tiek pārvirzīti uz citu konveijeru vai būvi.
* Sistēma neprasa spēlētāja iejaukšanos; priekšmeti pārvietojas pa konveijeru 90 grādu pagriezienos, ja konveijers tā ir izvietots.

Izvaddati:

* Parādīt vizuālo efektu vai animāciju, kas attēlo priekšmetu pārvietošanos pa konveijeru.
* Priekšmeti, kas bija konveijera sākumā, pakāpeniski pārvietojas uz tā galu vai uz citu savienotu konveijeru/būvi.

**PR.16. Dienas un nakts cikls**

Mērķis:

* Ļaut spēlei automātiski mainīt dienas/nakts ciklu, lai radītu dinamisku spēles vidi.

Ievaddati:

* Spēles laika sistēma, kas seko līdzi spēles iekšējam laikam.

Apstrāde:

* Sākot spēli, uzstādīt dienas laiku.
* Pēc 5 minūtēm spēles laika, pārslēgties uz nakts laiku.
* Pēc vēl 5 minūtēm, atgriezties pie dienas laika.
* Šis cikls turpinās bez pārtraukuma, mainoties starp dienu un nakti ik pēc 5 minūtēm spēles laika.
* Atjaunināt vides apgaismojumu un atmosfēru.

Izvaddati:

* Mainīt spēles pasaules apgaismojumu, lai atspoguļotu dienas vai nakts laiku.

**PR.17. Ienaidnieku spavns naktī**

Mērķis:

* Palielināt spēles izaicinājumu, spavnojot ienaidnieku mobus naktī, ar pieaugošu skaitu katru nakti.

Ievaddati:

* Spēles laika sistēma, kas norāda, ka ir nakts.

Apstrāde:

* Pirmā nakts: spavnot 1 mobu ārpus spēlētāja redzes lauka (aiz ekrāna robežas).
* Katru nākamo nakti, palielināt mobu skaitu par 1.5 reizi no iepriekšējās nakts.
* Skaitu apaļot uz augšu, lai nodrošinātu veselu mobu skaitu (piemēram, 1.5 = 2 mobi, 2.25 = 3 mobi).
* Ienaidnieki spavnojas nejaušās vietās aiz ekrāna robežas, bet pietiekami tuvu, lai varētu ietekmēt spēlētāju.

Izvaddati:

* Parādīt jaunos ienaidniekus, kas sāk kustēties uz spēlētāja pusi no spavn vietām.
* Spēles pasaulē var redzēt vai dzirdēt pazīmes, ka ienaidnieki ir spavnojušies (piemēram, skaņas efekti, vizuālas norādes).
* Spēles interfeisā vai pasaulei apkārtējās indikācijās var būt redzams, ka ienaidnieku skaits vai draudi ir palielinājušies.

**PR.18. Mobu pārvietošanās uz spēlētāju**

Mērķis:

* Iedot mobiem spēju pārvietoties spēles pasaulē ar mērķi tuvošanos spēlētājam.

Ievaddati:

* Mobu atrašanās vieta pēc spavna vai pēc pēdējās kustības.
* Spēlētāja pašreizējā pozīcija.

Apstrāde:

* Noteikt spēlētāja pozīciju attiecībā pret katra moba atrašanās vietu.
* Aprēķināt optimālo maršrutu vai virzienu, kurā mobam pārvietoties uz spēlētāja pusi, ņemot vērā spēles pasaules šķēršļus vai topogrāfiju.
* Pārvietot mobus vienmērīgā ātrumā vai ar noteiktu ātruma algoritmu uz spēlētāja virzienu.
* Ja spēlētājs kustās, mobiem ir jāpielāgo savs ceļš vai jāmaina virziens, lai turpinātu sekot spēlētājam

Izvaddati:

* Parādīt mobus, kas pārvietojas uz spēlētāja virzienu, mainot savu pozīciju spēles pasaulē.
* Vizualizēt mobu kustību ar animācijām vai kustības efektiem, lai spēlētājs varētu redzēt, ka mobi tuvojas.
* Iespējams, radīt skaņas efektus, lai norādītu mobu kustību vai tuvināšanos spēlētājam.

**PR.19. Mobu spēja iznīcināt būves**

Mērķis:

* Ļaut mobiem iznīcināt būves, ja tās atrodas uz ceļa uz spēlētāju, lai radītu papildu izaicinājumus un spriedzi spēlē.

Ievaddati:

* Mobu atrašanās vieta un virziens, kurā tie pārvietojas.
* Būves, kas atrodas starp mobiem un spēlētāju.

Apstrāde:

* Kad mobs sastopas ar būvi, kas bloķē ceļu uz spēlētāju, sākt būves iznīcināšanas procesu.
* Kad būve ir pilnībā iznīcināta, atjaunināt spēles pasauli, noņemot būvi no tās atrašanās vietas.

Izvaddati:

* Parādīt animāciju vai vizuālo efektu, ka mobs uzbrūk vai iznīcina būvi.
* Spēles pasaulē būve pakāpeniski zaudē savu izturību vai izskatu, līdz tā pilnībā izzūd, atstājot vietu mobam turpināt ceļu uz spēlētāju.
* Iespējams, pievienot skaņas efektus, kas atspoguļo būves iznīcināšanu

**PR.20. Mobu uzbrukums spēlētājam**

Mērķis:

* Radīt reālu draudu spēlētāja veselībai, kad mobi pietuvojas pietiekami tuvu, lai uzbruktu.

Ievaddati:

* Mobu atrašanās vieta attiecībā pret spēlētāju.
* Spēlētāja pašreizējais veselības stāvoklis (sākotnēji 20 vienības).

Apstrāde:

* Kad mobs ir pietiekami tuvu spēlētājam (piemēram, ietvaros noteiktā rādiusā), sākt uzbrukuma mehāniku.
* Katrs mobs, kas ir tuvu pietiekami, lai uzbruktu, ik sekundi nodara spēlētājam 2 vienības veselības zaudējumu.
* Atjaunināt spēlētāja veselības punktu skaitu, samazinot to par 2 punktiem katru sekundi, kamēr mobs ir uzbrukuma rādiusā.

Izvaddati:

* Parādīt vizuālo indikatoru vai animāciju, kas norāda, ka mobs uzbrūk spēlētājam.
* Atjaunināt spēles interfeisā spēlētāja veselības joslu vai indikatoru, lai atspoguļotu veselības zaudēšanu.
* Pievienot skaņas efektus, kas atspoguļo uzbrukumu.

**PR.21. Spēlētāja nāve**

Mērķis:

* Informēt spēlētāju par spēles zaudējumu, kad viņa veselība sasniedz 0, un sniegt informāciju par spēles ilgumu.

Ievaddati:

* Spēlētāja veselības stāvoklis (sasniedz 0).
* Laiks, cik ilgi spēlētājs ir spēlējis pašreizējā sesijā.

Apstrāde:

* Kad spēlētāja veselība sasniedz 0, apturēt spēles procesus un aktivizēt nāves ekrānu.
* Saskaitīt spēles laiku kopš sesijas sākuma līdz nāves brīdim.
* Sagatavot modālo logu vai ekrānu ar paziņojumu, ka spēlētājs ir miris.

Izvaddati:

* Parādīt modālo logu ar tekstu, piemēram, "Jūs esat miris".
* Iekļaut informāciju par to, cik ilgi spēlētājs bija izdzīvojis šajā spēles sesijā (piemēram, "Jūs spēlējāt: [laiks]").
* Piedāvāt opcijas: "Atgriezties galvenajā izvēlnē" vai "Mēģināt vēlreiz".

**PR.22. Spēlētāja uzbrukums mobiem**

Mērķis:

* Ļaut spēlētājam uzbrukt mobiem ar lielāku uzbrukuma rādiusu nekā mobiem, un bez kavējuma starp uzbrukumiem, balstoties uz spēlētāja ievadi.

Ievaddati:

* Spēlētāja peles kreisā klikšķa ievade.
* Mobu atrašanās vieta attiecībā pret spēlētāju.

Apstrāde:

* Noteikt uzbrukuma rādiusu, kas ir lielāks spēlētājam nekā mobiem, lai spēlētājs varētu uzbrukt no lielāka attāluma.
* Katrs kreisā peles taustiņa klikšķis izraisa uzbrukumu, nodarot 2 veselības punktu zaudējumu tuvākajam mobam, kas atrodas uzbrukuma rādiusā.
* Nav uzbrukuma laika kavējuma; spēlētājs var uzbrukt tik bieži, cik spēj klikšķināt.
* Ja moba veselība sasniedz 0, noņemt to no spēles pasaules.

Izvaddati:

* Parādīt vizuālus un skaņas efektus, kas norāda uz uzbrukumu (piemēram, animācijas, skaņas efekti).
* Atjaunināt moba veselības joslu vai indikatoru, lai parādītu zaudēto veselību.
* Ja mobs tiek nogalināts, animācija vai efekts, kas parāda nāvi.

**PR.23. Turrets uzbrukums mobiem**

Mērķis:

* Nodrošināt, ka turrets automātiski uzbrūk mobiem, nodrošinot aizsardzību spēlētājam ar plašāku uzbrukuma rādiusu un nepārtrauktu uguni.

Ievaddati:

* Mobu atrašanās vieta attiecībā pret turrets.
* Turrets status (munīcija).

Apstrāde:

* Noteikt turrets uzbrukuma rādiusu, kas ir lielāks nekā spēlētāja uzbrukuma rādiusu.
* Turrets automātiski uzbrūk visiem mobiem, kas ieiet tās uzbrukuma rādiusā, bez vajadzības pēc papildu komandām.
* Katras 0.5 sekundes turrets izšauj vienu lodi, nodarot 1 veselības punktu zaudējumu.
* Turrets var šaut 360 grādos, tātad uzbrukt mobiem no visām pusēm.
* Atjaunināt mobu veselību un pārbaudīt, vai tie ir jānoņem no spēles pasaules, ja veselība sasniedz 0.

Izvaddati:

* Parādīt vizuālo efektu lodes lidošanai no turrets uz mobu.
* Skaņas efekti, kas norāda uz šaušanu un trāpījumiem.
* Atjaunināt mobu veselības indikatorus, lai parādītu zaudēto veselību.
* Ja mobs tiek nogalināts, attēlot tā nāves animāciju vai efektu.

**PR.24. Pasaules ģenerācija**

Mērķis:

* Izveidot dinamisku un mainīgu spēles pasauli, izmantojot procedurālo ģenerāciju ar dažādiem šumiem, lai radītu dabiskas ainavas.

Ievaddati:

* Ģenerācijas sēkla vai nejaušības faktors, kas nodrošina unikālās pasaules izveidi.

Apstrāde:

* Izmantot vienu šuma algoritmu (Perlin noise) zemes un ūdens ģenerācijai, kas nosaka reljefa augstumu un ūdens līmeņus.
* Otrs šuma algoritms ar atšķirīgiem parametriem tiek izmantots resursu un biotopu izvietojuma ģenerācijai, piemēram, meži, minerāli.
* Izveidot divdimensiju masīvu, kur katrs elements attēlo vienu bloku spēles pasaulē.
* Katrai šūnai piešķirt atbilstošu tipu (zeme, ūdens, koks, minerāls utt.) balstoties uz šumu datiem.
* Šo procesu veikt vienu reizi spēles sesijas sākumā vai pie jaunas pasaules izveides, lai izveidotu pamata pasaules struktūru.

Izvaddati:

* Spēles pasaule, kas vizuāli attēlo ģenerēto reljefu, ūdeni, resursus un biotopus, balstoties uz izmantotajiem šumiem.
* Spēles dati tiek saglabāti kā 2D masīvs, kas tiek izmantots, lai atspoguļotu spēles pasauli spēlētājam, ieskaitot to, kur viņš var staigāt, kur atrodas resursi un šķēršļi.

**PR.25. Dinamiskā pasaules ģenerācija**

Mērķis:

* Optimizēt spēles veiktspēju, ģenerējot tikai nepieciešamo pasaules daļu un pievienojot jaunas teritorijas, kad spēlētājs tuvojas kartes malām.

Ievaddati:

* Spēlētāja pozīcija pasaulē.
* Aktuāli ģenerētās pasaules izmēri un robežas.

Apstrāde:

* Sākotnēji ģenerēt tikai mazu kartes daļu ap spēlētāja sākuma punktu, izmantojot tās pašas šuma algoritmus kā pilnai ģenerācijai.
* Izveidot "buffer zonu" ap pašreizējo ģenerēto pasauli, kas nav redzama spēlētājam, bet gatava ātrai ģenerācijai.
* Kad spēlētājs tuvojas pašreizējās kartes robežai, automātiski sākt ģenerēt jaunas kartes daļas, izmantojot iepriekš definētos šuma algoritmus, lai nodrošinātu vienmērīgu pāreju un nepārtrauktu pasaules struktūru.
* Šis process notiek fonā, lai izvairītos no spēles bremzēšanas, kad spēlētājs pārvietojas.
* Atjaunināt pasaules karti, pievienojot jaunas šūnas masīvam, kas attēlo spēles pasauli.

Izvaddati:

* Spēlētājs redz vienmērīgi paplašinātu pasauli, kas šķiet bezgalīga, bet patiesībā tikai nepieciešamās daļas tiek ģenerētas reālā laikā.
* Pasaules izmaiņas ir praktiski nemanāmas spēlētājam, jo jaunas daļas tiek pievienotas pirms spēlētājs sasniedz esošās kartes robežu.
* Veiktspēja tiek optimizēta, jo tiek izmantoti resursi tikai tai pasaules daļai, kas tiešām ir vai drīz būs spēlētāja redzes laukā.

**PR.26. Raķetes būvēšana un palaišana**

Mērķis:

* Noslēgt spēles sesiju, pabeidzot galveno mērķi - raķetes palaišanu, un parādīt spēlētājam, cik ilgi viņš spēlēja, lai to paveiktu.

Ievaddati:

* Visi nepieciešamie komponenti raķetes būvniecībai.
* Spēlētāja darbības, lai sāktu raķetes palaišanu.

Apstrāde:

* Kad spēlētājs ir savācis un piegādājis visus nepieciešamos resursus un komponentus raķetei, iespējot tās palaišanu.
* Pēc raķetes palaišanas apturēt spēles procesus.
* Saskaitīt laiku, kas pagājis kopš spēles sesijas sākuma līdz raķetes palaišanai.

Izvaddati:

* Parādīt modālo logu vai ekrānu, kas informē spēlētāju par spēles pabeigšanu, piemēram, "Misija izpildīta!".
* Iekļaut informāciju par laiku, cik ilgi spēlētājs spēlēja, lai sasniegtu šo mērķi, piemēram, "Jūs spēlējāt: [laiks]".
* Iespējams, piedāvāt opcijas, piemēram, "Spēlēt vēlreiz" vai "Atgriezties galvenajā izvēlnē".

**PR.27. Sāglabājuma dzēšana**

Mērķis:

* Ļaut spēlētājam droši dzēst nevajadzīgus spēles saglabājumus, piedāvājot apstiprinājumu, lai izvairītos no nejaušas datu zaudēšanas.

Ievaddati:

* Spēlētāja izvēle izdzēst konkrētu saglabājumu no saraksta.

Apstrāde:

* Kad spēlētājs izvēlas dzēšanas opciju blakus kādam no saglabājumiem, parādīt apstiprinājuma dialogu.
* Apstiprinājuma dialogā jautāt, vai spēlētājs tiešām vēlas dzēst šo saglabājumu, ar iespējām "Jā" un "Nē".
* Ja spēlētājs apstiprina ar "Jā", izdzēst attiecīgo failu, kur glabājas saglabājums.
* Atjaunināt saglabājumu sarakstu, lai atspoguļotu izmaiņas.

Izvaddati:

* Parādīt apstiprinājuma dialogu ar jautājumu, piemēram, "Vai jūs tiešām vēlaties dzēst šo saglabājumu?" un pogām "Jā" un "Nē".
* Pēc apstiprināšanas dzēšanas, atjaunināt sarakstu ar saglabājumiem, noņemot dzēsto saglabājumu.

## Sistēmas nefunkcionālās prasības

**NR.1. Veiktspēja**

* Spēlei jānodrošina gluds kadru ātrums (FPS) vismaz 60 kadri sekundē pie vidējās klases aparatūras.

**NR.2.** **Drošība**

* Spēles saglabājumi jāaizsargā pret datu zudumiem, nodrošinot regulārus automātiskus saglabājumus.

**NR.3. Lietotāja pieredze**

* Saskarnei jābūt intuitīvai, lai spēlētāji varētu ātri apgūt spēles pamata mehānikas.

**NR.4. Saderība**

* Spēle jāizstrādā tā, lai tā darbotos dažādās operētājsistēmās (piemēram, Windows, macOS, Linux).

**NR.5. Uzticamība**

* Spēlei jābūt stabilai, ar minimālu kļūdu skaitu, kas varētu traucēt spēlēšanas procesu.

## Gala lietotāja raksturiezīmes

Paredzētais spēles lietotājs ir jaunietis vecumā no 12 līdz 19 gadiem. Šie ir jaunieši, kuriem ir daudz brīvā laika un kuri meklē ne tikai izklaidi, bet arī iespēju pārdomāt sarežģītākas situācijas spēles gaitā.

Šiem lietotājiem, visticamāk, patīk 2D spēles, kas prasa izaicinājumus un piedāvā sarežģītākas spēles mehānikas. Spēle ir veidota kā "smilšu kaste", kas nozīmē, ka lietotājam pašam jāprot sevi izklaidēt un veidot spēles pieredzi, jo tajā nav konkrēta sižeta, bet ir izvirzīti mērķi, kurus spēlētājam jāsasniedz.

Lietotājs, kurš izvēlēsies šo spēli, ir radošs, pacietīgs un gatavs ieguldīt laiku, lai atklātu un izmantotu spēles piedāvātās iespējas. Šī spēle būs īpaši pievilcīga tiem, kuri meklē līdzsvaru starp izklaidi un izaicinājumiem.

# Izstrādes līdzekļu, rīku apraksts un izvēles pamatojum

Šajā nodaļā darba autors apraksta programmēšanas rīkus, kurus izvēlējās savas kvalifikācijas darba izstrādei, aprakstot, ko viņš izvēlējās un kāpēc, kā arī ko viņš neizvēlējās. Izvēloties programmatūru, tika izvirzīti vairāki kritēriji. Pirmkārt, lai programma būtu viegla un neprasītu daudz datora resursu. Otrkārt, lai tai būtu ērts un patīkams programmatūras interfeiss. Treškārt, lai programma būtu bezmaksas un pieejama visiem lietotājiem.

## Izvēlēto risinājuma līdzekļu un valodu apraksts

**Godot Engine**

Šis spēļu dzinējs piedāvā daudz iespēju gan 2D, gan 3D spēļu izstrādei. Tas ir ļoti viegls un ātri darbojas. Salīdzinot ar konkurentiem, tam ir vairākas priekšrocības. Pirmkārt, pateicoties tā vieglumam, tas ielādējās ātrāk nekā citi spēļu dzinēji un var tikt palaists no USB ierīces. Otrkārt, tam nav nepieciešama koda kompilācija spēles izstrādes laikā, kas nozīmē, ka, rakstot kodu spēlei, tā var tikt palaista un jebkuri jaunie koda fragmenti, kas tikko ir saglabāti, momentāni stājas spēkā, un izmaiņas ir redzamas pat bez spēles pārstartēšanas.

**GDScript**

Iebūvētā programmēšanas valoda Godot Engine spēļu dzinējā. Koda rakstīšana notiek tieši spēļu dzinējā, tāpēc nav nepieciešams atvērt atsevišķu koda redaktoru, viss notiek iekšējā ekosistēmā. Šī programmēšanas valoda ir balstīta uz Python, un tās ir diezgan līdzīgas. GDscript tika izvēlēts tā labās integrācijas dēļ ar spēļu dzinēju, kas vienkāršo un paātrina darbu. Papildus tam, spēļu dzinējā ir iebūvēta dokumentācija šai valodai, kas atvieglo apmācību. No trūkumiem jāatzīmē, ka, salīdzinot ar C#, veiktspēja būs nedaudz zemāka, taču tas ir jūtams tikai lielos 3D projektos, kur katrs kadrs ir svarīgs. 2D spēlēm izvēlētā programmēšanas valoda ir ideāli piemērota.

**GitHub Desktop**

Tika izvēlēts projekta versijas kontrolei. Šis lietojumprogramma tiek izmantota darbam ar GIT, tai ir patīkams un, galvenais, ērts grafiskais interfeiss. Lai saglabātu projektu uz GitHub, ir jānospiež tikai 2 pogas, un projekts automātiski tiks nosūtīts uz serveriem. Tāpat var redzēt izmaiņas, kas veiktas katrā failā, un ar vienas pogas nospiešanu tās var atcelt pilnībā vai daļēji.

## Iespējamo (alternatīvo) risinājuma līdzekļu un valodu apraksts

**Unity**

Šis spēļu dzinējs netika izvēlēts, pamatojoties uz vairākiem kritērijiem. Pirmkārt, šis programmatūra ir nosacīti bezmaksas. Tas nozīmē, ka ne visas spēļu dzinēja funkcijas ir pieejamas bezmaksas versijā, un spēle, kas izstrādāta ar Unity un gūst peļņu, ir spiesta dalīties ar ienākumiem ar uzņēmumu, kas attīsta spēļu dzinēju. Otrkārt, šis spēļu dzinējs nav viegls. Tas aizņem daudz diska vietas un prasa ilgu laiku, lai ielādētos, salīdzinot ar Godot Engine. Turklāt, veiktspēja nav tā spēcīgā puse. Šis spēļu dzinējs prasa daudz resursu ne tikai spēles izstrādes laikā, bet arī spēles darbībai. Lielo 3D spēļu optimizācija, izmantojot šo dzinēju, ir ļoti sarežģīta, un liela daļa laika tiek veltīta optimizācijai, nevis pašai izstrādei. Arī koda kompilācija pēc katra izmaiņas prasa laiku, jo pirms spēles palaišanas kods ir jākompilē, kas ievērojami palielina izstrādes laiku.

**C#**

Programmēšanas valoda, kas labi noder aplikāciju un spēļu izstrādei. Netika izvēlēta tāpēc, ka Godot Engine piedāvā daudz labāku integrāciju ar GDscript. Tas nozīmē, ka, lai izstrādātu vienādas metodes, C# prasītu vairāk laika algoritma rakstīšanai.

**Git**

Tiek izmantots koda versijas kontrolei, un visas darbības notiek konsolē. Programma labi pilda savus pienākumus, tomēr tā neatbilst izvēlētajiem kritērijiem šim projektam, jo tai nav grafiskā interfeisa.

**Visual Studio Code**

Viegls koda redaktors, kas bieži tiek izmantots programmatūras izstrādei. Netika izvēlēts, jo Godot Engine piedāvā iebūvētu koda redaktoru, kurā ir ērtāk izstrādāt spēli.

# Sistēmas modelēšana un projektēšana

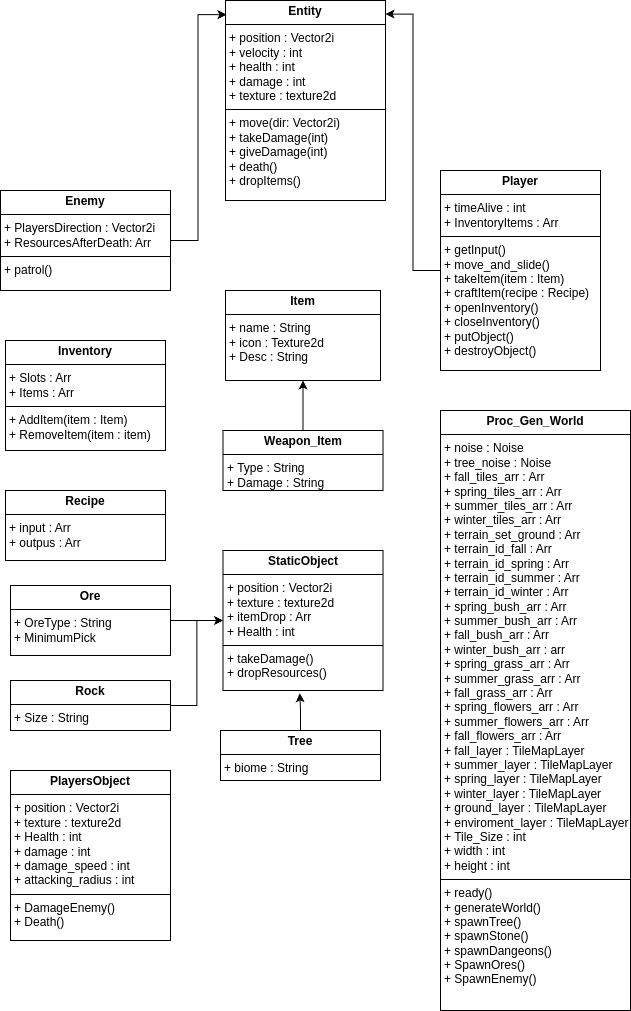
Šajā sadaļā tiks iekļautas dažādas diagrammas, kas vizuāli ataino izstrādātās spēles darbības loģiku un tās galvenos procesus.

## Sistēmas struktūras modelis

Sistēmas struktūras modelis attēlo izstrādātās spēles galvenās sastāvdaļas un to savstarpējās saistības. Šajā modelī tiek uzskatāmi parādīti spēles galvenie moduļi, piemēram, lietotāja interfeiss (UI), spēles loģika, pasaules ģenerēšana, skaņas un animāciju vadība, kā arī inventāra un kraftēšanas sistēmas. Diagramma palīdz saprast, kā dažādas sistēmas daļas sadarbojas savā starpā, nodrošinot vienotu un funkcionējošu spēles vidi.

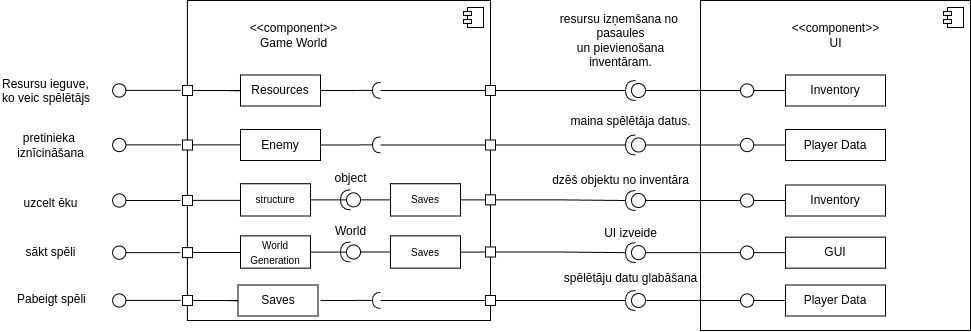
### Klašu diagramma

Klašu diagramma attēlo spēles galveno objektu struktūru un to savstarpējās attiecības. Šajā diagrammā ir definētas galvenās klases, piemēram, Entity, Player, Enemy, Inventory, Item, Recipe, Proc\_Gen\_World u.c., kur katrai klasei ir norādīti tās atribūti un metodes. Diagramma uzskatāmi parāda mantošanas attiecības, piemēram, Player un Enemy paplašina bāzes klasi Entity, kā arī objekts Weapon\_Item ir specializācija no Item. Šāda vizualizācija ļauj saprast spēles arhitektūru objektorientētā līmenī, kā arī atvieglo turpmāko sistēmas izstrādi un uzturēšanu.

1. attēls. klašu diagramma

### komponentu diagramma

Komponentu diagramma attēlo sistēmas dalījumu neatkarīgos komponentos, kas veido spēles arhitektūras pamatu. Diagrammā redzami divi galvenie komponenti – Game World un UI, kuri satur iekšējos apakškomponentus ar noteiktām atbildībām. Komponents Game World ietver tādus elementus kā Resources, Enemy, World Generation, Saves un citus, kas ir atbildīgi par spēles loģiku un pasaules stāvokļa pārvaldību. Komponents UI koncentrējas uz spēlētāja interfeisu, ietverot Inventory, Player Data un GUI. Diagramma arī uzskatāmi parāda savstarpējo saistību starp komponentiem, nodrošinot skaidru un modulāru sistēmas uzbūvi.

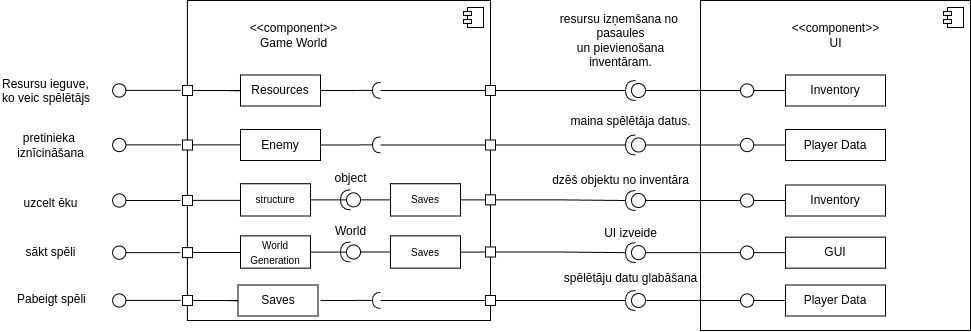
2. attēls. komponentu diagramma

## Funkcionālais un dinamiskais sistēmas modelis

Šī nodaļa ietver spēles funkcionālā un dinamiskā modeļa aprakstu. Funkcionālais modelis parāda, kādas darbības lietotājs var veikt spēlē, savukārt dinamiskais – kā sistēma uzvedas dažādos spēles brīžos. Apakšnodaļās attēlotas aktivitāšu, lietojumgadījumu un stāvokļu diagrammas, kas kopā ilustrē gan spēlētāja rīcību, gan objektu uzvedības scenārijus.

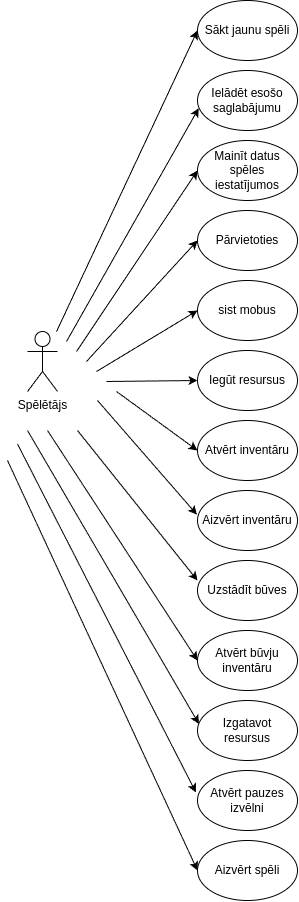
### Aktivitāšu diagramma

Šajā aktivitāšu diagrammā attēlots spēles lietotāja darbību secīgums, sākot no galvenās izvēlnes līdz aktīvai spēles norisei. Diagramma parāda galvenos procesus, piemēram, pasaules ģenerēšanu, spēlētāja pārvietošanos, mijiedarbību ar objektiem, resursu vākšanu, kraftēšanu un cīņu ar pretiniekiem. Tā palīdz vizuāli saprast spēles loģisko plūsmu un to, kā lietotājs ietekmē sistēmas darbību dažādos spēles brīžos.

3. attēls. Aktivitāšu diagramma

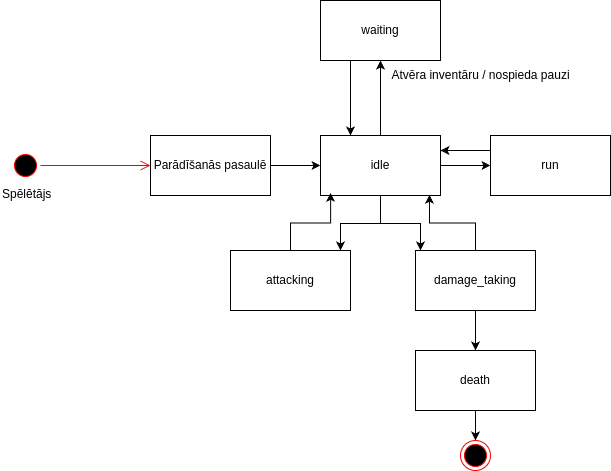
### Lietojumgadījumu diagramma

Lietojumgadījumu diagramma atspoguļo galvenās darbības, ko spēlētājs var veikt spēles laikā. Tajā iekļauti tādi scenāriji kā jaunas spēles uzsākšana, saglabājumu ielāde, pārvietošanās, resursu iegūšana, cīņa ar pretiniekiem, priekšmetu kraftēšana, būvju uzstādīšana un izvēlņu izmantošana. Diagramma palīdz strukturēti attēlot spēlētāja iespējamos mijiedarbības veidus ar sistēmu.

4. attēls. Lietojumgadījumu diagramma

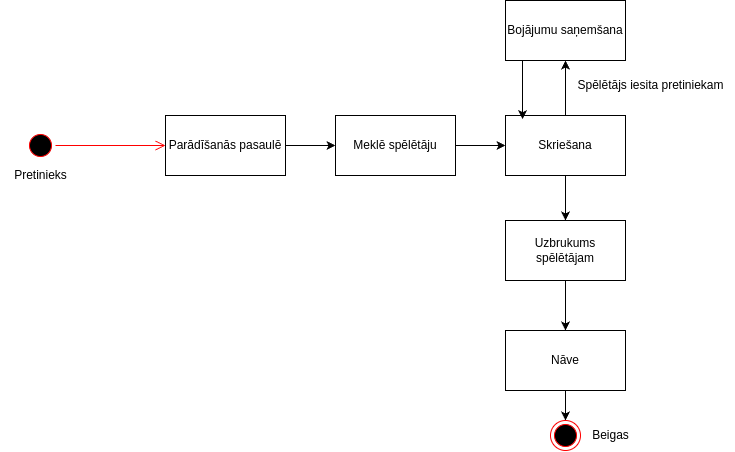
### Stāvokļu diagramma

Stāvokļu diagramma attēlo spēlētāja objekta uzvedības ciklu, parādot iespējamos stāvokļus un pārejas starp tiem spēles laikā. Diagrammā redzams, kā spēlētājs no parādīšanās pasaulē nonāk dažādos stāvokļos. Šis modelis palīdz saprast spēlētāja loģisko darbību secību un reakciju uz apkārtējās vides izmaiņām.



5. attēls. Stāvokļu diagramma Player

Diagramma attēlo pretinieka stāvokļus spēlē – no parādīšanās pasaulē līdz nāvei. Tajā redzami galvenie uzvedības posmi: spēlētāja meklēšana, skriešana, uzbrukums un bojājumu saņemšana.

6. attēls. Stāvokļu diagramma Pretinieks

## Datu struktūru apraksts

**Masīvi (Array)** tiek plaši izmantoti spēles pasaules ģenerēšanā, piemēram, lai uzglabātu dažādu elementu (zāle, krūmi, koki u.c.) datus, kurus vēlāk iespējams ģenerēt uz kartes. Masīvs ir efektīva struktūra fiksēta lieluma datu glabāšanai, kas nodrošina ātru piekļuvi pēc indeksa.

**Vector2i** tiek izmantots pozīcijas un koordinātu attēlošanai pasaulē. Atšķirībā no Vector2, kas glabā peldošā komata vērtības, Vector2i izmanto veselus skaitļus (int), kas padara to precīzāku un piemērotāku pozicionēšanai režģa (tile-based) pasaulē, kur objektiem ir jāatrodas konkrētās šūnās

**Resource** Godot dzinī tiek izmantots kā datu konteiners, kas ļauj definēt pielāgotas datu struktūras, piemēram, priekšmetus, receptes vai spēles iestatījumus. Tā ir klasēs balstīta struktūra, kuru var saglabāt, ielādēt un rediģēt redaktorā, nodrošinot ērtu atkārtotu izmantošanu un datu modularitāti.

# Lietotāju ceļvedis

# Testēšanas dokumentācija

## Izvēlētās testēšanas metodes, rīku apraksts un pamatojums

## Alternatīvās testēšanas metodes un rīki

## Testpiemēru kopa

## Testēšanas žurnāls

# Secinājumi

# Lietoto saīsinājumu un terminu skaidrojums

**Mobs** - Spēlēs mobi ir jebkuri kustīgi objekti, kas var mijiedarboties ar spēlētāju, piemēram, dzīvnieki vai ienaidnieki.

**NR** - Nefunkcionālā prasība,

**Perlin trokšņis** - Tas ir algoritms, kas rada dabisku izskatošos, gludus šumus, ko izmanto, lai ģenerētu pasauli spēlēs, piemēram, ainavas un tekstūras,

**PR** - Funkcionālā prasība.

# Literatūras un informācijas avoti

## Iespējamo risinājuma līdzekļu un valodu apraksts