„Liepājas Valsts tehnikums”

Datorspēle "Survive 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
1. Uzdevuma formulējums....4
2. Programmatūras prasību specifikācija....5
2.1. Produkta perspektīva....5
2.2. Sistēmas funkcionālās prasības....5
2.3. Sistēmas nefunkcionālās prasības....16
2.4. Gala lietotāja raksturiezīmes....16
3. Izstrādes līdzekļu, rīku apraksts un izvēles pamatojum....17
3.1. Izvēlēto risinājuma līdzekļu un valodu apraksts....17
3.2. Iespējamo (alternatīvo) risinājuma līdzekļu un valodu apraksts....18
4. Sistēmas modelēšana un projektēšana....19
4.1. Sistēmas struktūras modelis....19
4.1.1. Klašu diagramma....19
4.1.2. komponentu diagramma....20
4.2. Funkcionālais un dinamiskais sistēmas modelis....21
4.2.1. Aktivitāšu diagramma....21
4.2.2. Lietojumgadījumu diagramma....22
4.2.3. Stāvokļu diagramma....23
4.3. Datu struktūru apraksts....24
5. Lietotāju ceļvedis....26
5.1. Instalēšana un palaišana....26
5.2. Galvenā izvēlne un spēles uzsākšana....26
5.3. Spēles gaita un mērķis....29
5.4. Spēles saglabāšana un turpināšana....30
5.5. Lietotāja interfeiss un inventāra sistēma....31
5.6. Vadība un mijiedarbība ar spēles pasauli....31
5.7. Ieeja alā....32
5.8. Nakts iestāšanās....32
5.9. Spēles beigas....33
6. Testēšanas dokumentācija....34
6.1. Izvēlētās testēšanas metodes, rīku apraksts un pamatojums....34
6.2. Alternatīvās testēšanas metodes un rīki....34

6.3. Testpiemēru kopa.....35
6.4. Testēšanas žurnāls.....37
6.5. Testēšanas rezultāts.....39
Secinājumi.....40
7. Lietoto saīsinājumu un terminu skaidrojums.....41
Literatūras un informācijas avoti.....42
Pielikumi.....43

Ievads

Šajā dokumentā darba autors apraksta spēles "Survive Protocol" izstrādes procesu un tās struktūru. Visas spēlē ieviestās funkcijas ir detalizēti aprakstītas, pārbaudītas un dokumentētas, kā arī papildinātas ar diagrammām, lai nodrošinātu pilnīgu izpratni par spēles uzbūvi. Dokumentā ir iekļauti arī spēles ekrānattēli un koda fragmenti, kas sniedz vizuālu ieskatu projekta realizācijā.

Šī spēle ir veidota kā izaicinājums, kur spēlētāja galvenais mērķis ir izdzīvot pēc iespējas vairāk naktis un noturēties spēlē pēc iespējas ilgāk. Spēlētājam nokļūstot spēles pasaulē, sākas rīts, viņam nav nekādu resursu un viņš ir pilnīgi viens. Ir piešķirts neliels laika periods, lai adaptētos jaunajai videi, pirms iestājas nakts.

Nakts laikā spēlētāju sāks apdraudēt ienaidnieki — tā saucamie mobi, kas katru nākamo nakti kļūs gan spēcīgāki, gan skaitliski vairāk. Spēlētājam būs iespēja sevi aizsargāt, attīstīties un uzlabot savas spējas. Viņš varēs iegūt resursus un doties alās, lai izpētītu apkārtni un vāktu nepieciešamos materiālus. Iegūtie resursi ļaus uzlabot ekipējumu un palielināt izdzīvošanas iespējas.

Spēles galvenā iezīme ir pilnībā nejauši ģenerēta pasaule — katra spēles sesija tiek veidota no jauna, un katra pasaule ir unikāla. Šī pasaules ģenerācija balstās uz trokšņu algoritmiem, kas nodrošina dabisku un neatkārtojamu vidi. Tas ne tikai rada izdzīvošanas spriedzi, bet arī piedāvā spēlētājam izpētes un piedzīvojumu sajūtu.

Šī spēle piedāvā izaicinājumu — izdzīvot pēc iespējas ilgāk. Visinteresantākā tās sastāvdaļa ir konkurences elements, kur spēlētāji var salīdzināt savus rezultātus, sacenšoties savā starpā — kurš spēj ātrāk pielāgoties mainīgajiem apstākļiem un domāt stratēģiski. Spēle ir īpaši piemērota jauniešiem, kuri vēlas aizraujoši un jēgpilni pavadīt savu brīvo laiku. Tā vēl jo vairāk iegūst vērtību, ja tiek spēlēta kopā ar draugiem, radot kopīgu pieredzi un veselīgu sacensības garu.

Darba autors izvēlējās spēļu dzinēju Godot Engine savas 2D spēles izstrādei. Šis dzinējs ir salīdzinoši viegli apgūstams, jo programmēšana notiek valodā GDScript, kas sintaktiski līdzinās Python un ir draudzīga iesācējiem. Turklāt Godot ir tehniski viegls un kompakts — to iespējams palaist pat no USB zibatmiņas un tas darbojas arī uz datoriem ar zemākiem tehniskajiem parametriem. Tas nozīmē, ka cilvēki ar ierobežotiem resursiem var ne tikai spēlēt, bet arī paši izstrādāt spēles šajā vidē.

Spēles pirmkods un projekta materiāli ir pieejami GitHub repozitorijā: https://github.com/Malekitos/survive-protocol, kur iespējams iepazīties ar pilnu izstrādes gaitu un tehnisko realizāciju.

1. 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 un būvniecības sistēmas ieviešanu. 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.

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.

2. 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.

2.1. Produkta perspektīva

Šī spēle ir unikāla galvenokārt ar to, ka katru reizi, izveidojot jaunu pasauli, tā tiek ģenerēta no jauna. Tas spēlētājam sniedz pavisam jaunu pieredzi, jo pasaule katru reizi ir atšķirīga. Arī alas tiek ģenerētas no jauna, tādējādi nav iespējams, ka spēlētājs atkārtoti spēlētu vienā un tajā pašā pasaulē.

Resursu daudzums pasaulē ir ierobežots, tādēļ spēlētājam jāizstrādā stratēģija — vai nu pievērsties aizsardzībai, izgatavojot vairāk bruņu un stiprinot sienas, vai arī izvēlēties uzbrukuma taktiku, koncentrējoties uz zobenu izgatavošanu, lai nodarītu lielāku kaitējumu. Taču šādā gadījumā katra spēlētāja kļūda var rezultēties ar zaudējumu.

Izdzīvošanas izaicinājumi ar nakts režīmu pievieno papildu aspektu, kur spēlētājiem ir jāadaptējas un jāaizsargājas pret mobiem.

Šī spēle atšķiras no citām ar to, ka daudzas mehāniku idejas ir aizgūtas no citām spēlēm. Šis projekts ir dažādu mehāniku un ideju apvienojums no spēlēm, kas pašam projekta autoram ir iecienītas.

2.2. 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:

- Pēc pogas nospiešanas pārvietot lietotāju uz citu spēles ainu, ielādējot attiecīgo scēnu.
- Izvaddati:
- Parādīt galvenā ekrāna fona attēlu.

- Izveidot un parādīt opcijas: “New Game”, “Load Game”, “Settings”, “Records”, “Instruction”, “Exit”, .

PR.2. Spēles turpināšana

Mērķis:

- Ļaut spēlētājam atsākt iepriekšējo spēles sesiju.

Ievaddati:

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

Apstrāde:

- Izgūt un apstrādāt failis ar saglabātajām spēles pasauli no failiem.
- Ielādēt izvēlēto spēles pasauli un tās datus, piemēram, spēlētāja stāvokli, resursus un pasauli.

Izvaddati:

- Ielādēt saglabāto spēles pasauli no faila.
- Pārslēgties uz attiecīgo scēnu.
- Attēlot spēles pasauli ar tās elementiem.

PR.3. Iestatījumi

Mērķis:

- Ļaut spēlētājam pielāgot spēles iestatījumus, lai uzlabotu pieredzi.

Ievaddati:

- 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. Rekordu izvēlnes atvēršana

Mērķis:

- Nodrošināt piekļuvi spēlētāja rekordu sarakstam, kas saglabāts lokālā failā.

Ievaddati:

- Spēlētāja darbība — pogas "Records" nospiešana galvenajā izvēlnē.

Apstrāde:

- Atvērt rekordu izvēlni.
- Nolasīt rekordu datus no JSON faila.

Izvaddati:

- Attēlot pirmos piecus rekordu ierakstus, sakārtotus pēc spēlē pavadītā laika.
- Parādīt pasaules nosaukumu un attiecīgo laiku katram ierakstam.

PR.6. Instrukciju izvēlnes atvēršana

Mērķis:

- Nodrošināt lietotājam piekļuvi spēles instrukcijām, lai izprastu spēles pamatmehānikas.

Ievaddati:

- Spēlētāja darbība — pogas "Instrukcijas" nospiešana galvenajā izvēlnē.
- Atvērt instrukciju izvēlni.
- Ielādēt attēlus un aprakstošo tekstu no resursu mapes vai failiem.

Izvaddati:

- Attēlot instrukciju ekrānu ar fotogrāfijām un skaidrojošu tekstu par spēles darbību.

PR.7. 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.8. Resursu ieguve

Mērķis:

- Nodrošināt spēlētājam iespēju iegūt resursus no spēles vides, mijiedarbojoties ar tiem.

Ievaddati:

- Spēlētāja klikšķis uz resursa objekta.
- Spēlētāja esošais aprīkojums.

Apstrāde:

- Noteikt, vai spēlētājam ir piemērots rīks.
- Aprēķināt nepieciešamo klikšķu skaitu, ņemot vērā rīka efektivitāti.
- Pakāpeniski samazināt resursa izturību katra klikšķa laikā.

Izvaddati:

- Kad resurss ir pilnībā izstrādāts, tas pazūd no vides.
- Iegūtais resurss tiek pievienots spēlētāja inventāram.

PR.9. Inventāra atvēršana

Mērķis:

- Ļaut spēlētājam atvērt inventāra saskarni.

Ievaddati:

- Spēlētāja tastatūras ievade (TAB 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.10. Inventāra saglabāšana

Mērķis:

- Nodrošināt, ka spēlētāja inventāra saturs tiek saglabāts starp spēles sesijām.

Ievaddati:

- Spēlētāja darbība — izeja no spēles pasaules vai spēles aizvēršana.
- Pašreizējais inventāra stāvoklis (priekšmetu saraksts).

Apstrāde:

- Izveidot vai atjaunināt **Resource** tipa failu.
- Saglabāt masīvu ar visiem inventārā esošajiem priekšmetiem atsevišķā failā.

Izvaddati:

- Fiziski saglabāts fails ar inventāra datiem, kas tiek izmantots nākamajā ielādē.

PR.11. Inventāra ielāde no saglabātā faila

Mērķis:

- Nodrošināt spēlētājam piekļuvi iepriekšējam inventāram, turpinot spēli no saglabātā stāvokļa.

Ievaddati:

- Spēlētāja darbība — spēles pasaules ielāde no saglabātā stāvokļa.
- Inventāra dati no saglabātā **Resource** faila.

Apstrāde:

- Nolasīt inventāra masīvu no atbilstošā faila.
- Atjaunot inventāru spēlē, pievienojot katru priekšmetu tā sākotnējā pozīcijā.

Izvaddati:

- Spēlētāja inventārs spēlē tiek ielādēts ar visiem iepriekš saglabātajiem priekšmetiem.

PR.12. Inventāra pārvaldība

Mērķis:

- Nodrošināt spēlētājam iespēju ērti pārvaldīt priekšmetus inventārā, izmantojot velkšanas un nomešanas (drag and drop) funkcionalitāti.

Ievaddati:

- Spēlētāja klikšķis uz priekšmeta inventārā.
- Pele tiek pārvietota uz citu inventāra lauciņu.
- Peles pogas atlaišana uz mērķa lauciņa.

Apstrāde:

- Aktivizēt drag and drop režīmu pēc klikšķa uz priekšmeta.
- Pārbaudīt, vai mērķa lauciņš ir tukšs vai jau aizņemts.
- Ja lauciņš ir tukšs — pārvietot priekšmetu uz jauno vietu.
- Ja lauciņš ir aizņemts — samainīt abu priekšmetu vietas.

Izvaddati:

- Atjaunināts inventārs ar priekšmetu jauno pozīciju.
- Vizuāla atgriezeniskā saite par priekšmeta pārvietošanu vai apmaiņu.

PR.13. Komponentu izgatavošana

Mērķis:

- Ļaut spēlētājam izgatavot komponentus no inventārā esošajiem resursiem.

Ievaddati:

- Spēlētāja ievade (klikšķis uz vēlamā komponenta receptes inventārā).

Apstrāde:

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

Izvaddati:

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

PR.14. 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.15. Ienaidnieku parādīšanās naktī

Mērķis:

- Palielināt spēles izaicinājumu, palielinot ienaidnieku mobus skaitu katru naktinakti.

Ievaddati:

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

Apstrāde:

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

Izvaddati:

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

PR.16. Mobu pārvietošanā

Mērķis:

- Piešķirt mobiem spēju pārvietoties spēles pasaulē ar mērķi tuvoties spēlētājam.

Ievaddati:

- Mobu atrašanās vieta pēc parādīšanās 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 spēlētāja virzienā, mainot 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.17. 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, noteiktā rādiusā), sākt uzbrukuma mehāniku.
- Katrs mobs, kas ir pietiekami tuvu, 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.18. 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.19. 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, parādās animācija vai efekts, kas parāda nāvi.

PR.20. Pasaules ģenerācija

Mērķis:

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

Ievaddati:

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

Apstrāde:

- Izmantot trokšņu algoritmu (Perlin noise) zemes un ūdens ģenerācijai, kas nosaka reljefa augstumu un ūdens līmeņus.
- Otrs trokšņa 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 trokšņa 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 trokšņiem.
- 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.21. Pazemes alu ģenerācija

Mērķis:

- Izveidot alas ar resursiem un izejām, izmantojot procedurālo ģenerāciju.

Ievaddati:

- Ģenerācijas sēkla vai nejaušības faktors.

Apstrāde:

- Alas tiek ģenerētas ar **Perlin noise** algoritmu dažādos dziļuma līmeņos.
- Katras alas iekšienē tiek novietoti resursi (piemēram, akmens, metāli).
- Daļā alu automātiski veidojas izejas uz citiem līmeņiem vai virszemi.
- Ģenerācija notiek pasaules izveides brīdī vai spēlētājam tuvojoties teritorijai.

Izvaddati:

- Pazemē izveidojas alas ar resursiem un savienojumiem ar citiem apgabaliem.

PR.22. Priekšmeta izmešana ar taustiņu G

Mērķis:

- Ļaut spēlētājam ātri izmest aktīvo priekšmetu no HotBar.

Ievaddati:

- Spēlētāja tastatūras ievade (taustiņš **G**).
- Aktīvais HotBar slots ar priekšmetu.

Apstrāde:

- Nospiežot taustiņu **G**, spēles pasaulē tiek izveidots objekts — izmestais priekšmets.
- Tas tiek izņemts no spēlētāja inventāra.

Izvaddati:

- Uz zemes parādās priekšmets, ko spēlētājs nometis.
- HotBar tiek atjaunināts, izņemot izmesto priekšmetu.

PR.23. Aktīvā HotBar slota maiņa

Mērķis:

- Nodrošināt spēlētājam iespēju pārslēgt aktīvo slotu HotBar joslā.

Ievaddati:

- Tastatūras ievade (taustiņi no 1 līdz 9) vai peles rullīša ritināšana.

Apstrāde:

- Spēles apakšdaļā vienmēr tiek attēlota **HotBar** josla ar 9 šūnām.
- Pēc spēles starta aktīvs ir pirmais slots, kas tiek vizuāli iezīmēts (piemēram, ar nelielu ēnojumu).
- Ritinot peles rullīti uz augšu vai leju, vai nospiežot taustiņus no 1 līdz 9, spēlētājs var pārslēgt aktīvo slotu.

Izvaddati:

- HotBar vizuāli atjaunojas, parādot, kurš slots ir aktīvs.
- Spēles darbības (piemēram, priekšmetu izmantošana vai novietošana) tiek veiktas ar aktīvajā slotā esošo priekšmetu.

2.3. Sistēmas nefunkcionālās prasības

NR.1. Veiktspēja

- Spēlei ir jānodrošina vienmērīgs kadru ātrums (FPS) — vismaz 60 kadri sekundē, izmantojot vidējās klases aparatūru. Tā kā spēle ir izstrādāta 2D vidē, pietiek ar šādiem sistēmas resursiem:
- **Procesors:** divkodolu (dual-core) CPU ar taktsfrekvenci vismaz 2.0 GHz
- **Operatīvā atmiņa:** vismaz 4 GB RAM
- **Grafiskā karte:** integrēta grafika ar OpenGL 3.0 atbalstu (piemēram, Intel HD Graphics)
- **Operētājsistēma:** Windows 10 / Linux / macOS

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.

2.4. 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.

Šīs spēles spēlētāji ir mērķtiecīgi cilvēki, kuri ir gatavi trenēties un uzstādīt jaunus rekordus. Viņi ir metodiski un centīgi, lai pilnībā izprastu katru spēles mehāniku.

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.

3. Izstrādes līdzekļu, rīku apraksts un izvēles pamatojum

Šajā nodaļā darba autors apraksta programmēšanas rīkus, kurus izvēlējās kvalifikācijas darba izstrādei, aprakstot, ko un kāpēc izvēlējās, kā arī ko 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.

3.1. 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. Šī 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.

3.2. Iespējamo (alternatīvo) risinājuma līdzekļu un valodu apraksts

Unity

Šis spēļu dzinējs netika izvēlēts, pamatojoties uz vairākiem kritērijiem. Pirmkārt, šī 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 lietojumprogrammu 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.

4. Sistēmas modelēšana un projektēšana

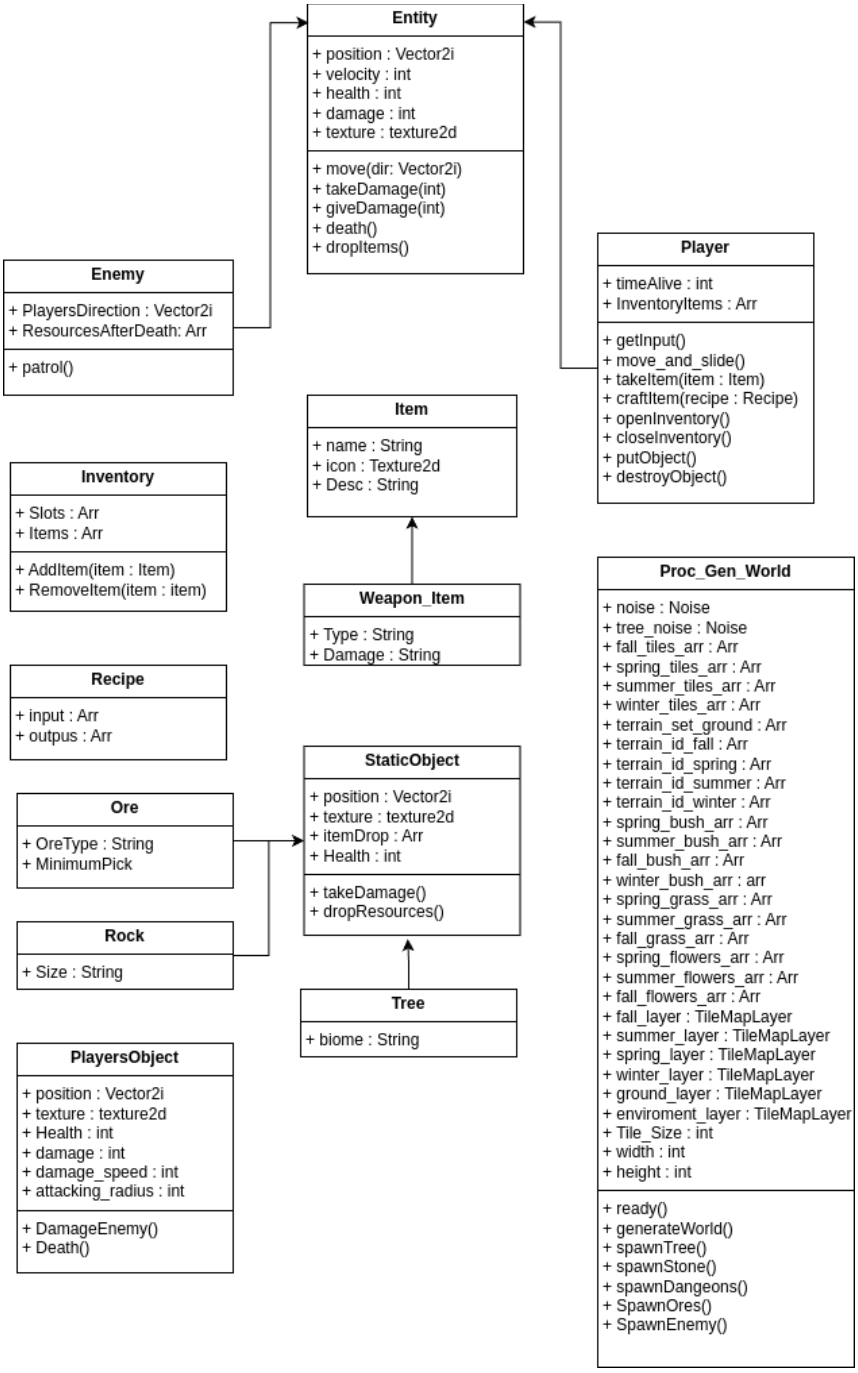
Šajā sadaļā tiks iekļautas dažādas diagrammas, kas vizuāli ataino izstrādātās spēles darbības loģiku un tās galvenos procesus. Daži elementi šajās diagrammās apzināti ir izveidoti angļu valodā. Spēles pirmkods būs atvērts, lai ikviens, kurš vēlēsies to modernizēt vai pārveidot, varētu vieglāk izprast, kā šī spēle darbojas. Šajā spēlē netiek izmantota datubāze. Visi dati, kas tiek glabāti, tiek saglabāti kā **Resource**, **PackedScene** vai **JSON** formātā. Tie ir atsevišķi faili, kas tiek fiziski glabāti projekta direktorijā.

4.1. 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 ieguves un priekšnmetu veidošanas sistēmas. Diagramma palīdz saprast, kā dažādas sistēmas daļas sadarbojas savā starpā, nodrošinot vienotu un funkcionējošu spēles vidi.

4.1.1. 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.

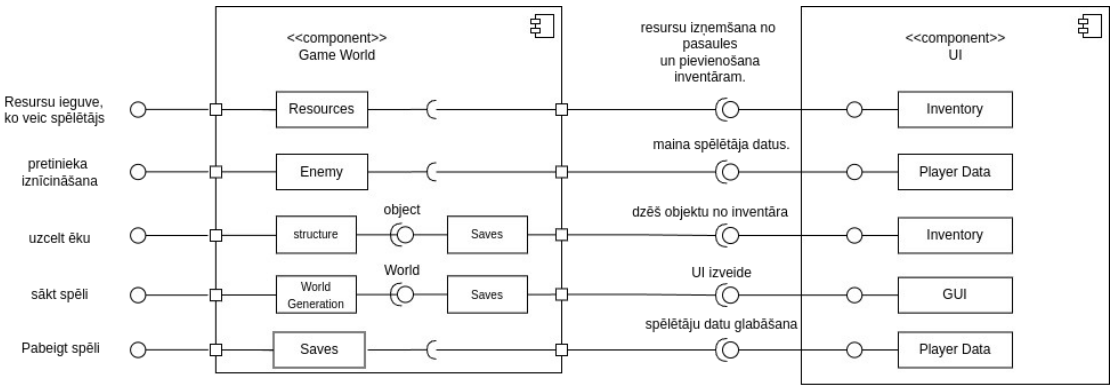


1. attēls. klašu diagramma

4.1.2. 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.



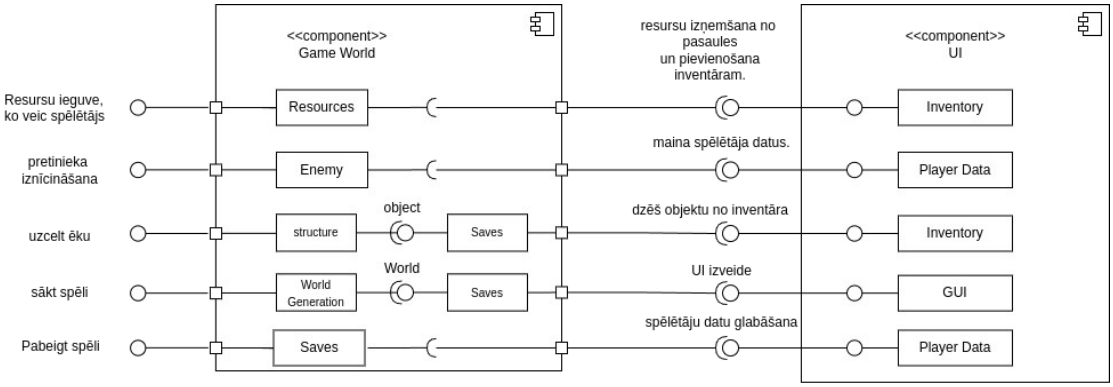
2. attēls. komponentu diagramma

4.2. 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.

4.2.1. Aktivitāšu diagramma

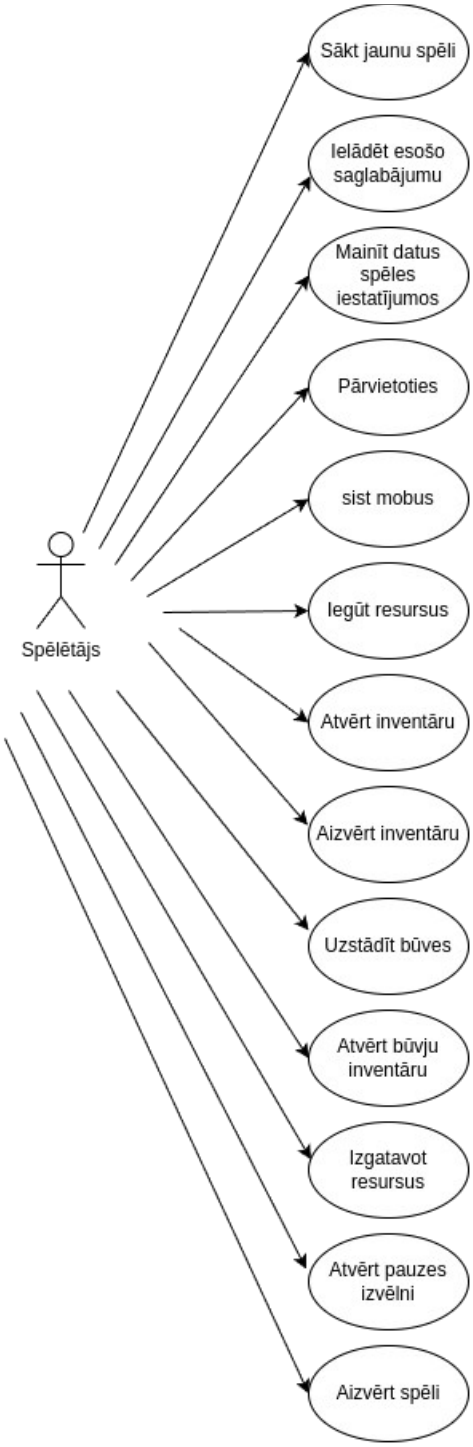
Šajā aktivitāšu diagrammā attēlots spēles lietotāja darbību secīgums, sākot no galvenās izvēlnes līdz aktīvai spēles norisei. Diagramma parāda galvenos procesus, piemēram, pasaules ģenerēšanu, spēlētāja pārvietošanos, mijiedarbību ar objektiem, resursu vākšanu, priekšmetu veidošanu un cīņu ar pretiniekiem.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



3. attēls. Aktivitāšu diagramma

4.2.2. Lietojumgadījumu diagramma

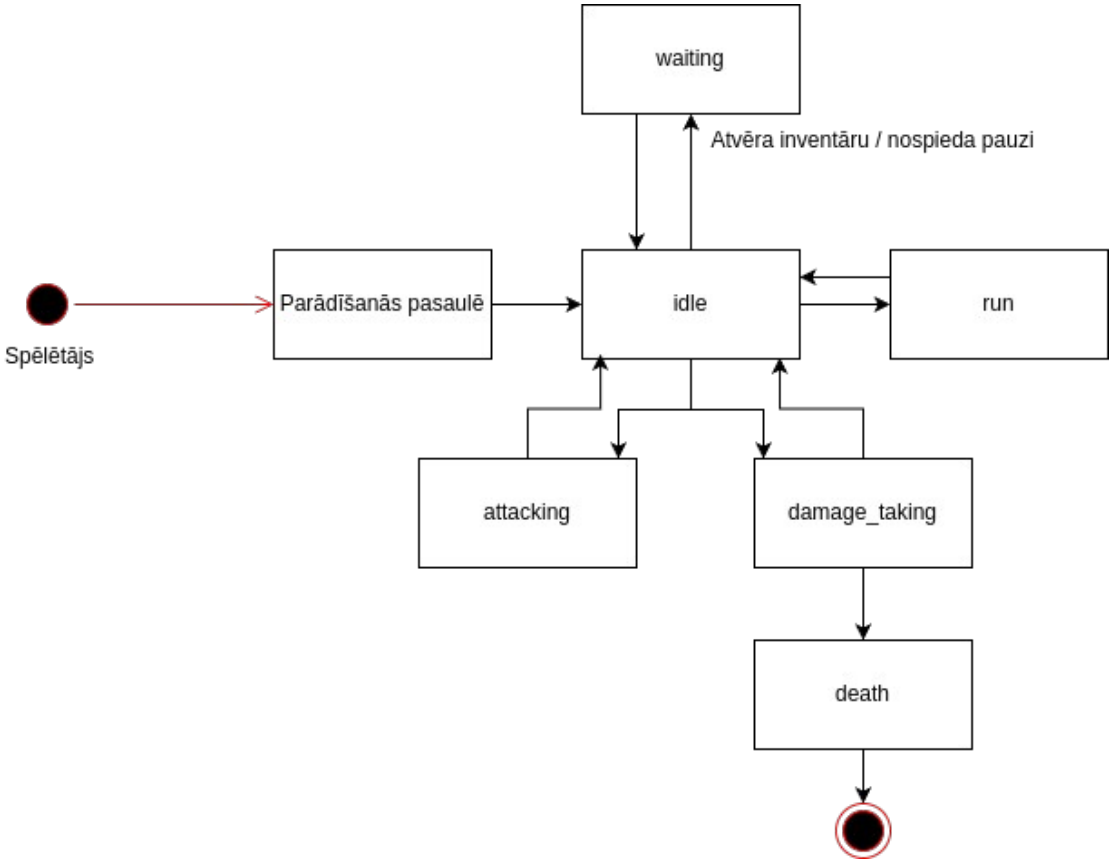
Lietojumgadījumu diagramma atspoguļo galvenās darbības, ko spēlētājs var veikt spēles laikā. Tajā iekļauti tādi scenāriji kā jaunas spēles uzsākšana, saglabājumu ielāde, pārvietošanās, resursu iegūšana, cīņa ar pretiniekiem, priekšmetu veidošana, būvju uzstādīšana un izvēlņu izmantošana. Diagramma palīdz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



4. attēls. Lietojumgadījumu diagramma

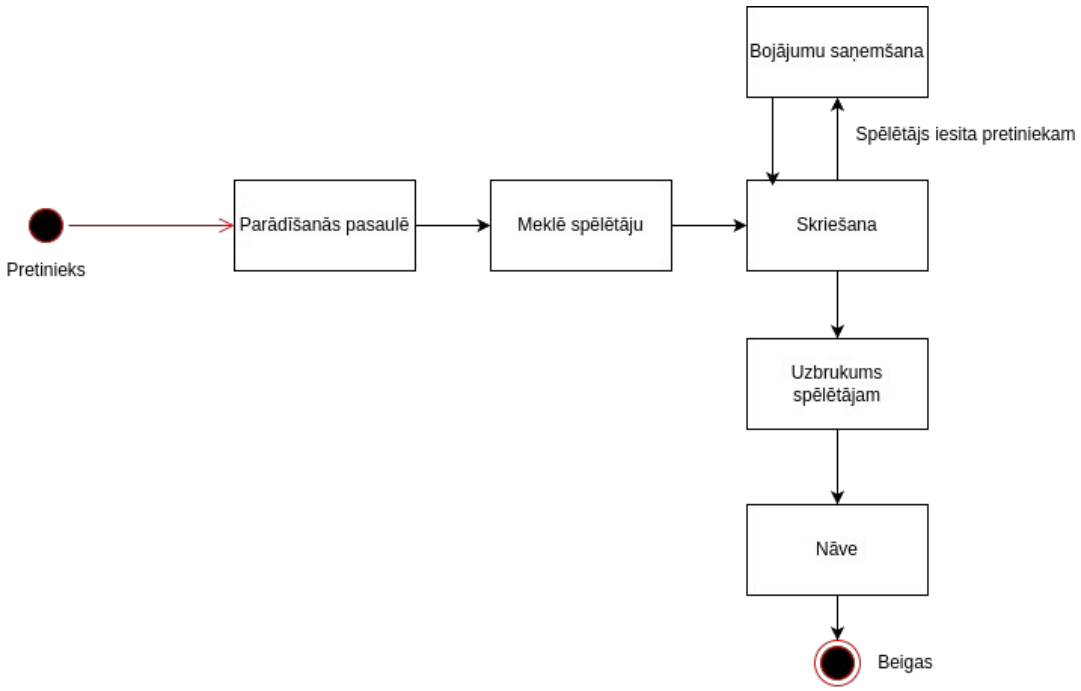
4.2.3. Stāvokļu diagramma

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

4.3. 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. Tos var izmantot arī kā spēlētāja datu saglabāšanas veidu, piemēram, veselības stāvoklim vai priekšmetiem, kas atrodas inventārā. Visa informācija tiek saglabāta failā ar paplašinājumu .tres.

PackedScene Godot dzinējā apzīmē ainas iepakotu versiju. Aina ir vieta, kurā norisinās spēles process – katrs nozīmīgāks spēles elements ir veidots kā atsevišķa aina. Piemēram, inventārs ir atsevišķa aina, spēlētājs ir atsevišķa aina, un galvenā aina satur vairākas šādas apakšainas. PackedScene ir iepakota ainas versija, kas tiek saglabāta

.tres formātā. Jebkuru ainu, piemēram, spēles pasaules ainu, var iepakot kā **PackedScene** un saglabāt failā. Vēlāk šo failu iespējams ielādēt un atpakot, tādējādi atjaunojot spēles pasauli. Tieši šādā veidā notiek spēles pasaules saglabāšana un ielāde.

Tiek izmantots arī **JSON** fails rekordu saglabāšanai — tajā tiek ierakstīts pasaules nosaukums un laiks, ko spēlētājs tajā pavadījis. Šī informācija ļauj vēlāk datus kārtot un attēlot spēlētāju reitingu.

5. Lietotāju ceļvedis

Šī nodaļa sniedz detalizētu pārskatu par spēles **"Survive Protocol"** lietošanas aspektiem, sākot no instalēšanas un palaišanas iespējām līdz pat spēles vadības un interfeisa izmantošanai. Aprakstīti divi galvenie spēles palaišanas veidi – caur izpildāmo failu vai izmantojot Godot Engine – atbilstoši lietotāja vajadzībām un tehniskajām prasmēm. Tiek izskaidrota spēles gaita, diennakts cikla ietekme, mērķi un izdzīvošanas mehānika. Nodaļā apskatīta arī galvenā izvēlne, spēles saglabāšanas sistēma, spēlētāja inventārs un tā pārvaldība, kā arī lietotājam draudzīgā vadība, kas nodrošina intuitīvu mijiedarbību ar spēles pasauli. Ceļvedis kalpo kā praktisks norādījums spēlētājiem un interesentiem, kuri vēlas izprast spēles funkcionalitāti un pilnvērtīgi izmantot tās iespējas.

5.1. Instalēšana un palaišana

Spēle **"Survive Protocol"** ir izstrādāta, izmantojot **Godot Engine 4**, kas nodrošina iespēju palaist spēli uz dažādām operētājsistēmām, tostarp **Windows**, **macOS** un **Linux**.

Lietotājam ir divas iespējas, kā palaist spēli:

1. **Izmantojot izpildāmo failu**

   Lietotājs var palaist spēli tieši, atverot failu Survive-Protocol.exe, kas atrodas spēles **Bin** mapē. Šī metode ir vienkāršākā un paredzēta lietotājiem, kuri nevēlas strādāt ar spēles iekšējo struktūru.

2. **Izmantojot Godot Engine**

   Lietotājs var lejupielādēt spēles projektu no GitHub "github.com/Malekitos/survive-protocol", importēt to **Godot Engine 4** vidē, pārskatīt projekta avota failus un palaist spēli tieši no redaktora. Šī metode ir īpaši noderīga izstrādātājiem vai interesentiem, kuri vēlas iepazīties ar spēles kodu un struktūru.

5.2. Galvenā izvēlne un spēles uzsākšana

Pēc spēles palaišanas tiek attēlots galvenais izvēlnes ekrāns ar vairākām pamatfunkcijām. Lietotājs var sākt jaunu spēli, ielādēt iepriekš saglabātu progresu, pielāgot iestatījumus vai iziet no spēles. Iestatījumu sadaļā iespējams regulēt skaņas skaļumu, mainīt ekrāna izšķirtspēju un pārslēgt pilnekrāna režīmu.

Ja spēle tiek atvērta pirmo reizi, lietotājam vispirms ir jāizveido jauna pasaule. Šim nolūkam ir nepieciešams izvēlēties iespēju izveidot jaunu pasauli, ievadīt tās nosaukumu un apstiprināt. Pēc šī soļa spēlētājs automātiski nonāk spēles vidē un var sākt piedzīvojumu.

1. attēls. Galvenā izvēlne

2. attēls. Pasaules izveidošana

Nospiežot pogu **"Records"**, tiek attēloti spēlētāju rekordi. Tiek parādīti tikai pirmie pieci rezultāti, kas ir sakārtoti pēc spēlē pavadītā laika. Katram ierakstam iespējams redzēt, cik ilgu laiku spēlētājs ir pavadījis konkrētajā pasaulē, kā arī attiecīgās pasaules nosaukumu.

Ja pasaules izveides brīdī spēlētājs nav norādījis nosaukumu, tad pēc noklusējuma tiek izmantots nosaukums **"World: Undefined"**.

3. attēls. Rekodri

Šeit lietotājs var mainīt spēles skaņas efektu un mūzikas skaļumu. Šie iestatījumi tiek saglabāti failā, tādējādi nākamajā spēles palaišanas reizē tie automātiski tiek ielādēti, un lietotājam nav nepieciešams katru reizi tos iestatīt no jauna.

4. attēls. Iestatījumi

Ja mēs ieejam instrukciju izvēlnē, mēs redzēsim daudz attēlu, blakus kuriem ir teksts. Šeit ir aprakstīts, kā spēlēt spēli, galvenais mērķis un viss, kas tajā jādara.

5. attēls. Instrukcija

5.3. Spēles gaita un mērķis

Kad spēlētājs pirmo reizi nonāk spēles pasaulē, viņš tiek ievietots vidē, kurā valda diennakts cikls. Spēle sākas ar saullēktu, un dienas laikā spēlētājam ir jāizpēta apkārtne, jāiegūst nepieciešamie resursi, jāveido instrumenti, jāiegūst bruņas un jāgatavojas nakts pārbaudījumiem. Resursu vākšana un amatniecības sistēma ļauj spēlētājam izstrādāt arvien efektīvāku aprīkojumu, kas ir būtiski izdzīvošanas nodrošināšanai.

Karte, kurā norisinās spēle, tiek ģenerēta procedurāli, nodrošinot unikālu pieredzi katrā jaunā spēlē. Pēc noteikta laika spēlē iestājas nakts, un tieši tad uz kartes sāk parādīties dažādi monstri. Spēlētāja uzdevums ir izdzīvot šo nakti, aktīvi aizstāvoties un izmantojot iepriekš sagatavoto ekipējumu.

Šajā attēlā apakšējā daļā redzams **Hotbar** (ātrās piekļuves rīku josla). Tas sniedz spēlētājam daudz svarīgas informācijas.

Pirmkārt, zilā josla norāda bruņu līmeni — jo vairāk bruņu tiek izgatavots, jo vairāk šī josla piepildās. Zem tās atrodas zaļā josla, kas attēlo spēlētāja veselības līmeni — katru reizi, saņemot bojājumus, šī josla samazinās.

Zemāk ir redzami spēlētāja izvēlētie priekšmeti **Hotbar** joslā. Ja kāda priekšmeta ikona ir nedaudz pelēkāka nekā pārējās, tas nozīmē, ka šis priekšmets ir pašlaik izvēlēts. Nospiežot taustiņu **G**, izvēlētais priekšmets tiks izmests.

5. attēls. HotBar

Kad nakts beidzas, iestājas īss miera periods, kura laikā spēlētājs var atgūt spēkus, salabot aprīkojumu un sagatavoties nākamajam izaicinājumam. Šāds cikls – diena, nakts, izdzīvošana – atkārtojas līdz brīdim, kad spēlētājs zaudē dzīvību. Spēles galvenais mērķis ir izdzīvot pēc iespējas ilgāk un uzstādīt jaunu personīgo rekordu.

5.4. Spēles saglabāšana un turpināšana

Lietotājam ir iespēja jebkurā brīdī pārtraukt spēli un atgriezties galvenajā izvēlnē, nezaudējot progresu. Spēles pasaules un spēlētāja statuss tiek automātiski saglabāts, tāpēc nav nepieciešamības manuāli veikt saglabāšanu pirms iziešanas no spēles. Šī funkcionalitāte nodrošina ērtu spēles pieredzi un ļauj lietotājam bez raizēm pārtraukt spēli, piemēram, ja nepieciešams doties prom no datora.

Atkārtoti palaižot spēli un izvēloties opciju “Ielādēt saglabājumu”, lietotājam tiek parādīts to pasaulju saraksts, kurās viņš ir spēlējis. Izvēloties kādu no tām, spēle tiek ielādēta tieši no pēdējā automātiski saglabātā brīža, ļaujot spēlētājam turpināt piedzīvojumu tieši no vietas, kur tas tika pārtraukts.

5.5. Lietotāja interfeiss un inventāra sistēma

Spēles laikā lietotājam ir pieejams inventārs, kurā jebkurā brīdī var pārskatīt savāktos resursus un priekšmetus. Inventāra sistēma ir būtiska spēles sastāvdaļa, jo tā ļauj efektīvi pārvaldīt krājumus, plānot darbības un veikt amatniecību.

Spēlētājs var atvērt inventāru, lai apskatītu visus pieejamos resursus, piemēram, ieguves materiālus, rīkus, vai izgatavotus priekšmetus. No šīs pašas izvēlnes ir iespējams izgatavot jaunus rīkus vai ekipējumu, ja spēlētāja rīcībā ir nepieciešamie materiāli. Amatniecības process notiek intuitīvā un lietotājam draudzīgā veidā.

Tāpat spēlētājam ir iespēja izdzēst jebkuru nevajadzīgu priekšmetu no inventāra. Šī funkcija palīdz uzturēt kārtību un atbrīvot vietu jauniem resursiem. Inventārs kopumā kalpo kā galvenais spēles pārvaldības instruments, kas ļauj spēlētājam pielāgoties situācijām un efektīvi reaģēt uz izdzīvošanas izaicinājumiem.

6. attēls. Inventāra pārvaldība un iespēja izgatavot priekšmetus

5.6. Vadība un mijiedarbība ar spēles pasauli

Spēles vadība ir intuitīva un balstīta uz standarta shēmu, kas pazīstama lielākajai daļai datoru lietotāju. Spēlētāja pārvietošanās pa pasauli tiek veikta, izmantojot tastatūras taustiņus **W, A, S, D**, kas ļauj virzīties uz priekšu, atpakaļ un uz sāniem. Šī vadības metode nodrošina vienmērīgu un precīzu kustību, kas ir būtiska gan resursu vākšanā, gan cīņās ar pretiniekiem.

Visas citas darbības spēles pasaulē, piemēram, resursu ieguve vai cīņa ar monstriem, tiek veiktas, izmantojot **kreisās peles pogas klikšķi**. Spēlētājam ir

jāklikšķina uz objekta vai mērķa vairākkārt, lai veiktu attiecīgo darbību – piemēram, lai uzvarētu pretinieku.

Papildu tam, **inventārs tiek atvērts ar taustiņu TAB**, kas ļauj spēlētājam jebkurā brīdī piekļūt saviem resursiem, izgatavot priekšmetus vai pārvaldīt krājumus. Šī funkcionalitāte nodrošina ātru un ērtu piekļuvi visam nepieciešamajam izdzīvošanai.

5.7. Ieeja alā

Kartē ir redzamas vairākas alas, un, ieejot kādā no tām, spēlētājs tiek pārnests uz atsevišķu alas pasauli, kas katru reizi tiek ģenerēta no jauna. Šajās alās spēlētājs var iegūt dažādus resursus.

7. attēls. Ala

5.8. Nakts iestāšanās

Ik pēc noteikta laika spēlē iestājas nakts, un sāk parādīties ienaidnieki (mobi). Attēlā var redzēt nakts brīdi un to, kā ir parādījies mobs.

8. attēls. Nakts un Mobs

5.9. Spēles beigas

Saņemot bojājumus no mobiem, spēlētājam samazinās veselības līmenis. Kad veselība sasniedz nulli, spēlētājs mirst un spēle beidzas. Šajā attēlā redzama nāves izvēlne, kurā parādās pasaules nosaukums un laiks, cik ilgi spēlētājs tajā ir izdzīvojis.

9. attēls. Nāves izvēlne

6. Testēšanas dokumentācija

Šajā sadaļā darba autors apraksta, kā tika testēta izstrādātā spēle "Survive Protocol". Tiks izklāstītas izvēlētās manuālās testēšanas metodes, izmantotie rīki un to izvēles pamatojums. Galvenais mērķis bija pārbaudīt spēles funkcionalitāti, identificēt kļūdas un novērtēt lietošanas ērtumu.

6.1. Izvēlētās testēšanas metodes, rīku apraksts un pamatojums

Izstrādātās spēles "Survive Protocol" testēšanai tika izvēlēta **Black Box** testēšanas metode. Šī pieeja ļauj novērtēt spēles funkcionalitāti no lietotāja skatpunkta, nepievēršoties programmas iekšējai struktūrai vai koda darbībai. Tika pārbaudīts, vai spēles elementi – piemēram, vadība, mijiedarbība ar objektiem, līmeņu pārejas un spēles mehānikas – darbojas atbilstoši paredzētajam. Tā kā spēle tiek testēta manuāli, šī metode ir vispiemērotākā, lai identificētu vizuālās, loģiskās un lietojamības kļūdas, kuras varētu ietekmēt lietotāja pieredzi.

Papildus autoram, spēli testēja arī **trešās personas** – citi spēlētāji, kuri sniedza atgriezenisko saiti par spēles gaitu, vadību, kļūdām un kopējo lietošanas pieredzi. Šī neformālā lietotāju testēšana palīdzēja identificēt kļūdas, kuras autors pats nebūtu pamanījis, un deva vērtīgus ieteikumus spēles uzlabošanai.

Spēles testēšana tika veikta uz **Fedora Linux** operētājsistēmas, kas tika izmantota kā galvenā izstrādes un testēšanas vide. Papildu pārbaudei spēle tika testēta arī uz **Windows** operētājsistēmas, izmantojot virtuālo mašīnu, lai pārliecinātos par tās darbību dažādās platformās.

Visi testēšanas rezultāti un novērotās kļūdas tika sākotnēji reģistrēti tabulā, kas izveidota programmā **LibreOffice Calc**. Šāda pieeja ļāva strukturēti un ērti sekot līdzi testēšanas gaitai, kļūdām un to labošanas progresam.

6.2. Alternatīvās testēšanas metodes un rīki

Kā alternatīva izvēlētajai Black Box testēšanai tika apsvērta arī **White Box** testēšanas pieeja. Šī metode ļautu pārbaudīt spēles iekšējo loģiku, tostarp funkciju darbību, nosacījumu izpildi un ciklus, kas īpaši noderīgi būtu, piemēram, **procedurālās pasaules ģenerēšanas vai inventāra sistēmas** pārbaudē. Tomēr, ņemot vērā projekta apjomu un testēšanas mērķi, šī pieeja netika izmantota.

Tāpat tika apsvērta **automātiskās testēšanas** iespēja, izmantojot rīku **Godot UnitTest framework (GUT)**, kas ir īpaši paredzēts Godot Engine lietotājiem. Šis rīks ļauj veidot vienkāršus un atkārtojamus vienību testus, lai automatizēti pārbaudītu noteiktas koda daļas. Neskatoties uz to potenciālajām priekšrocībām, GUT netika izmantots šajā projektā laika ierobežojumu un manuālās testēšanas pietiekamības dēļ.

6.3. Testpiemēru kopa

Šajā tabulā ir apkopoti galvenie testēšanas piemēri, kas izmantoti, lai pārbaudītu spēles "Survive Protocol" funkcionalitāti. Katrs testa piemērs ietver īsu aprakstu, mērķi, ievaddatus, testēšanas soļus un sagaidāmo rezultātu. Šī kopa kalpo kā pamats testēšanas procesam, lai nodrošinātu, ka visas būtiskās spēles funkcijas darbojas paredzētajā veidā un lietotāja pieredze ir kvalitatīva.

1. tabula

Testēšanas piemēru kopums

ID	Testa nosaukums	Mērķis	Ievaddati	Testēšanas soļi	Sagaidāmais rezultāts
PR.1	Galvenais izvēlnes ekrāns	Nodrošināt piekļuvi spēles galvenajām funkcijām un iestatījumiem	Tastatūras ievade, peles klikšķis, pasaules nosaukuma ievade	Atvērt spēli, nospiest pogu "Sākt jaunu spēli", ievadīt pasaules nosaukumu, nospiest pogu "Izveidot".	Nospiežot pogas, lietotājs tiek pārvietots uz citu izvēlni, kā arī, uzsākot spēli, spēlētājs tiek pārvietots uz spēles pasauli.
PR.2	Spēles turpināšana	Ielādēt iepriekš saglabātu spēles sesiju	Tastatūras ievade, peles klikšķis, saglabājuma izvēle	Izvēlēties 'Turpināt', izvēlēties saglabāto pasauli	Ielādējas saglabātā spēles pasaule ar pareizu spēlētāja stāvokli
PR.3	Iestatījumi	Pielāgot skaņas līmeņus un pārbaudīt iestatījumu saglabāšanu	Tastatūras vai peles ievade iestatījumu izvēlnē	Atvērt iestatījumu izvēlni	Parādās skaņas līmeņu slīdņi, iespējams tos mainīt
PR.4	Iziet no spēles	Nodrošināt drošu spēles aizvēršanu pēc apstiprinājuma	Peles klikšķis uz “Iziet” un apstiprinājuma pogas	Izvēlēties 'Iziet', apstiprināt	Spēle aizveras pēc apstiprinājuma
PR.5	Spēlētāja pārvietošanās	Pārbaudīt, vai spēlētājs var brīvi pārvietoties pa pasauli	Tastatūras taustiņi: W, A, S, D	Nospiežot W, A, S, D spēlētājs pārvietojas	Spēlētāja pozīcija mainās atbilstoši komandu virzienam
PR.6	Resursu ieguve	Pārbaudīt resursu vākšanu un to	Peles kreisais klikšķis uz resursa	Noklikšķināt uz koka ar peli	Koks tiek iegūts un parādās

		pievienošanu inventāram			inventārā
PR.7	Inventāra atvēršana	Atvērt un vizuāli attēlot inventāru	TAB taustiņš	Nospiežot TAB	Parādās tukšs inventāra logs ar 20 šūnām
PR.8	Inventāra pārvaldība	Pārbaudīt, vai priekšmeti var tikt pārvietoti starp šūnām	Peles klikšķis un vilkšana uz šūnām inventārā	Vilkt priekšmetu no vienas šūnas uz citu	Priekšmeti apmainās vietām
PR.9	Komponentu izgatavošana	Izgatavot priekšmetus no esošajiem resursiem	Peles klikšķis uz komponenta receptes, ja resursi ir	Nospiežot uz receptes, ja ir resursi	Komponents parādās inventārā, resursi samazinās
PR.10	Dienas un nakts cikls	Pārbaudīt automātisku apgaismojuma maiņu spēlē	Spēles iekšējais laiks (automātisks)	Spēlēt 10 minūtes	Mainās apgaismojums starp dienu un nakti ik pēc 5 minūtēm
PR.11	Mobu spavns naktī	Pārbaudīt, vai naktī parādās mobi un to skaits pieaug	Spēles laiks = nakts (automātisks)	Sasniegt nakti spēlē	Spavnojas 1 mobs, katru nakti vairāk
PR.12	Mobu pārvietošanās	Nodrošināt, ka mobi tuvojas spēlētājam	Mobu un spēlētāja pozīcija	Stāvēt redzamības zonā, gaidīt mobus	Mobi tuvojas spēlētājam
PR.13	Mobu uzbrukums	Pārbaudīt, vai mobi nodara kaitējumu, kad ir pietiekami tuvu	Mobu pozīcija spēlētājam tuvu, spēlētāja veselība	Mobiem pietuvoties spēlētājam	Samazinās spēlētāja veselība ik sekundi
PR.14	Spēlētāja nāve	Pārbaudīt nāves ekrāna parādīšanos, kad veselība sasniedz 0	Uzbrukumi no mobiem līdz veselība = 0	Spēlētājs saņem uzbrukumus, līdz veselība sasniedz 0	Parādās nāves ekrāns ar paziņojumu un spēles ilguma informāciju
PR.15	Spēlētāja uzbrukums mobiem	Nodrošināt spēlētājam iespēju uzbrukt mobiem	Peles klikšķis uz mobiem	Klikšķināt uz mobiem ar peli	Mobiem samazinās veselība, tie var nomirt
PR.16	Pasaules ģenerācija	Izveidot pasauli, balstoties uz šuma algoritmu	Pasaules nosaukums, spēles sākšana	Sākt jaunu spēli	Izveidota jauna pasaule ar šuma algoritmiem
PR.17	Pazemes alu ģenerācija	Ģenerēt alas ar resursiem un izejam no alas	Spēlētāja darbības	Pārvietoties pa spēles pasauli, identificēt alas ieeju un pietuvoties tai līdz noteiktam attālumam	Ģenerējas ala ar resursiem un izejam no alas
PR.18	Dinamiskā pasaules ģenerācija	Automātiski ģenerēt jaunu pasauli spēlētājam pārvietojoties	Spēlētāja pozīcija pasaulē	Pārvietoties līdz kartes malai	Ģenerējas jaunas teritorijas
PR.19	Saglabājuma dzēšana	Nodrošināt iespēju izdzēst	Peles klikšķis uz “Dzēst”,	Izvēlēties dzēst saglabājumu,	Saglabājums tiek noņemts

		saglabātu spēli	apstiprinājums	apstiprināt	no saraksta
PR.20	Krājkastes atvēršana	Atvērt krājkastes inventāru	Peles klikšķis uz novietotas krājkastes	Klikšķināt uz novietotas krājkastes	Parādās krājkastes inventārs
PR.21	Krājkastes iznīcināšana	Iznīcināt krājkasti un izmest tās saturu	Peles klikšķi līdz iznīcināta	Uzbrukt krājkastei līdz iznīcināta	Krājkaste pazūd, priekšmeti parādās uz zemes
PR.22	Priekšmeta izmešana	Ļaut spēlētājam izmest priekšmetu no HotBar	Taustiņš G, ja aktīvajā slotā ir priekšmets	Nospiežot G, kamēr aktīvs priekšmets	Priekšmets parādās uz zemes, pazūd no HotBar
PR.23	HotBar slota maiņa	Pārbaudīt slotu maiņu un aktīvā slota atjaunināšanu	Taustiņi 1–9 vai peles rullītis	Ritināt peli vai spiest 1–9	Mainās aktīvais slots un vizuālā iezīme

6.4. Testēšanas žurnāls

Šī tabula atspoguļo testēšanas rezultātus, pamatojoties uz testpiemēriem, kas iekļauti tabulā *Testēšanas piemēru kopa*. Lai taupītu vietu un izvairītos no informācijas dublēšanas, šeit netiek atkārtoti lauki kā *Testa apraksts*, *Mērķis*, *Ievaddati*, *Testēšanas soļi* un *Sagaidāmais rezultāts*. Tā vietā tiek sniegta atsauce uz konkrēto testa ID, reālais rezultāts un testa statuss, kā arī papildus komentāri, ja nepieciešams.

2. tabula

Testēšanas žurnāls

ID	Reālais rezultāts	Statuss	Komentāri / Piezīmes	Testēšanas datums	Testētājs
PR.6	Pēc koka iegūšanas tas neparādījās inventārā	nesekmīgs	Inventārs atveras, bet tas ir pilnīgi tukšs, lai gan koks tika iegūts	16.05.2024	A. Sidorčenko
PR.2	Spēlētājs parādās pasaulē tajā pašā vietā, kur iepriekš izietajā reizē, taču inventārs ir tukšs	nesekmīgs	Problēma ar inventāru - kādā brīdī tas salūza un vairs vispār nerāda, kas tajā atrodas	16.05.2024	Dž. Svilis-Sudints
PR.12	Mobi dodas spēlētāja virzienā, bet iestrēgst kokos	nesekmīgs	Nepieciešams panākt, lai mobi apiet šķēršļus un uzlabot to mākslīgo intelektu	16.05.2024	Dž. Svilis-Sudints
PR.17	Ģenerējas ala ar resursiem bet bez izejam	nesekmīgs	Spēlētājs nonāk alā, taču tajā nav izeju,	16.05.2024	Dž. Svilis-Sudints

			un viņš paliek tur		
PR.1	Nospiežot pogas, lietotājs tiek pārvietots uz citu izvēlni, kā arī, uzsākot spēli, spēlētājs tiek pārvietots uz spēles pasauli.	veiksmīgs		17.05.2024	Dž. Svilis-Sudints
PR.2	Ielādējas saglabātā spēles pasaule ar pareizu spēlētāja stāvokli	veiksmīgs		17.05.2024	Dž. Svilis-Sudints
PR.3	Parādās skaņas līmeņu slīdņi, iespējams tos mainīt	veiksmīgs		17.05.2024	Dž. Svilis-Sudints
PR.4	Spēle aizveras pēc apstiprinājuma	veiksmīgs		17.05.2024	Dž. Svilis-Sudints
PR.5	Spēlētāja pozīcija mainās atbilstoši komandu virzienam	veiksmīgs		17.05.2024	Dž. Svilis-Sudints
PR.6	Koks tiek iegūts un parādās inventārā	veiksmīgs		17.05.2024	Dž. Svilis-Sudints
PR.7	Parādās tukšs inventāra logs ar 20 šūnām	veiksmīgs		17.05.2024	Dž. Svilis-Sudints
PR.8	Priekšmeti apmainās vietām	veiksmīgs		17.05.2024	Dž. Svilis-Sudints
PR.12	Mobi tuvojas spēlētājam	veiksmīgs		17.05.2024	Dž. Svilis-Sudints
PR.13	Samazinās spēlētāja veselība ik sekundi	veiksmīgs		17.05.2024	Dž. Svilis-Sudints
PR.14	Parādās nāves ekrāns ar paziņojumu un spēles ilguma informāciju	veiksmīgs		17.05.2024	Dž. Svilis-Sudints
PR.15	Mobiem samazinās veselība, tie var nomirt	veiksmīgs		17.05.2024	Dž. Svilis-Sudints
PR.16	Izveidota jauna pasaule ar šuma algoritmiem	veiksmīgs		17.05.2024	Dž. Svilis-Sudints
PR.17	Ģenerējas ala	veiksmīgs		17.05.2024	Dž. Svilis-

	ar resursiem			Sudints
PR.18	Ģenerējas jaunas teritorijas	veiksmīgs	17.05.2024	Dž. Svilis-Sudints
PR.19	Saglabājums tiek noņemts no saraksta	veiksmīgs	17.05.2024	Dž. Svilis-Sudints
PR.20	Parādās krājkastes inventārs	veiksmīgs	17.05.2024	Dž. Svilis-Sudints
PR.21	Krājkaste pazūd, priekšmeti parādās uz zemes	veiksmīgs	17.05.2024	Dž. Svilis-Sudints
PR.9	Komponents parādās inventārā, resursi samazinās	veiksmīgs	18.05.2024	A. Sidorčenko
PR.10	Mainās apgaismojums starp dienu un nakti ik pēc 5 minūtēm	veiksmīgs	18.05.2024	A. Sidorčenko
PR.11	Spavnojas 1 mobs, katru nakti vairāk	veiksmīgs	18.05.2024	A. Sidorčenko
PR.22	Priekšmets parādās uz zemes, pazūd no HotBar	veiksmīgs	18.05.2024	A. Sidorčenko
PR.23	Mainās aktīvais slots un vizuālā iezīme	veiksmīgs	18.05.2024	A. Sidorčenko

6.5. Testēšanas rezultāts

Sākotnēji tika pārbaudītas un fiksētas kritiskākās kļūdas, kuras bija nepieciešams nekavējoties novērst, jo tās būtiski traucēja spēles procesu. Trešās personas, kas testēja spēli, palīdzēja atklāt kļūdas un nepilnības, par kurām darba autors pat nebūtu iedomājies un, iespējams, pats nekad nepārbaudītu. Šie testētāji nedaudz palīdzēja mainīt spēles virzienu un pārveidot dažas funkcijas, lai tās darbotos citādi.

Viena no testētāju atsauksmēm:

"Spēle ir interesanta un neparasta – katru reizi pēc nāves rodas vēlme spēlēt vēlreiz, lai uzstādītu jaunu rekordu. Vai arī tad, kad kāds cits uzstāda rekordu ar ilgāku spēles laiku nekā tev, rodas vēlme ieiet spēlē un pārspēt šo rezultātu."

Secinājumi

Projekta izstrāde aizņēma aptuveni 100 stundas, ko iespējams redzēt arī darba autora Steam profilā, kurā tika palaists Godot Engine. Tas bija laikietilpīgs un izaicinošs process, kura laikā tika vairākkārt pārveidotas spēles daļas, līdz tika atrasts atbilstošs vizuālais stils un panākta vēlamā pasaules ģenerācijas kvalitāte.

1. secinājums - Liela daļa laika tika veltīta algoritmu parametru pielāgošanai, lai nodrošinātu loģisku, vizuāli pievilcīgu un vienmērīgu spēles vidi. Šis darbs ļāva labāk izprast pasaules ģenerēšanas principus un to, cik svarīga ir detalizēta testēšana.

2. secinājums - Izstrādes laikā tika izmantota plaša dokumentācija, mācību materiāli un video pamācības. Šis resurss kļuva par pamatu pašmācībai, ņemot vērā, ka tehnikuma programmā nebija iekļauts Godot dzinējs un trūka iepriekšējas pieredzes ar līdzīgām mehānikām.

3. secinājums - Spēļu elementu struktūra tika vairākas reizes pārskatīta un optimizēta. Viens no svarīgākajiem secinājumiem — nepieciešams laicīgi plānot spēles arhitektūru un pārdomāt datu saglabāšanas risinājumus (piemēram, .tres un PackedScene izmantošanu), lai izvairītos no pārlieku sarežģītas pārstrukturēšanas vēlākā posmā.

4. secinājums - Testēšana atklāja būtiskas kļūdas, kuras sākotnēji netika pamanītas. Līdzdalība testētājiem bija ļoti vērtīga, jo viņu komentāri palīdzēja mainīt funkcionalitāti un pilnveidot lietotāja pieredzi. Būtiski ir pieņemt ārēju kritiku un izmantot to kā iespēju uzlabot gala rezultātu.

5. secinājums - Projekta rezultāts, ņemot vērā, ka šī ir autora pirmā spēle, tiek vērtēts kā veiksmīgs. Lai gan tas vēl nav pilnībā pabeigts un var nebūt interesants visiem lietotājiem, iegūtā pieredze ir vērtīgs pamats turpmākai profesionālajai izaugsmei spēļu izstrādē.

7. Lietoto saīsinājumu un terminu skaidrojums

- **HotBar** – ir inventāra daļa, kas vienmēr tiek attēlota ekrāna apakšējā daļā un ļauj ātri piekļūt 9 aktīvajiem priekšmetiem.
- **Mobs** – Spēlēs mobi ir jebkuri kustīgi objekti, kas var mijiedarboties ar spēlētāju, piemēram, dzīvnieki vai ienaidnieki.
- **NR** – Nefunkcionālā prasība.
- **Perlin trokšņi** – Tas ir algoritms, kas rada dabisku izskatošos, gludus trokšņus, ko izmanto, lai ģenerētu pasauli spēlēs, piemēram, ainavas un tekstūras.
- **PR** – Funkcionālā prasība.

Literatūras un informācijas avoti

- **Godot Engine oficiālā dokumentācija**
  https://docs.godotengine.org
  (Pēdējo reizi apmeklēts: 17.05.2025)
- **KidsCanCode – Godot mācību materiāli**
  https://kidscancode.org/godot_recipes/
  (Pēdējo reizi apmeklēts: 1.04.2025)
- **DevForum – Procedural Generation with Noise in Godot**
  https://forum.godotengine.org
  (Pēdējo reizi apmeklēts: 12.02.2025)
- **GDQuest – Godot open source tutorials**
  https://www.gdquest.com
  (Pēdējo reizi apmeklēts: 4.01.2025)
- **LucidCharts – UML diagrammas**
  https://www.lucidchart.com/pages/uml-class-diagram
  (Pēdējo reizi apmeklēts: 10.04.2025)

Pielikumi

1. Pielikums

Šeit ir dienas un nakts maiņas koda fragments spēlē

```
○ ○ ○

extends CanvasModulate  # Paplašina CanvasModulate mezglu, lai
varētu modulēt ekrāna krāsas (piemēram, diennakts ciklam)

@onready var sky_gradient: GradientTexture1D = preload("res://
Assets/shit/daynightcycle-gradient-texture.tres")
# Iepriekš ielādēta gradienta tekstūra, kas attēlo debesu krāsu
pāreju diennakts laikā

@onready var time_label: Label = $CanvasLayer/MarginContainer/
time_label
# Atsauce uz teksta lauku, kurā tiks parādīts pašreizējais spēles
laiks

@onready var total_ingame_minutes : int
# Kopējais noietais spēles laiks minūtēs

var spawned_day : int
# Fiksē, kurā dienā pēdējo reizi tika palaisti ienaidnieki

func _ready() -> void:
    add_to_group('time')  # Pievieno šo mezglu grupai 'time', lai
citi mezgli varētu viegli atsaukties

func get_time() -> String:
    var time : String = "Day %d, %02d:%02d" % [current_day,
current_hour, current_minute]
    return time  # Atgriež pašreizējo laiku kā tekstu: "Day X,
HH:MM"

func get_min() -> int:
    var min : int = total_ingame_minutes
    return min  # Atgriež kopējo spēlē pavadīto minūšu skaitu

signal time_tick(day: int, hour: int, minute: int)
# Signāls, kas tiek izsaukts katru reizi, kad mainās minūte

var internal_time: float = 1
# Iekšējais laika skaitītājs, kuru izmanto laika aprēķināšanai

var last_checked_minute: int = -1
# Lai pārbaudītu, vai minūte jau ir apstrādāta

var current_day: int = 0
var current_hour: int = 0
var current_minute: int = 0
# Mainīgie pašreizējā dienas, stundas un minūtes uzglabāšanai

const DAY_MINUTES = 1440  # Minūtes vienā spēles dienā (24h * 60min)
const HOUR_MINUTES = 60   # Minūtes vienā spēles stundā
const TIME_SCALE = TAU / DAY_MINUTES
# Mēroga konstante, lai pārveidotu iekšējo laiku (radiānos) uz
minūtēm, izmantojot sin funkciju
```

Šeit ir dienas un nakts maiņas koda otrais fragments spēlē

```
func _process(delta: float) -> void:
    internal_time += delta / 4
    # Katru kadru pievieno laiku, sadalītu uz 4, lai laiks ritētu
lēnāk

    var normalized_time = (sin(internal_time - PI / 2.0) + 1.0) *
0.5
    # Normalizēts laiks no 0 līdz 1, izmantojot sinusoīdu (dabiska
pāreja starp dienu un nakti)

    self.color = sky_gradient.gradient.sample(normalized_time)
    # Iestata ekrāna toni atbilstoši gradientam un laikam

    _update_clock()  # Atjauno spēles laika rādītājus

func _update_clock() -> void:
    total_ingame_minutes = int(internal_time / TIME_SCALE)
    # Aprēķina kopējo spēles minūšu skaitu, pārrēķinot no iekšējā
laika

    current_day = total_ingame_minutes / DAY_MINUTES
    var minutes_today = total_ingame_minutes % DAY_MINUTES
    # Atdala minūtes šodienai no kopējā laika

    current_hour = minutes_today / HOUR_MINUTES
    current_minute = minutes_today % HOUR_MINUTES
    # Pārrēķina stundas un minūtes šodien

    if last_checked_minute != current_minute:
        last_checked_minute = current_minute
        time_tick.emit(current_day, current_hour, current_minute)
        # Ja minūte mainījusies — izsauc signālu ar jauno laiku

        time_label.text = "Day %d, %02d:%02d" % [current_day,
current_hour, current_minute]
        # Atjauno teksta lauku ar aktuālo dienu un laiku

        if current_hour == 2 and current_day != spawned_day:
            spawned_day = current_day
            SceneManager.spawn_enemy(int(ceil(spawned_day * 2)))
            # Ja ir 2:00 naktī un šī diena vēl nav apstrādāta —
palaiž ienaidniekus,
            # to skaits palielinās, pamatojoties uz dienu (piemēram,
2 ienaidnieki 1. dienā, 4 otrajā u.tml.)
```

3. Pielikums

Šis ir spēlētāja pirmais kods

```
func _change_health(new_health):
    health = new_health
    healthbar.health = health  # Atjauno UI veselības joslu

func _change_armor(new_armor):
    armor = armor_bar.armor
    armor_effect = 1 - (armor/100 * 0.8)  # Aprēķina bruņu efektu — 
samazina ienākošo bojājumu līdz pat 80%

func _ready() -> void:
    add_to_group('player')  # Pievieno šo objektu grupai "player"
    var pause = pause_scene.instantiate()
    add_child(pause)  # Izveido un pievieno pauzes izvēlni

    #var result = result_scene.instantiate()
    #add_child(result)  # (komentēts) iespēja pievienot rezultātu 
ainu

    healthbar.init_health(health)  # Inicializē veselības joslu ar 
sākotnējo vērtību
    armor_bar.init_armor(armor)    # Inicializē bruņu joslu

    if not is_loaded_from_save:
        var ui = ui_scene.instantiate()
        add_child(ui)
        ui.owner = self  # Pievieno UI un iestata īpašnieku

    GlobalSFX.apply_volume(damage_sound)  # Pielāgo skaļumu globāli 
ievainojuma skaņai
```

Šis ir spēlētāja otrais kods

```
func take_damage(damage : int):
    damage_sound.play()  # Atskaņo bojājuma skaņu

    if is_hurt: return  # Ja spēlētājs jau ir ievainots, pārtrauc
funkciju
    is_hurt = true

    match direction:  # Atskaņo atbilstošo animāciju atkarībā no
virziena
        "down":
            _animated_sprite.play("hurt_front")
        "back":
            _animated_sprite.play("hurt_back")
        "left":
            _animated_sprite.play("hurt_right")
            _animated_sprite.flip_h = true
        "right":
            _animated_sprite.play("hurt_right")
            _animated_sprite.flip_h = false

    health -= damage * armor_effect  # Samazina veselību, ņemot vērā
bruņu efektu
    await _animated_sprite.animation_finished  # Pagaida, līdz
animācija beidzas
    is_hurt = false

    if health <= 0:
        GlobalSFX.play(DEATH)  # Atskaņo nāves skaņu

        var time_node = get_tree().get_first_node_in_group("time")
 # Atrod laika pārvaldnieku
        var time = time_node.get_time()
        var minutes_ingame = time_node.get_min()

        SceneManager.time = time
        SceneManager.minutes_ingame = minutes_ingame

        get_tree().change_scene_to_file("res://Scenes/Menu_Scenes/
result.tscn")  # Atver rezultātu ainu
```

5. Pielikums

Šis ir spēlētāja trešais kods

```
func _physics_process(_delta: float) -> void:
    get_input()
    move_and_slide()  # Fizikālā kustība, izmantojot ievadi

func _process(_delta: float) -> void:
    if is_hurt:
        return  # Ja ievainots – neļauj pārvietoties

    # Atskaņo animācijas atkarībā no nospiestās pogas
    if Input.is_action_pressed("down"):
        _animated_sprite.play("walk")
        direction = "down"
    elif Input.is_action_pressed("up"):
        _animated_sprite.play("walk_back")
        direction = "back"
    elif Input.is_action_pressed("left"):
        _animated_sprite.play("walk_right")
        _animated_sprite.flip_h = true
        direction = "left"
    elif Input.is_action_pressed("right"):
        _animated_sprite.play("walk_right")
        _animated_sprite.flip_h = false
        direction = "right"

    if velocity.length() == 0:
        # Ja stāv uz vietas, attēlo atbilstošu idle animāciju
        match direction:
            "down":
                _animated_sprite.play("idle")
            "back":
                _animated_sprite.play("idle_back")
            "left":
                _animated_sprite.play("idle_right")
                _animated_sprite.flip_h = true
            "right":
                _animated_sprite.play("idle_right")
                _animated_sprite.flip_h = false
```

6. Pielikums

Šis ir pirmais pasaules ģenerēšanas kods

```
○ ○ ○


func _ready() -> void:
    add_to_group("proc_gen_world")  # Pievieno mezglu grupai
    var player = get_tree().get_first_node_in_group("player")
    player.global_position = Vector2(50*16 , 50*16)  # Novieto
spēlētāju kartes centrā

    # Trokšņa ģenerators reljefam un kokiem
    noise = noise_height_text.noise
    tree_noise = noise_tree_text.noise
    noise.seed = randi()
    tree_noise.seed = randi()

    if not is_loaded_from_save:
        generate_world()  # Izsauc pasaules ģenerēšanu
```

Šis ir otrais pasaules ģenerēšanas kods

```
func generate_world():
    for x in range(width):  # Iterē pa X asi visā kartes platumā
        for y in range(height):  # Iterē pa Y asi visā kartes
augstumā
            var noise_val = noise.get_noise_2d(x, y)  # Iegūst
trokšņa vērtību reljefam šajā punktā
            var tree_noise_val = tree_noise.get_noise_2d(x, y)  #
Iegūst trokšņa vērtību objektiem (koki, akmeņi)

            if x > 10 and y > 10:
                # Reģistrē šo punktu kā potenciālu vietu, kur
parādīties ienaidniekiem
                enemy_spawnpoints.append(Vector2i(x * 16, y * 16))

            var normalized = remap(noise_val, -1.0, 1.0, 0.0, 1.0)
 # Normalizē trokšņa vērtību uz intervālu [0;1]

            # --- RUDENS BIOMS ---
            if normalized >= 0 and normalized <= 0.45:
                fall_tiles_arr.append(Vector2i(x, y))  # Pievieno
punktu rudens bioma slānim

                if tree_noise_val > 0.90:
                    spawn_tree(Vector2i(x * Tile_Size, y *
Tile_Size), "fall")  # Ģenerē koku

                if tree_noise_val > 0.79 and tree_noise_val < 0.81:
                    ground_layer.set_cell(Vector2i(x, y), 0,
fall_bush_arr.pick_random())  # Buši

                if tree_noise_val > 0.81 and tree_noise_val < 0.84:
                    ground_layer.set_cell(Vector2i(x, y), 0,
fall_grass_arr.pick_random())  # Zāle

                if tree_noise_val > 0.84 and tree_noise_val < 0.87:
                    ground_layer.set_cell(Vector2i(x, y), 0,
fall_flowers_arr.pick_random())  # Ziedi

                if tree_noise_val > 0.87 and tree_noise_val < 0.89:
                    ground_layer.set_cell(Vector2i(x, y), 0,
small_stone_arr.pick_random())  # Akmeņi

                if tree_noise_val > 0.895 and tree_noise_val < 0.90:
                    spawn_stone(Vector2i(x * Tile_Size, y *
Tile_Size), "stone")  # 3D akmens objekts

                if tree_noise_val > 0.503 and tree_noise_val <
0.505:
                    spawn_mine(Vector2i(x * Tile_Size, y *
Tile_Size))  # Izvieto ieeju raktuvēs
```