Liepājas Valsts tehnikums

Datorspēlē “HockeyShot”

Kvalifikācijas eksāmena praktiskās daļas dokumentācija

Darba autors:
Tomijs Būmerts, 4PT-2

Darba vadītājs:
Skolotājs, Raimonds Kristovskis

Eksāmena datums 2025.gada __. Jūnijs

Liepāja 2025

Satura rādītājs

Ievads

Šis dokuments ir sagatavots saskaņā ar Liepājas Valsts tehnikuma mācību programmas "Programmēšanas tehniķis" noslēguma darba prasībām. Tā ietvaros paredzēts izstrādāt kvalifikācijas darbu, kas šajā gadījuma ir sporta spēle "HockeyShot". Tā būs datorspēle, veltīta Latvijā īpaši populārajam sporta veidam – hokejam. Šāda spēles izstrādes ideja tiks izvēlēta, jo piedāvājums šajā žanrā ir salīdzinoši ierobežots uz jebkuru spēles platformu.

Spēle tiks veidota no trešās personas skata, kas ļaus spēlētājiem pilnībā pārredzēt laukumu un sekot spēles gaitai. Katrs spēlētājs varēs vadīt vienu hokejistu, kontrolējot viņa kustības un saspēlēs darbības spēles laikā. Spēle būs pieejama multiplayer režīmā, kurā spēlētāji varēs sacensties savā starpā, veidojot komandas pēc savas velmes un izstrādāt stratēģijas, lai sasniegtu galveno spēles mērķi – gūt vārtus un uzvarēt. Tāpat būs pieejams arī treniņu režīms, kurā spēlētāji varēs apgūt un izprast spēles kontroles un fiziku, lai uzlabotu savas spējas un kļūtu par labākiem spēlētājiem.

Spēle piedāvās divus režīmus: tiešsaistes režīmu un treniņu režīmu. Šajos režīmos spēlētāji varēs sacensties savā starpā, nodrošinot dinamisku un komandas sadarbību veicinošu pieredzi.Spēles izstrāde ietvers vairākus svarīgus posmus. Sākumā tiks veikta uzdevuma detalizēta analīze, lai noskaidrotu spēles mērķus, funkcionalitāti un lietotāju vajadzības. Tiks definētas programmatūras prasības, ietverot gan funkcionālas, gan nefunkcionālās prasības. Projektā tiks izvēlēti atbilstoši rīki un tehnoloģijas, kā arī piedāvātas alternatīvas līdzīgu projektu realizācijai. Sistēmas modelēšanas un projektēšanas posmā tiks izstrādātas shēmas un diagrammas, kas nodrošina augstas kvalitātes projekta izstrādi ar visām plānotajām funkcijām. Paredzēts arī spēles testēšanas posms, kurā laikā iegūtie rezultāti tiks dokumentēti.

Dokumenta autors noslēgumā sniegs secinājumus, izvērtējot iestrādes procesu un sasniegto rezultātu.

1.Uzdevuma formulējums

Produktu nepieciešams izstrādāt, jo pašlaik datorspēļu tirgū ir ļoti ierobežots piedāvājums hokeja spēļu žanrā, un lielākā daļa šī tipa spēļu ir pieejamas tikai uz spēļu konsolēm. Šī situācija rada nepieciešamību izveidot datoram piemērotu hokeja spēli, kas būtu pieejama plašākai auditorijai, ieskaitot tos, kuriem nav pieejamas spēļu konsoles. Šāda spēle ne tikai aizpildītu tirgus trūkumu, bet arī sniegs iespēju spēlētājiem izbaudīt hokeja spēles pieredzi uz datora platformas.

Uzdevuma mērķis ir radīt pieejamu, aizraujošu un kvalitatīvu hokeja spēli, kas apvieno vienkāršību ar izaicinājumu. Spēlei būs intuitīvas kontroles mehānismi, kas padarīs to viegli saprotamu iesācējiem, taču tās dziļākās mehānikas ļaus pieredzējušiem spēlētājiem pilnveidot savas prasmes un stratēģijas. Tas veicinās spēlētāju iesaisti un regulāru atgriešanos pie spēles, radot ilgstošu interesi un vēlmi pilnveidoties.

Lai sasniegtu mērķi, spēle tiks izstrādāta datorplatformai, izmantojot mūsdienīgus izstrādes rīkus un tehnoloģijas. Izstrādes galvenie uzdevumi ietvers vienkāršu un intuitīvu vadības mehānismu izveidi, reālistisku ripas fiziku un spēlētāju kustību simulāciju uz ledus. Spēle koncentrēsies uz multiplayer pieredzi un piedāvās divus galvenos režīmus tiešsaistes režīmu, nodrošinot dinamisku un aizraujošu sadarbību starp spēlētājiem. Kā arī treniņu režīmu kurš ļaus spēlētājiem apgūt spēles pamatus un uzlabot savas prasmes.

Izstrādes procesā tiks izveidoti precīzi sistēmas modeļi un diagrammas, kas aprakstīs spēles darbības loģiku un mehānikas. Tiks veikta rūpīga spēles testēšanā, lai identificētu un novērstu kļūdas, garantējot spēles stabilitāti un kvalitāti. Uzsvars tiks likts uz spēles funkcionalitātēs atbilstību dokumentācija izvirzītājām prasībām.

Mērķis tiks uzskatīts, par sasniegtu, kad spēle piedāvās intuitīvu un viegli saprotamu vadību, kā arī pietiekami dziļas mehānikas, kas motivēs spēlētājus uzlabot savas prasmes un atgriezties pie spēles. Pozitīvi testēšanas rezultāti un pilnīga spēles funkcionalitātes atbilstība izvirzītājām prasībām būs galvenais apliecinājums, ka spēle ir veiksmīgi pabeigta. Galarezultāta spēlei jānodrošina stabila, aizraujoša un kvalitatīva pieredze dažādu līmeņu spēlētājiem.

2.Programmatūras prasību specifikācija

Šajā nodaļā tiek aprakstītas programmatūras prasību specifikācijas, lai veicinātu kvalitatīvu produkta izstrādi un izstrādes procesu. Tiks aprakstītas funkcionālas un nefunkcionālas prasības datorspēlei "HockeyShot", aprakstot sīki un precīzi, nepieciešamās prasības kuras jānodrošina gala produktam veicinās projekta izstrādes kvalitāti un izstrādes vienkāršumu.

2.1. Produkta perspektīva

Šī spēle uzlabos cilvēka zināšanas par hokeja noteikumiem, un komandas darbu un stratēģijas domāšanu. Produktu iespējams pilnveidot dažādos veidos, piemēram, var tikt izveidoti mākslīga intelekta pretiniekus kuriem būs iespēja uzlikt savu grūtības pakāpi. Kā arī var izveidot papildus noteikumus kuri ietekmētu spēles gaitu.

Spēle ir pietiekami universāla, lai to varētu spēlēt vairāku vecumu cilvēku grupas, un dažādu spēlētāju līmeņi.

2.2. Sistēmas funkcionālas prasības

P.1. Datorspēles "HockeyShot" galvenā izvēlne

Mērķis:

Funkcija "Galvenā izvēle" nodrošina iespēju lietotājam izvēlēties vai vēlas uzsākt spēli "HockeyShot", vai samainīt iestatījumus vai nu apturēt spēli.

Ievaddati:

Atvērta datorspēle "HockeyShot"

Apstrāde:

1. Funkcija pārbauda vai lietotājs ir atvēris datorspēli "HockeyShot" ;
2. Funkcija pārbauda vai lietotājs atrodas ainā "Galvenā izvēle";

Izvaddati:

1. Parādās poga "Spēlēt"
2. Parādās poga "Iestatījumi"
3. Parādās poga "Iziet"
4. Sāk skanēt fona mūzika.

P.2. Datorspēles “HockeyShot” spēles režīmu izvēle

Mērķis:

Funkcija nodrošina spēles režīmu izvēli starp pieejamajiem režīmiem

Ievaddati:

Uzspiesta poga “Sākt” galvēnās izvēles ainā

Apstrāde:

1. Funkcija pārbauda vai lietotājs atrodas galvenā ainā
2. Funkcija pārbauda vai lietotājs ir nospiedis pogu “Sākt”

Izvaddati:

1. Parādās režīmu izvēles panelis
2. Parādās režīmu “Treniņu režīms”
3. Parādās režīms “Tiešsaistes režīms”

P.3. Datorspēles “Treniņu režīma” uzsākšana

Mērķis:

Funkcijas “Treniņu režīma” ainas palaišana

Ievaddati:

Spēles režīmu izvēles paneli uzspiesta poga “Treniņu režīms”

Apstrāde:

1. Funkcija pārbauda vai lietotājs nospiedis pogu “Treniņu režīms”

Izvaddati:

1. Atveras aina “Treniņu režīms”.
2. Spēlētājs novietots spēles laukuma centrā
3. Tiek novietota hokeja ripa spēles laukumā

P.4. Datorspēles režīma “Tiešsaistes režīma” uzsākšana

Mērķis:

Funkcija nodrošina spēles “Tiešsaistes” uzsākšanu

Ievaddati:

Lietotajs nospiež pogu “Spēlēt” zem attēla 2v2

Apstrāde:

Funkcija pārbauda vai lietotajs ir nospiedis pogu “Spēlēt”

Izvaddati:

1. Atveras “Spēles gaiteņa” panelis
2. Izvadās spēles istabas kods
3. Izvadās pievienoto spēlētāju saraksts
4. Izvadās spēles uzsākšanas pogas

P.5. Datorspēles uzsākšana no "Spēles gaiteņa"

Mērķis:

Funkcija nodrošina spelētaju pārvietošanu uz atbilstošo ainu no "Spēles gaiteņa".

Ievaddati:

Izvēlēts kāds no spēles režīms kurš atbalsta vairākus spēlētājus un atvērts "Spēles gaiteņa" panelis

Apstrāde:

1. Funkcija pārbauda, vai ir atvērts spēles gaitēņa panelis
2. Funkcija pārbauda, vai nospiesta "spēlēt" poga gaiteņa panelī.

Izvaddati:

Spēlētājs, vai spēlētāji tiek pārvietoti uz atbilstošo izvēlēto spēles ainu.

P.6. Datorspēles iestatijumu maiņa

Mērķis:

Funkcija "Iestatijumu maiņa" nodrošina lietotājam mainīt esošos spēles iestatijumus.

Ievaddati:

Ar kreiso peles taustiņu ir nospiesta poga "Iestatijumi".

Apstrāde:

Funkcija pārbauda, vai lietotājs atrodas ainā "Galvenā izvēlne".

Izvaddati:

1. Atveras iestatijumu panelis
2. Parādās slaideris ar iespēju mainīt spēles skaņas skaļumu
3. Parādās dropdowns ar iespēju mainīt spēles izsķirtspēju
4. Parādās toggle poga ar iespeju spēli padarīt pilnekrāna režīmā

P.7. Datorspēles skaļuma maiņa

Mērķis:

Funkcija "Skaļuma maiņa" nodrošina lietotājam mainīt spēles skaļumu.

Ievaddati:

Lietotājs turot peles kreiso klikšķi kustina skaļuma slaideri

Apstrāde:

1. Funkcija pārbauda vai lietotājs ir atrodas "Iestatījumu" panelī.
2. Funkcija pārbauda, kur atrodas skaļuma slaideris

Izvaddati:

Spēles skaļums mainās

P.8. Datorspēles izšķirtspējas maiņa

Mērķis:

Funkcija “Izšķirtspējas maiņa” nodrošina lietotājam mainīt spēles izsķirtspēju

Ievaddati:

Ar kreiso peles klikšķi uzspiests un izvēlēta atbilstoša spēles izsķirtspēja.

Apstrāde:

Funkcija pārbauda vai lietotājs ir izvēlējies izšķirtspēju

Izvaddati:

Mainās spēles izšķirtspēja

P.9. Datorspēles pilnekrāna mainīšana

Mērķis:

Funkcija “pilnekrāna izvēle” nodrošina lietotājam iespēju uzlikt pilnekrāna režīmu, vai spēlēt loga režīmā

Ievaddati:

Ar kreiso klikški ir ieklikšķināts toggle pogā, to ieķeksējot

Apstrāde:

Funkcija pārbauda, vai lietotājs ir ieķeksējis pogu

Izvaddati:

Spēle mainās pilnekrāna režīma, vai loga režīmā.

P.10. Datorspēles iestatijuma apstiprināšana

Mērķis:

Funkcija “Iestatijuma apstiprināšana” nodrošina iestatijumu saglabāšanu

Ievaddati:

Lietotajs “Iestatijuma” panelī nospiež pogu “Apstiprināt”

Apstrāde:

Funkcija saglabā lietotāja izvēlētos iestatijumus un tos saglabā

Izvaddati:

Izvēlētie iestatijumi tiek saglabāti visu spēļu laikā.

P.11. Datorspēles iestatijumu attiestatīšana

Mērķis:

Funkcija "Iestatijumu attiestatīšana" nodrošina spēlētāja izvēlēto iestatijumu attiestatīšana uz sākuma izvēlētajiem.

Ievaddati:

Spēlētajs nospiež pogu "Attiestatīt" iestatijumu panelī

Apstrāde:

Funkcija atgriež mainīto iestatijumu atgriešanos uz sākotnējo izvēlni.

Izvaddati:

Spēlētāja iestatijumi tiek atgriezti uz sākotnējām vērtībām.

P.12. Datorspēles aizvēršana

Mērķis:

Funkcija nodrošina spēles aizvēršanu no galvenās ainas

Ievaddati:

Lietotajs galvenajā ainā nospiež pogu "Iziet"

Apstrāde:

1. Funkcija pārbauda, vai lietotājs atrodas galvenajā ainā
2. Funkcija pārbauda, vai lietotājs nospiedis pogu "Iziet"

Izvaddati:

Spēle tiek aizvērta

P.13. Datorspēles spēlētāja kustība pa laukumu

Mērķis:

Funkcija "PlayerMovement" nodrošina spelētāja kustību pa laukumu

Ievaddati:

Taustiņi wasd un bultiņu taustiņi

Apstrāde:

1. Funkcija pārbauda, vai lietotājs atrodas kādā no spēles režīmiem
2. Funkcija pārbauda, vai lietotajs nospiedis kādu no kustību taustiņiem

Izvaddati:

Spēlētājs kustas atkarība no nospiestā taustiņa

P.14. Datorspēles "HockeyShot" spēlētāja skriešanas funkcionalitāte

Mērķis:

Funkcija nodrošina spēlētāja ātrāku kustību pa spēles laukumu, iedodot spēlētājam mazu ātruma palielinājumu ar dažu sekunžu atgūšanas periodu.

Ievaddati:

Nospiežot taustiņu "Shift"

Apstrāde:

1. Funkcija pārbauda, vai lietotājs ir izvēlējies kādu no spēles režīmiem
2. Funkcija pārbauda, vai lietotājs ir nospiedis skriešanas taustiņu
3. Funkcija pārbauda, vai lietotājam, nav spēka izsīkums.

Izvaddati:

Spēlētājs kustības virzienā iegūst paātrinājumu.

P.15. Datorspēles "HockeyShot" komandas izvēle

Mērķis:

Funkcija "Komandas izvēle" nodrošina lietotājam iespēju izvēlēties spēles komandu "Spēles gaiteņa" panelī pirms spēles uzsākšanas

Ievaddati:

Ar kreiso peles klikšķi lietotājs izvēlas komandu: "Zilā", "Sarkanā"

Apstrāde:

1. Funkcija pārbauda, vai lietotajs izvēlējies komandu
2. Funkcija saglabā izvēlēto komandu.

Izvaddati:

1. Lietotājs izvēlēto komandas krāsu redz uz sava spēlētāja
2. Komandas īpašie atribūti tiek iestatīti spēles sākumā, vai maiņas gadījumā (formas krāsa, ķiveres krāsa utt.)

P.16. Datorspēles kontroles ātrā apstāšanas

Mērķis:

Funkcija "Ātrā apstāšanas" nodrošina iespēju spēlētājam spēles laikā ātri apstāties uz ledus

Ievaddati:

Lietotājs nospiedis taustiņu "Space"

Apstrāde:

1. Funkcija pārbauda, vai lietotājs ir nospiedis taustiņu "Space"
2. Funkcija palielina spēlētāja pretestību

Izvaddati:

Lietotāja kustība tiek apturēta.

P.17. Spēles "HockeyShot" iesisto vārtu skaitīšana

Mērķis:

Funkcija "SpelesPunkti" nodrošina spēlētāja iesisto vārtu noteikšanu

Ievaddati:

Ripa tiek ievietota hokeja vārtu objekta zonā

Apstrāde:

1. Funkcija pārbauda, vai ripa ir šķērsojusi vārta ieejas līniju
2. Funkcija nosaka, kurš spēlētājs ir pieskāries ripai pēdējais

Izvaddati:

1. Punkts tiek pieskaitīts komandai

P.18. Punktu pievienošana komandai

Mērķis:

Funkcija "Punktu skaitīšana" nodrosina iesisto punktu uzskaiti

Ievaddati:

Ripa pieskarās vārtu zonai

Apstrāde:

Funkcija pārbauda, vai ripa ir škērsojusi vārtu ieejas līniju

Izvaddati:

Punktu skaits tiek palielināts attiecīgai komandai

P.19. Datorspēles "Hokeja" spēles laika atskaite

Mērķis:

Funkcija nodrošina, laiku un 3. spēles puslaiku noteikšanu

Ievaddati:

Noteiktais spēles laiks 5. minūtes, ar trīs puslaikiem

Apstrāde:

1. Funkcija veic laika atskaiti spēles laikā uzrādot spēlētājiem sekundes
2. Funkcija nosaka, ja gadijumā spēles puslaiks ir beidzies tiek saglabāti punkti un atkārtoti palaista starta pozīcijas un atsākas laika atskaite.
3. Pēc laika atskaites tiek izsaukta funkcija "UzvarētājuNoteikšana"

Izvaddati:

Spēlētajiem tiek parādīts ar UI elementu palīdzību atlikušais laiks un perioda numurs, piemērām, periods Q1.

P.20. Spēlētāja ripas paņemšana

Merķis:

Funkcija nodrošina kontrolējot spēlētāju, lai tiktu paņemta hokeja ripa un ar to varētu veikt darbības.

Ievaddati:

Spēlētājs savu spēlētāju novieto ripas apkārtējā zonā un nospiež taustiņu e.

Apstrāde:

1. Funkcija pārbauda, vai spēlētāja un ripas zonas saskarās
2. Funkcija pārbauda, vai lietotājs nospedis taustiņu e.
3. Funkcija pievieno ripu pie spēlētāja elementa, sekojot tam.

Izvaddati:

Ripa vizuāli seko spēlētājam spēles laukumā

P.21. Spēlētāja iespēja mest ripu

Mērķis:

Funkcija “Puck” nodrošina spēlētāju ar iespēju mest ripu, lai padotu pasi, kā arī lai mestu ripu vārtos

Ievaddati:

Spēlētājs nospiež kreiso peles taustiņu

Apstrāde:

1. Funkcija pārbauda, vai lietotājam ir pievienojies ripas objekts
2. Funkcija pārbauda, vai lietotājs ir nospiedis kreiso peles taustiņu

Izvaddati:

No spēlētāja tiek objekta tiek izmesta ripa, spēlētāja skatīšanas virzienā.

P.22. Kameras sekošana spēlētājam

Mērķis:

Funkcija nodrošina kameras sekošanu spēlētājam no mugurpuses, sekojot spēlētāja rotācijai.

Ievaddati:

Spēlētāja ievietošana spēles ainā

Apstrāde:

1. Funkcija pārbauda vai spēlētājs ir novietots spēles ainā
2. Spēlētāja objektam tiek pievienota kamera, tā paredzētājā vietā

Izvaddati:

1. Lietotājam vizuāli redz kameras kustību, spēles ainā.
2. Kamera kustas līdz ar spēlētāju nodrošinot, to ka, kamera seko rotācijai un vienmēr ir vērsta kustības virziena.

P.23. Spēlētāja iespēja pievienoties citiem spēlētājiem.

Mērķis:

Nodrošināt iespēju spēlētājiem savā starpā pievienoties viens otram, radot iespēju spēlēt savā starpā ar "RelayManager"

Ievaddati:

Spēlētājs "Spēles gaiteņa" paneļa ievades logā ieraksta spēles hosta ģenerēto kodu.

Apstrāde:

1. Funkcija pārbauda, vai lietotāja ievadītais kods ir atbilst, kādā no izveidotājām spēlēm.
2. Funkcija pievieno klientu spēles ainai, attiecīgajā komandā

Izvaddati:

Spēlētājs tiek pievienots attiecīgajai spēles ainas telpai, ar konkrēto īpašnieku.

P.24. Spēlētāja iespēja izvēlēties komandu

Mērķis:

Nodrošināt iespēju spēlētājiem izvēlēties komandu, starp zilo, sarkano. Tas dos iespēju izvēlēties savus biedrus, vai pretiniekus ar ko spēlēt komandā, vai pret

Ievaddati:

Spēlētājs "Spēlēs gaitēņa" panelī nospiežot pogu "zils", "sarkans"

Apstrāde:

1. Funkcija pārbauda, vai lietotājs izvēlējies spēlētāju
2. Funkcija pārbauda, vai lietotājs nospiedis pogu "zils",vai "sarkans"

Izvaddati:

1. Spēlētāja vārds tiek attiecīgi iekrāsots izvēlētās komandas krāsa
2. Spēlētaja forma spēles laikā attiecīgi krāsota komandas krāsā.

P.25. Pauzes ekrāns spēles ainā

Mērķis:

Nodrošināt iespēju spēlētājiem spēles laikā, mainīt iestatijumus, kā arī iziet no esošās spēles istabas.

Ievaddati:

Spēlētājs spēles ainā nospiež taustiņu “esc”

Apstrāde:

1. Pārbauda, vai atrodas spēles ainā
2. Notiek pārbaude, vai nospiests taustiņš “esc”

Izvaddati:

1. Spēles ainā atveras panelis
2. Spēles atvērtajā panelī paradās poga “Atgriezties”, “Iestatijumi”, “Iziet no spēles”, “Uz sākumu”

P.26. Pauzes ekrāna iestatijumi

Mērķis:

Nodrošināt iestatijumu maiņu spēles laikā, kā arī spēlētāja iestatijumus saglabāt.

Ievaddati:

Spēlētājs spēles pauzes ekrāna nospiež pogu “Iestatijumi”

Apstrāde:

Pārbauda, vai nospiesta poga panelī “Iestatijumi”

Izvaddati:

Atveras spēles panelis ar iestatijumiem

P.27. Spēles istabas izveide

Mērķis:

Nodrošināt iespēju izveidot spēles istabu, uz kuru lietotāji spēs pievienoties, lai spēlētu kopā.

Ievaddati:

Spēlētājs nospiež pogu izveidot uz kādu no spēles režīmiem

Apstrāde:

1. Pārbauda, vai ir izvēlēts, kāds no pieejamajiem spēles režīmiem
2. Pārbauda, vai ir nospiesta poga izveidot istabu.

Izvaddati:

1. Atveras "Spēles gaiteņa" panelis
2. UI panelī tiek parādīts istabas kods, ko nodot citiem spēlētājiem.

P.28.Spēlētāju pozīciju novietošana pēc vārtiem

Mērķis:

Novietot spēlētājus atpakaļ, izvēlētājā pozīcijā pēc vārtu iegūšanas

Ievaddati:

Ripas objekta vārtu līnijas šķērsošana

Apstrāde:

1. Funkcija pārbauda, vai ripa ir šķērsojusi vārtu sākuma līniju
2. Funkcija iegūst spēlētāju novietošanas pozīcijas no inspektora

Izvaddati:

Spēlētāji tiek novietoti savās izvēlētājās pozīcijās.

P.29. Spēlētāju iespēja nomainīt segvārdu

Mērķis:

Nodrošināt iespēju katram spēlētājam sevi identificēt, pēc savām velmēt nodrošinot iespēju nomainīt spēlētāja segvārdu.

Ievaddati:

Iestatījumu panelī ievada lauciņa ieraksta savu segvārdu

Apstrāde:

1. Funkcija pārbauda, vai lietotājs ir ievadījis ievades lauciņā segvārdu
2. Funkcija pārbauda, vai lauciņš nav tukšs
3. Funkcija pārbauda, vai lietotājs ir nospiedis pogu apstiprināt

Izvaddati:

Spēlētāja segvārds parādas spēles ainās un spēles istabā.

P.30. Spēlētāja iespēja atgriezties atpakal no iestatijumiem

Mērķis:

Nodrošina iespēju lietotājam no spēles paneļa atgriezties uz kādu no citiem paneļiem, vai to aizvērt

Apstrāde:

1. Funkcija pārbauda, vai lietotājs ir atvēris "Iestatījuma paneli"
2. Funkcija pārbauda, vai lietotājs ir nospiedis pogu "Atpakaļ"
3. Funkcija paslēpj iestatījuma paneli

Izvaddati:

Spēlētājam vizuāli tiek aizvērts "Iestatijumu" panelis

P.31. Ripas atgriešanas sākotnēja pozīcija

Mērķis:

Nodrošināt funkcionalitāti, lai iesitot vārtus ripa atgriežas sākotnējā centra pozīcijā

Apstrāde:

1. Funkcija pārbauda, vai ripa ir šķērsojusi vārtu līnijas robežas
2. Funkcija atgriež ripu sākotnēja atrašanās vietā, ko nosaka spēles režīms

Izvaddati:

Ripa objekts tiek novietots spēles noteiktajā pozīcija, kurā tā spēles sakumā tika saglabāta.

P.32. Ripas zagšana no spēlētājiem

Mērķis:

Nodrošinat funkcionalitāti, lai spēlētāji viens otram varētu zagt ripu

Apstrāde:

1. Funkcija pārbauda, vai ripa un cits īpašnieks atrodas blakus
2. Funkcija pārbauda, vai lietotājs ir nospiedis taustiņu e

Izvaddati:

Ripa tiek nozagta no cita spēlētāja un tiek pievienota savam spēlētājam

2.3 Sistēmas nefunkcionālas prasības

2.3.1. Valoda

Datorspēle "HockeyShot" ir jābūt izstrādātai Latvijas Republikas oficiālā valodā, kā arī ir jābūt izstrādātai Angļu valodā.

2.3.2. Saskaņotība

Datorspēlei “HockeyShot” ir jābūt saskaņotai tā, lai būtu spēles kontroles viegli saprotamas, un lai nebūtu problēmas spēlēt šo spēli.

2.3.3. Vizuālais izskats

Datorspēlei “HockeyShot” vizuālajam izskatam jābūt vienkāršam, bez spilgtām krāsām, vai pārmērīgiem efektiem. Spēlei sākoties pretinieki būs sarkanā, vai zilā krāsā.

2.3.4. Datorspēles optimizācija

Datorspēlei “HockeyShot” jābūt pietiekami optimizētai, lai to varētu palaist uz vidējais specifikācijas datoram, ar lielākajiem vizuālajiem iestatījumiem.

2.3.5. Spēles platforma

Datorspēlei “HockeyShot” pieejamā platforma būs tikai Windows operētājsistēmas.

2.4. Gala lietotāja raksturiezīmes

Hokeja spēlei “HockeyShot”, galvenā vecuma grupa ir sākot no 8 gadu vecuma, jo šajā vecumā bērni un jaunieši sāk izrādīt intersi par sportu un videospēlēm. Hokeja spēle būs piemērota gan iesācējiem, gan pieredzējušiem spēlētājiem, kuriem patīk gan ātra, dinamiska spēle, gan taktiskāka pieeja. Spēlei būs intuitīvas vadības kontroles, kas padarīs to viegli spēlējamu. Spēles vizuālais stils būs piemērots gan bērniem, gan pieaugušajiem, ļaujot izbaudīt reālistisku hokeju pieredzi ar dažādiem pieejamiem režīmiem un dažādām komandu konfigurācijām.

3. Izstrādes līdzekļu, rīku apraksts un izvēles pamatojums

Šajā nodaļa tiks aprakstītas izstrādes līdzekļu, rīku apraksts un izvēles pamatojums, lai zinātu kādus rīkus izstrādātājs izmantos šim projektam. Tiks aprakstīti rīki un pamatojums, kāpēc izstrādātājs izmanto šos rīkus datorspēlei "HockeyShot", kā arī aprakstīti tiks alternatīvie izstrādes līdzekļi.

3.1. Izvēlēto risinājumu līdzekļu un valodu apraksts

3.1.1. C#

Apraksts:

C# ir programmēšanas valoda, ko ir izstrādājusi Microsoft Corporation. Tā ir objektorientēta programmēšanas valoda, kas plaši izmantota dažādās programmēšanas jomās, tostarp spēļu izstrādē, lietojumprogrammu attīstībā un daudzas citās jomās. C# ir plaši izmantota valoda, ir pieejami daudzi resursi, dokumentācijas, bibliotēkas un pamācības, kas atvieglo to izmantošanu jaunajiem programmētājiem, gan arī pieredzējušiem programmētājiem.

Pamatojums:

Izstrādātājs izvēlējies C# valodu, jo Unity dzinis izmanto C# valodu, kā arī izstrādātājam ir iepriekšēja pieredze, lai izmantotu to datorspēles "HockeyShot" izstrādei. C# valoda ir līdzīga ar Java, kur izstrādātājs arī ir guvis iepriekšēju pieredzi, jo abas valodas ir objektorientētas valodas.

3.1.2. Unity

Apraksts:

Unity ir viens no vispopulārākajiem datorspēļu izstrādes rīkiem. Tā ir platforma, kur var veidot spēles datoriem, mobilajām ierīcēm, konsolēm, virtuālajai realitātei un citiem platformu veidiem. Unity ir ļoti labs rīks iesācēju izstrādātājiem, kā arī pietiekami spēcīgs rīks, lai to izmantotu arī pieredzējuši izstrādātāji.

Pamatojums:

Izstrādātājs izvēlējās Unity platformu, jo salīdzinot ar citiem datorspēļu izstrādes rīkiem nebija tik liela pieredze. Kā arī Unity tika izvēlēts, tā vieglās izmantojamības dēļ, kā arī Unity rīks ir ļoti labs datorspēļu rīks kurš sniedz visas nepieciešamās funkcijas, pat pieredzējušiem datorspēļu izstrādātājiem.

3.1.3. Visual Studio Code

Apraksts:

Visual Studio Code ir Microsoft izstrādāts pirmkoda redaktors operētājsistēmai Windows, Linux un MacOS. Funkcija ietver atbalstu atkļūdošanai, sintakses izcelšanai, automātiskā koda pabeigšanai un vēl citas funkcijas, kuras atvieglo darbu, kā paplašinājumus.

Pamatojums:

Izstrādātājs izvēlējies Visual Studio Code, jo tas ir viens no populārākajiem rakstīšanas rīkiem, kā arī šim rīkam ir ļoti draudzīga lietotāja saskarne. Var arī noinstalēt vajadzīgos paplašinājumus, kuri noder datorspēlei "HockeyShot"

3.1.4. GitHub

Apraksts:

GitHub ir plaši izmantota versiju kontroles sistēma, kas ļauj veidot programmatūru sadarbības veidā. Tā nodrošina platformu, kurā var glabāt un pārvaldīt kodu, GitHub piedāvā daudzās un dažādas funkcijas, piemēram, zaru pārvaldība, koda pārskatu un problēmu izsekošanu, kas veicina efektīvu koda izstrādi un sadarbību, kā arī ir iespēja atgriezties uz iepriekšējo versiju, ja izstrādes procesa tas ir nepieciešams.

Pamatojums:

Izstrādātājs izvēlējas GitHub, jo ir ļoti populāra kontroles sistēma, kur var nolikt savu projektu darbu, kā arī tur arī ir likti iepriekšējie projekta darbi.

3.1.5. Unity Service Relay

Apraksts:

Unity Service Relay ir Unity izstrādātāju izmantojams pakalpojums, kas ļauj spēļu izstrādātājiem izveidot tīkla savienojumu starp spēlētājiem, izmantojot Unity Multiplayer rīkus. Tas nodrošina uzticamu un drošu datu pārraidi starp klientiem vai starp klientu un serveri. Unity Relay ir augsti optimizēts, lai nodrošinātu labu savienojumu un augstu veiktspēju, kas ir būtiski reāllaika spēļu izstrādei. Šajā gadījuma sporta spēlei "HockeyShot"

Pamatojums:

Izstrādātājs izvēlējies izmantot Unity Service Relay, jo tas ir pilnība integrēts ar Unity dzinēju un tā Multiplayer risinājumiem. Tas nodrošina salīdzinoši vienkāršu konfigurāciju un API, kas ērti lietojama spēļu tīkla funkcionalitātes izstrādei. Tas atvieglo izstrādi un nerada nepieciešamību izmantot citus rīkus.

3.2. Iespējamo (alternatīvo) risinājuma līdzekļu un valodu apraksts

3.2.1. C++

Apraksts:

C++ ir programmēšanas valoda, kas ir plaši izmantota visur, kā piemērām, datorzinātnēs, programmatūras izstrādēs, operētājsistēmu veidošanā, spēļu izstrādē un daudzās citās jomās. Tā ir izstrādāta ka C valodas paplašinājums, piedāvājot objektorientētas programmēšanas iespējas, kā arī daudzas citu valodas funkcijas.

C++ valoda ir izmantojama gan maziem projektiem, gan arī lieliem projektiem, un tā ir populāra izvēle daudzas industrijās, kur tiek prasīta augsta veiktspēja

Pamatojums:

Izstrādājs iepriekš ir izmantojis C# spēļu izstrādei Unity diskdzinī tādējādi, lai saglabātu iepriekšējās zināšanas un, lai varētu ņemt piemēru no iepriekšējiem darbiem, netika izmantota C++ valoda. Kā arī, lai nebūtu laika zudums mazu nianšu apgūšanai.

3.2.2. Unreal Engine

Apraksts:

Unreal Engine ir viens no pasaulē vadošajiem spēļu izstrādes rīkiem, ko izstrādājusi Epic Games studija. Tas piedāvā visaptverošu platformu augstas kvalitātes spēļu veidošanai uz dažādām platformām, tostarp datoriem, spēļu konsolēm, un mobilajām ierīcēm. Unreal Engine ir populārs gan neatkarīgo izstrādātāju, gan lielo spēļu izstrādes studiju vidū. Tas ir izmantots daudzās augstākās klases spēlēs, piemēram, “Fortnite”, “Final Fantasy VII Remake” un citās.

Pamatojums:

Unreal Engine netika izmantots, jo tā apguve prasītu ievērojamu laika ieguldījumu. Lai gan izstrādātājs apzinājās, ka šis rīks varētu atvieglot izstrādes procesu, nepieciešamais apmācības laiks un ieguldījums neatmaksātu zaudēto laiku, kas būtu nepieciešams, lai apgūtu šo platformu.

3.2.3. Notepad++

Apraksts:

Notepad++ ir bezmaksas, universāls teksta redaktors, kas piedāvā plašu funkcionalitāti gan programmētājiem, gan ikdienas lietotājiem. Tas atbalsta audzas programmēšanas valodas, tostarp C#, Java, Javascript un C++, un nodrošina tādas funkcijas kā sintakses izcelšana, automātiskā koda pabeigšanā un pielāgojama lietotāja saskarne. Notepadd++ ir viegli lietojams, ātrs un efektīvs, tādēļ tas ir iecienīts lietotāju vidū. Tā pielāgojamība un veiktspēja padara to par lielisku rīku dažādām teksta rediģēšanas vajadzībām.

Pamatojums:

Lai gan Notepad++ ir noderīgs redaktors, izstrādātājs to neizmantoja, jo visas nepieciešamās spraudņu konfigurācijas un darba vide jau bija uzstādīta Visual Studio Code redaktorā, kas ļāva uzsākt darbu nekavējoties.

3.2.4. SourceForge

Apraksts:

SourceForge ir tiešsaistes platforma, kas piedāvā daudzveidīgus atvērtā koda projektus un programmatūras resursus. Tā aptver plašu jomu spektru, tostarp spēļu un lietojumprogrammu izstrādi. SourceForge ir viens no vecākajiem un visilgāk darbojošajiem atvērta koda programmatūras izplatīšanas portāliem. Platforma ļauj izstrādātājiem publisko projektus, dalīties ar kodu, izsekot versiju vēsturi un iegūt atsauksmes no lietotājiem.

Pamatojums:

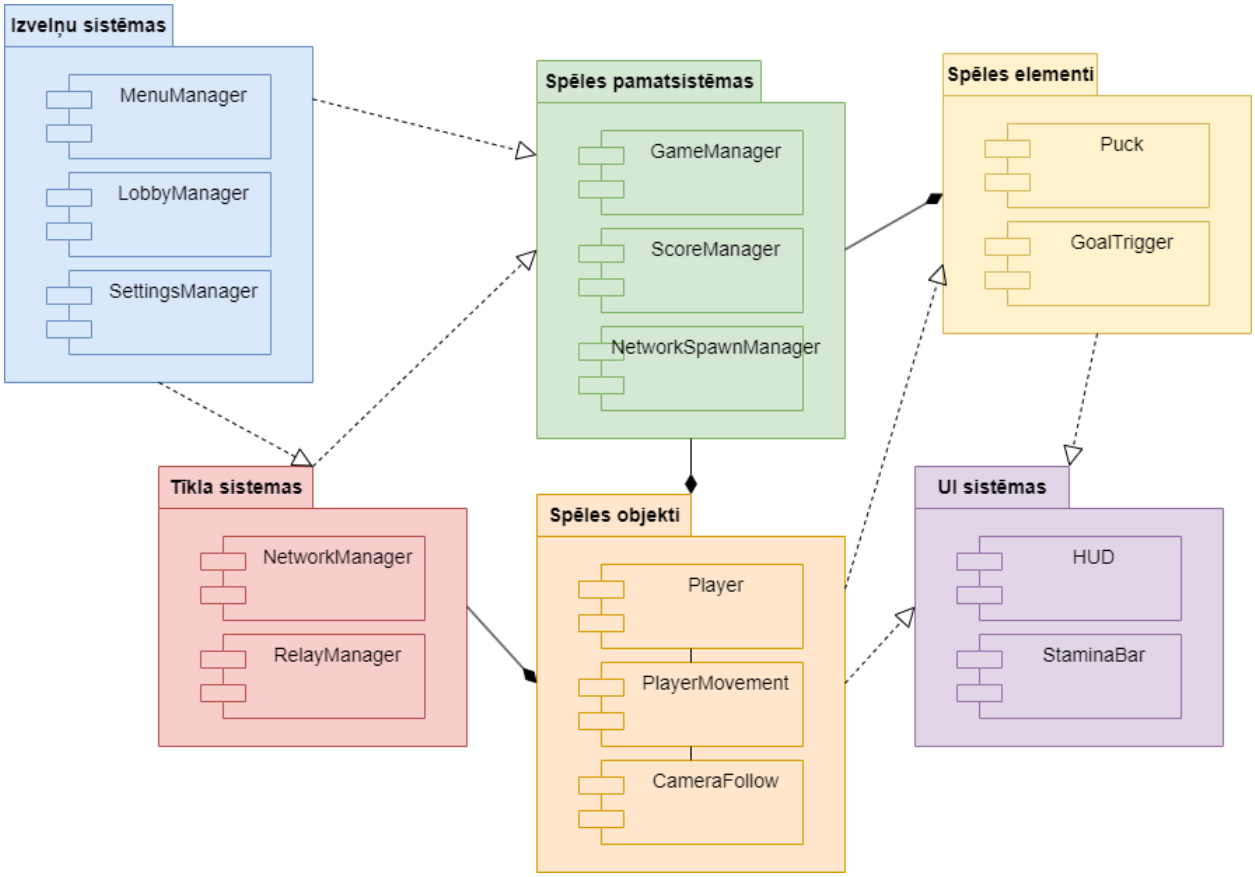
Izstrādātājs izvelējās neizmantot SourceForge, jo visu programmēšanas apmācību un pieredzes laiku ir strādājis ar GitHub. Izvēloties palikt pie pazīstamas platformas, tika izvairīts no papildus mācību laika un iespējamām nesaskaņām ar projekta versionēšānu.

4. Sistēmas modelēšana un projektēšana

Šajā nodaļā tiks parādīti dažādu diagrammu veidi, lai vizualizētu datorspēli "HockeyShot" sistēmu struktūru, darbības plūsmas.

4.1.1. Sistēmas struktūras modelis

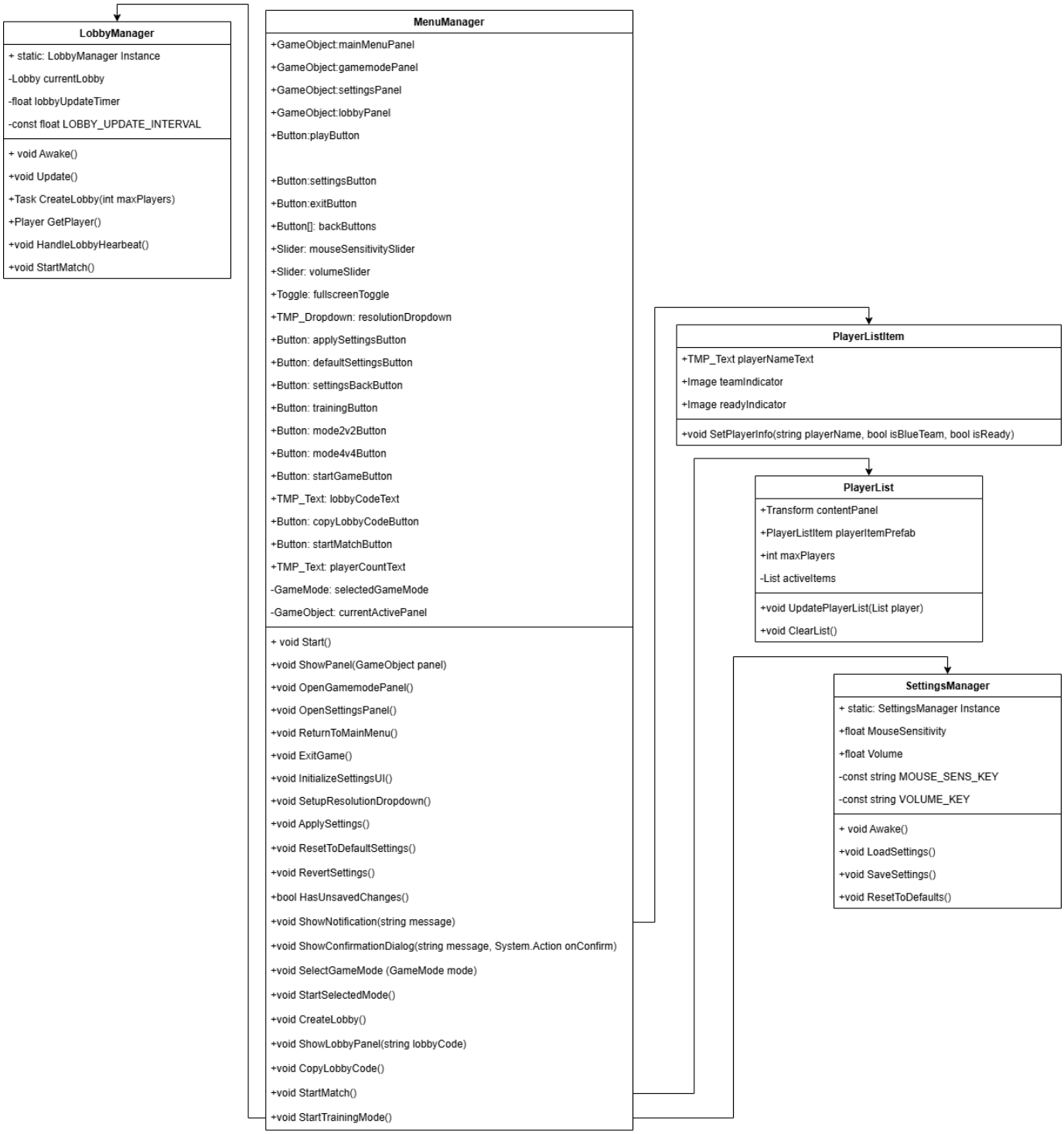
Šaja sadaļā ir parādīts kā sistēmas struktūras modelis strādā ar komponentes diagrammas palīdzību (Skatīt 1. attēlu)



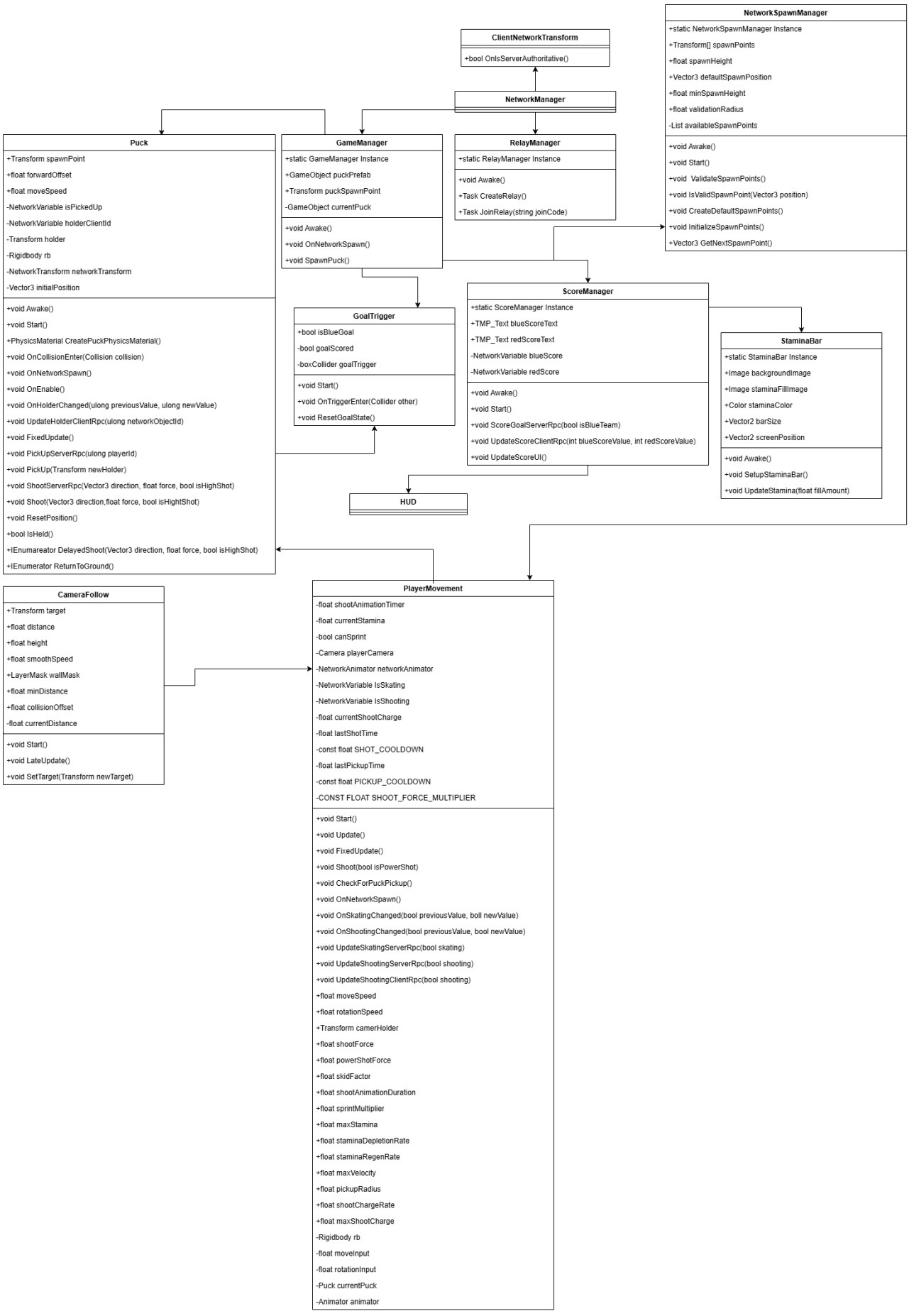
1. attēls Sistēmas struktūras modelis

4.1.2. Klašu diagramma

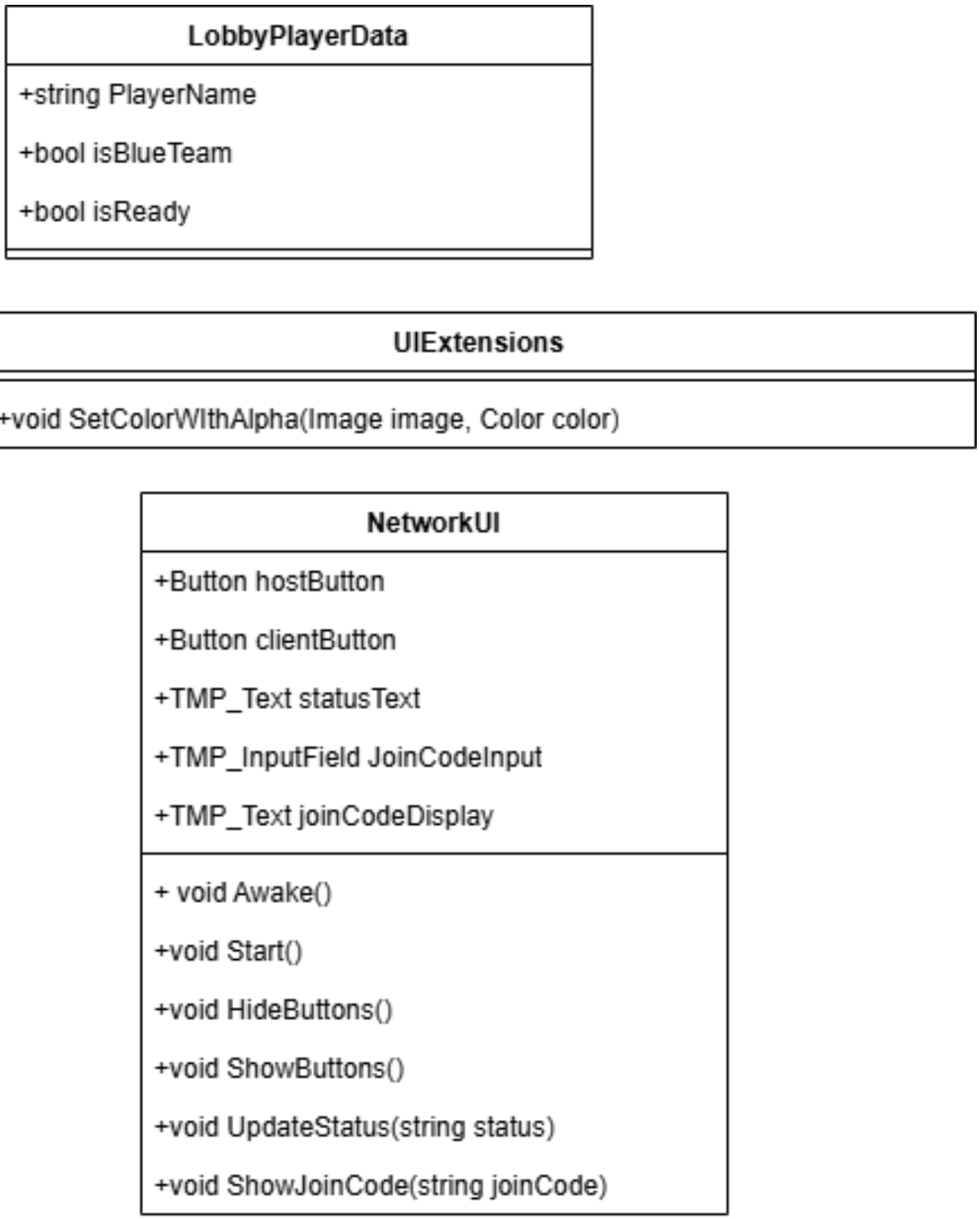
Klašu diagramma ir UML diagrammas veids, kas tiek izmantots, lai vizuāli attēlotu objektorientētas sistēmas strukturālo dizainu. Klašu diagramma palīdz izprast projekta galveno komponentu struktūru, to atribūtus, metodes un attiecības starp klasēm. Tā nodrošina skaidru pārskatu par sistēmas arhitektūru. Skatīt (2. ,3., 4. attēlu)



1. attēls klases diagramma menedžeru daļa



2. attēls klases diagramma spēles atkarīgie

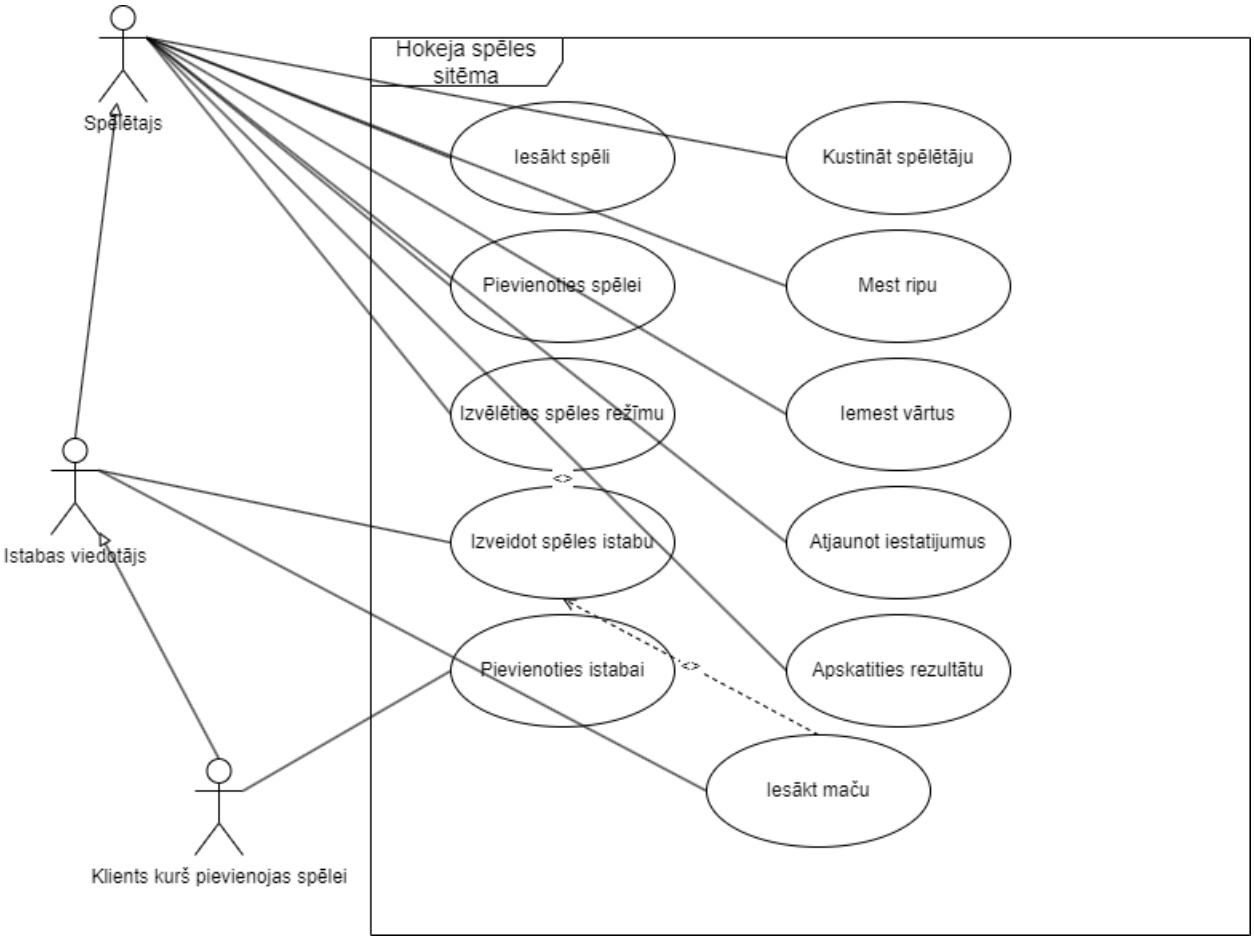


3. klases diagramma individuālie skripti

4.2. Funkcionālais un dinamiskais sistēmas modelis

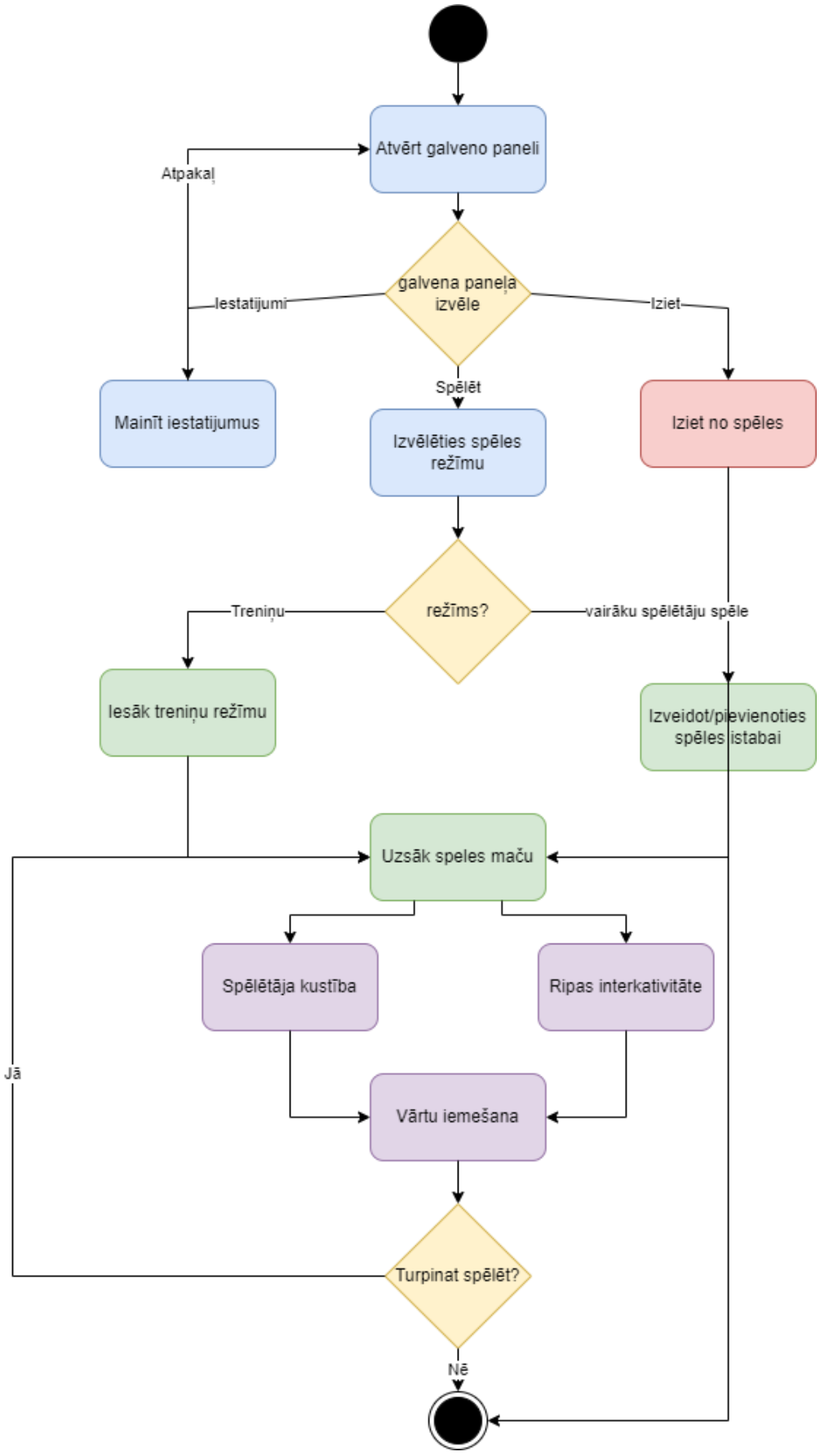
4.2.1. Lietojumgadījumu diagramma

Lietojumgadījuma diagramma ir UML diagrammas veids, kas tiek izmantots, lai attēlotu sistēmas funkcionalitāti no lietotāja perspektīvas. Tā parāda, kā dažādi lietotāji mijiedarbojas ar sistēmu un kādas funkcijas sistēma piedāvā. Skatīt (5. attēlu)



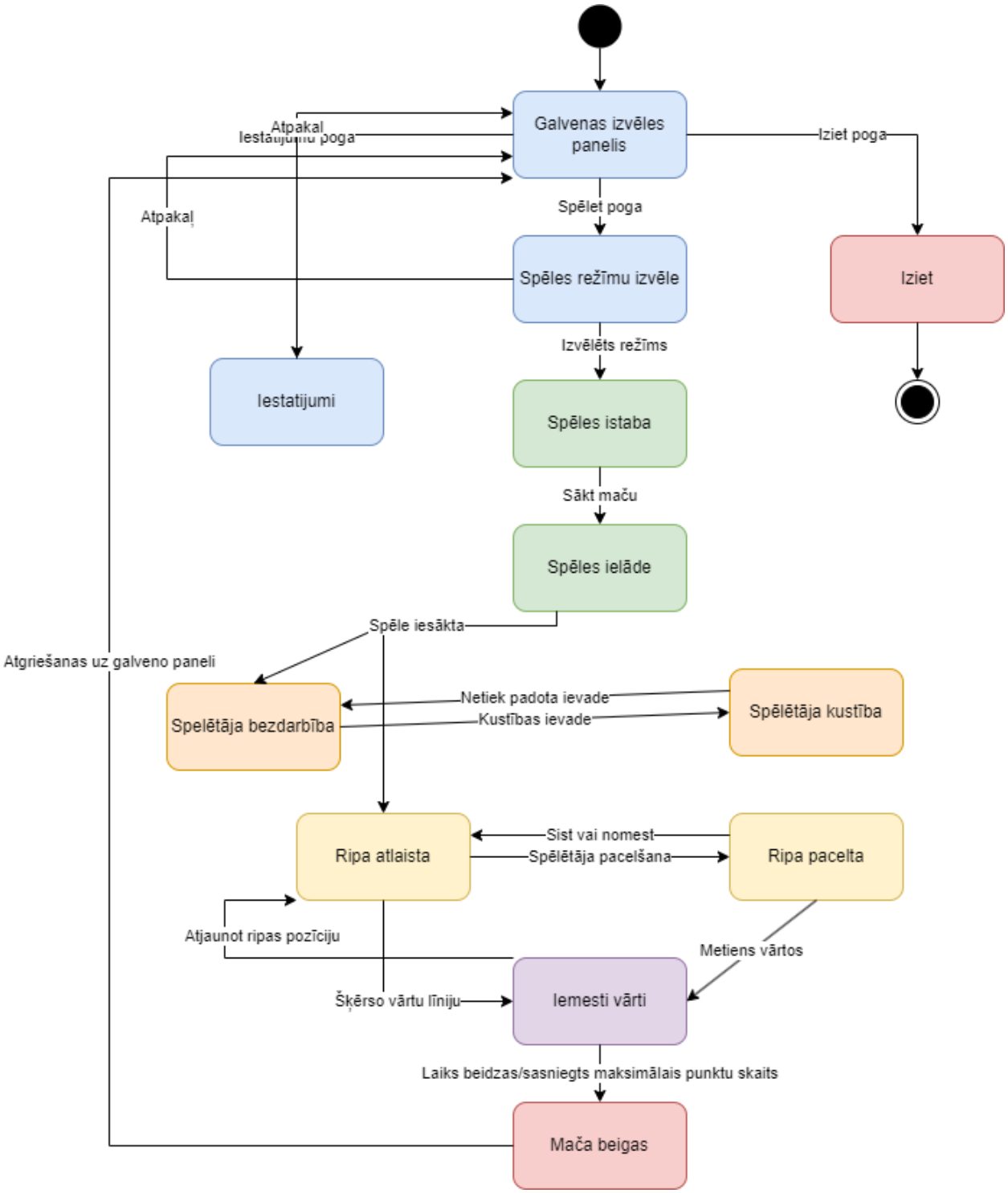
5.attēls Lietojumgadijuma diagramma

4.2.2. Aktivitāšu diagramma



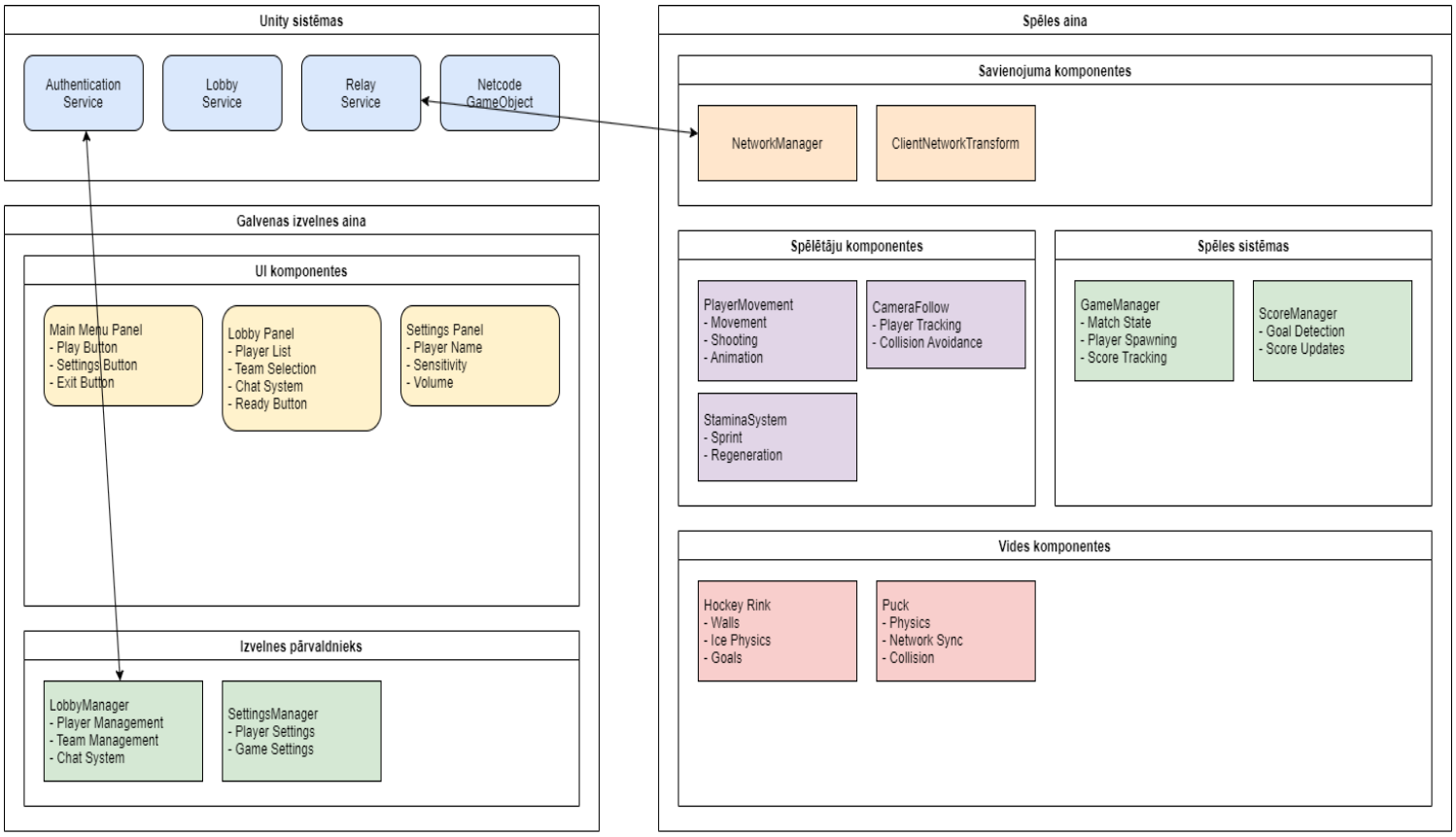
6.attēls Aktivitāšu diagramma

4.2.3. Stāvokļu diagramma



7.attēls Stāvokļu diagramma

4.3. Datu struktūru apraksts

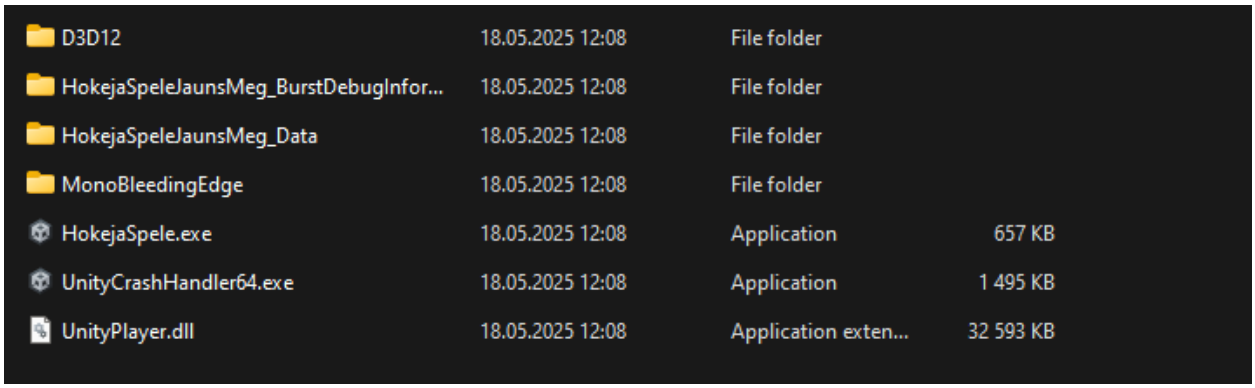


8. attēls datu struktūru diagramma

5. Lietotāju ceļvedis

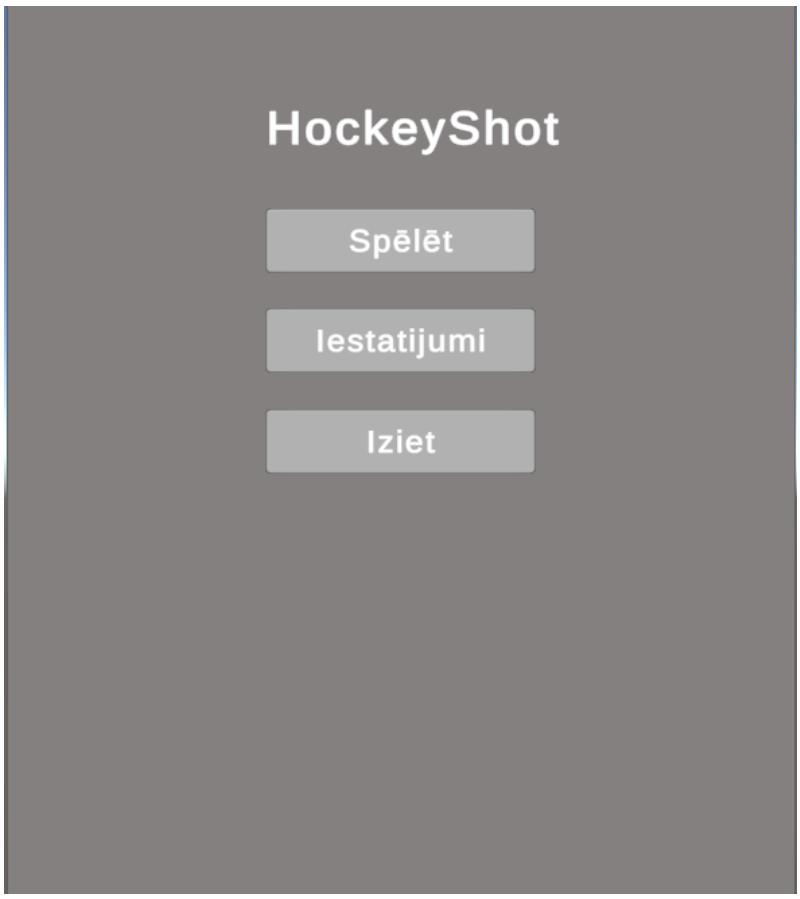
5.1. Spēles atvēršana

- Sākotnēji lejupieladējiet spēli no Github projekta “HokejaSpele”
- Pēc tam atveriem failu
- Un noklikšķiniet 2 reizes uz “HokejaSpele” exe (Skatīt 9. attēlu)



9. attēls spēles atvēršana

- Rezultātā atveras šāds logs, kur parādās sākuma aina ar 3 pogām – spēlēt, iestatījumi, iziet(Skatīt 10. attēlu)



10. attēls atvēršanas rezultāts

5.2. Spēles galvenā izvēlne

- Kad ir atvērta spēle parādās 3 opcijas – var sākt spēli, var ieet iestatijumos un var iziet ārā no spēles.
- Iestatijumos var veikt 5 izmaiņas – Var samainīt kursora ātrumu spēles paneļos, mainīt spēles skaļumu, samainīt izšķirtspēju, samainīt spēlētāja vārdu, un nomainīt spēles ekrāna veidu.(Skatīt 11. attēlu)

11.attēls iestatījumu panelis

5.3. Spēles izvēlnes aina

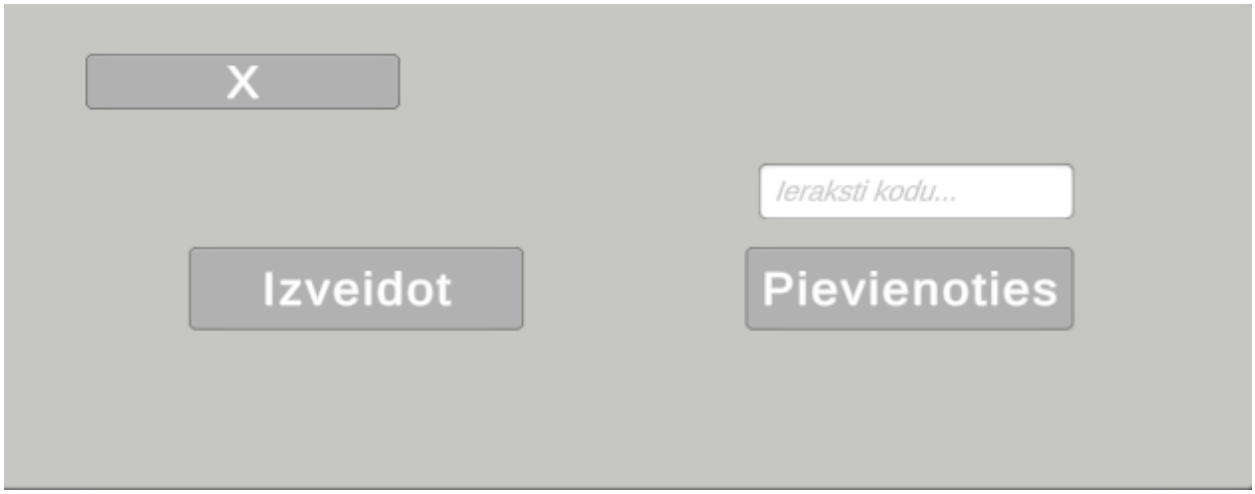
- Atrodoties galvenā ainā nospiežot pogu “Spēlēt”, jums atvērsies spēles izvēles aina un tā izskatās šādi (Skatīt 12. attēlu)

12.attēls spēles izvēles aina

- Pirms spēles sākšanas, būs jaizvēlas viena no 3 izvēlnēm,
    - Treniņu režīms – Spēlet vienam pašam spēles laukumā ar spēju iemācīties kontroles
    - Tiešsaistes režīms – Spēlēt komandās pa divi ar citiem spēlētājiem
- Ja vēlaties atgriezties atpakaļ, to var izdarīt ar pogu atpakaļ

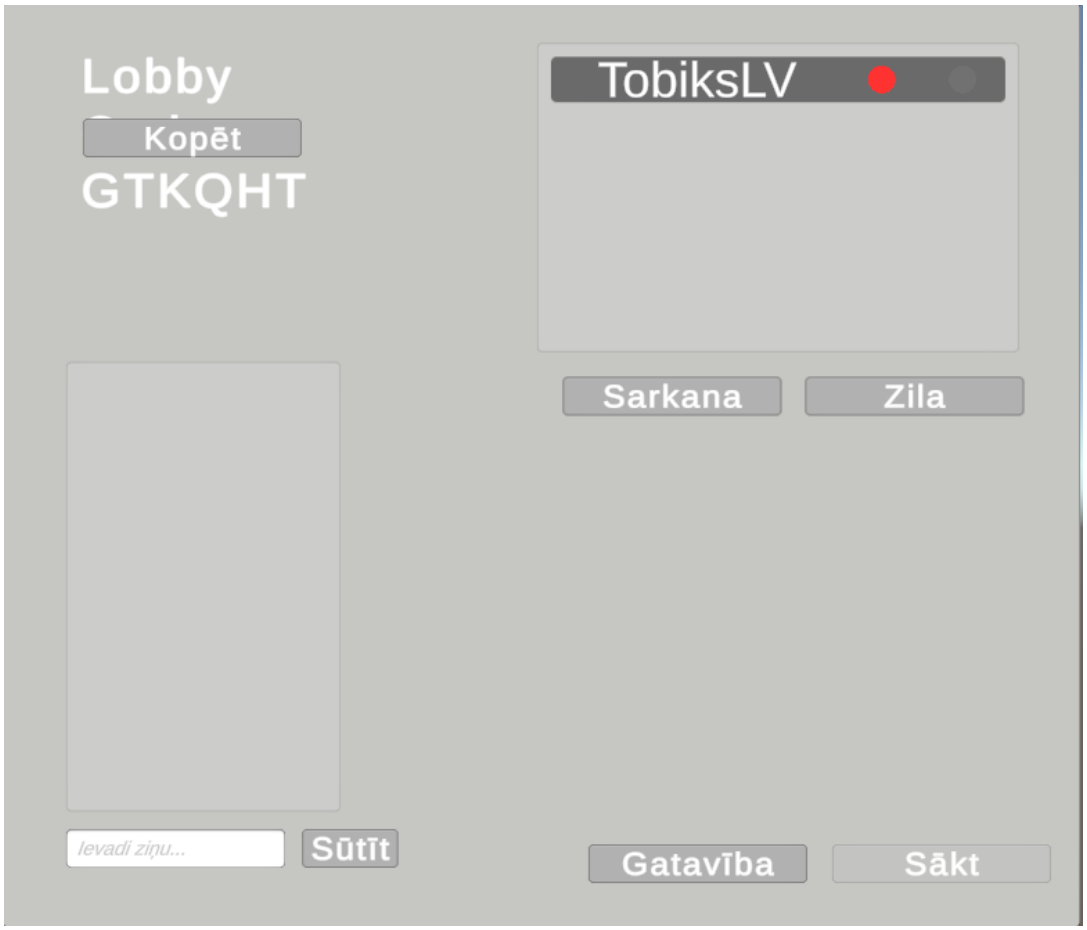
3.1. Spēles telpas aina

- Kad ir izvēlēts tiešsaistes režīms tad spēlētājam tiks piedāvāta iespēja pievienoties jau izveidotajai spēlei, ar kodu, vai izveidot savu spēles telpu. (Skatīt 13. attēlu)



13. attēls istabas pievienošanas vai izveidošanas

- Pēc spēles istabas izveides, vai pievienošanas atveras jauna aina kura ir pirmspēles istaba ar dažādām iespējām tajā – Sarunāties ar citiem spēlētājiem caur izveidoto čatu, samainīt komandu, mainīt savu gatavību, redzēt spēles kodu un to kopēt un nosūtīt citiem. (Skatīt 14. attēlu)



14. attēls istabas telpas aina

5.4. Spēles aina

- Visiem spēlētājiem pievienojoties un sagatavojoties, vai pievienojoties treniņu režīmā tiek pievienots spēlētājs spēles ainā(Skatīt 15. attēlu)

15. attēls spēles aina

- Spēles ainā lietotājs ir spējīgs redzēt laika, atskaiti, redzēt komandu iemestos vārtus.
- Spēles ainā spēlētājs var parvietoties ar kustību taustiņu palīdzību w/s, lai veiktu kustību uz priekšu a/d, lai veiktu rotāciju pa kreisi vai labi. Kā arī ir iespēja skriet izmantojot shift taustiņu un space taustiņu, lai apturētu spēlētāju ātri. Kā arī alt taustiņu, lai kustinātu kameru un peles kreiso klikšķi izmanto mešanai.
- Ar "esc" taustiņu var ieslēgt pauzes ekrānu kurā ir iespējams iziet no spēles telpas, mainīt iestatijumus, iziet no pašas spēles. (Skatīt 16. attēlu)

16. attēls pauzes ekrāns spēles ainā

5.5. Spēles beigas

- Spēlei beidzoties, laikam notekot, spēles ainā tiek parādīts panelis ar informāciju par kura komanda uzvarējusi, kā arī dod iespeju atgriezties uz spēles galveno ainu kur var uzsākt jaunu spēli, vai arī iziet no spēles(Skatīt 17. attēlu)

17. attēls uzvaras paziņojums

5.6. Treniņu aina

- Spēles režīma izvelnē izvēloties treniņu režīmu spēlētājs tiek ievietots spēles ainā ar iespēju izjust spēles kontroles, kā arī ir redzamas kontroles saraksts, lai lietotājs tās varētu apgūt.

6. Testēšanas dokumentācija

Šajā nodaļa tiks aprakstīta Datorspēles "HockeyShot" testēšana, kādas testēšanas metodes tiks izmantotas un testpiemēru rezultāti.

6.1. Izvēlētas testēšanas metodes, rīku apraksts un pamatojums

Projekta testēšanai tiek izmantota manuāla testēšanas metode, dokumentējot rezultātus Excel formātā. Šāda pieeja tika izvēlēta, balstoties uz iepriekšējo pieredzi ar līdzīgu testēšanas dokumentācijas izvedi skolas apmācības procesā.

Excel rīks tiek izvēlēts, kā vispiemērotākais izstrādātāja projektā, jo tas nodrošina ērtu tabulas veidošanu, piedāva datu organizēšanu, starp testējamajām prasībām, kā arī Izstrādātājs salīdzinoši bieži ir izmantojis Excel rīku un tajā ļoti labi orientējas.

Lai, veiktu testēšanu izstrādātajs to veic divas pieejas:

1. Black box testēšanas pieejā:
    - Pārbauda sistēmu no lietotāja perspektīvas
    - Fokusējās uz funkcionalitātes darbību
    - Testē lietotāja saskarni un datu plūsmu
2. White box testēšanas pieejā:
    - Pārbauda ieksējo programmas struktūru
    - Analizē koda darbību
    - Nodrošina tehnisko kvalitāti visu projektu funkcijās

Izmantojot abas šīs pieejas, tas ļauj nodrošināt visa projektā tehnisko kvalitāti, gan lietotājam funkcionalitāti no viņu puses.

6.2. Testpiemēru kopa

1. tabula

Datorspēle "Hokejs"	
Identifikātoru atšifrējums:	
Piemērs: TP.IZV.01	
PR	Prasība
TP	Testpiemērs
Piederība modulim:	
SAK	Sākums
IES	Iestatījumi
IZV	Izvēle
LOB	Lobby
VAD	Vadība
MEH	Mehānika
TIK	Tīkls
SPE	Spēle
CAT	Čats
KOM	Komanda
REZ	Režīmi

2. tabula

Prasības ID	Prasība
Sākums	
PR.SAK.01	Hokeja spēles aizvēršana
PR.SAK.02	Iesatījumu paneļa atvēršana
PR.SAK.03	Spēles režīmu paneļa atvēršana
PR.SAK.04	Par spēli paneļa atvēršana
Iestatijumi	
PR.IES.01	Iestatijuma paneļa aizvēršana
PR.IES.02	Spēles skaļuma maiņa
PR.IES.03	Peles jūtīguma maiņa
PR.IES.04	Spēlētāja vārda maiņa
PR.IES.05	Iestatijumu apstiprināšana
PR.IES.06	Iestatijumu attiestatīšana
Spēles režīmi	
PR.REZ.01	Treniņu režīms
PR.REZ.02	Spēlēt 2 pret 2
PR.REZ.03	Spēlēt 4 pret 4
Lobby sistēma	
PR.LOB.01	Lobby izveide
PR.LOB.02	Pievienošanas lobby
PR.LOB.03	Komandas izvēle
PR.LOB.04	Gatavības statusa maiņa
PR.LOB.05	Čata izmantošana
PR.LOB.06	Lobby koda kopēšana
Spēlētāja vadība	
PR.VAD.01	Spēlētāja kustības (WASD)
PR.VAD.02	Skriešana(Shift)
PR.VAD.03	Ripas sistiens(peles kreisais klikšķis)
PR.VAD.04	Enerģijas sistēma
Tīkla funkcija	
PR.TIK.01	Spēlētāju pozīciju sinhronizācija
PR.TIK.02	Rezultātu sinhronizācija
PR.TIK.03	Ripas sinhronizācija
PR.TIK.04	Spēlētāja atvienošanas apstrāde
Spēle	
PR.SPE.01	Vārtu gūšanas sistēma
PR.SPE.02	Rezultātu tabula
PR.SPE.03	Spēles laika kontrole
PR.SPE.04	Spelētāju sadursmes

3. tabula

Testpiemēra ID	Testpiemēra nosaukums	Testpiemēra izpildes nosacījumi	Testpiemēra apraksts	Testpiemēra izpildes soļi	Testpiemēra ievades dati	Sagaidāmais rezultāts	Prasības ID
Black Box							
TP.SAK.01	Spēles poga	Spēle palaista	Pārbaudīt Spēlēt pogu	1) Nospiesta Spēlēt poga sākuma izvēlnē	Peles kreisais klikšķis	Atveras režīmu izvēlne	PR.SAK.01
TP.SAK.02	Iestatījumu poga	Spēle palaista	Pārbaudīt Iestatijumu pogu	1) Nospiesta Iestatijuma poga	Peles kreisais klikšķis	Atveras iestatījumu izvēlne	PR.SAK.02
TP.SAK.03	Izejas poga	Spēle palaista	Pārbaudīt Iziet pogas darbību	1) Nospiest Iziet pogu	Peles kreisais klikšķis	Spēle aizveras	PR.SAK.03
TP.IES.01	Skaņas regulēšana	Atvērti iestatījumi	Pārbaudīt skaņas slaidera darbību	1) Pārvietot skaļuma slaideri	Slaidera kustību	Mainās spēles skaļums	PR.IES.01
TP.IES.02	Vārda maiņa	Atvērti iestatījumi	Pārbaudīt vārda ievadi	1) Ievadīt jaunu vārdu	Teksts "Player1"	Vārds nomainīts	PR.IES.02
TP.IES.03	Iestatījumu saglabāšana	Mainīti iestatījumi	Pārbaudīt saglabāšanu	1) Nospiesta poga "Saglabāt"	Peles kreisais klikšķis	Iestatījumi saglabāti	PR.IES.03
TP.REZ.01	Treniņu režīms	Atvērta režīmu izvēlne	Pārbaudīt Training režīmu	1) Nospiesta poga Sākt zem treniņu režīma	Peles kreisais klikšķis	Sākas spēle treniņa režīmā	PR.REZ.01
TP.REZ.02	2v2 režīms	Atvērta režīmu izvēlne	Pārbaudīt 2v2 režīmu	1) Nospiesta poga Sākt zem 2v2 režīma	Peles kreisais klikšķis	Atveras lobbu izveide	PR.REZ.02
TP.LOB.01	Lobby izveide	Izvēlēts režīms	Pārbauda Izveidot pogas funkcionalitāti	1) Nospiesta poga "Izveidot" lobby izveides/pievienošanas paneli	Peles kreisais klikšķis	Izveidots jauns lobby	PR.LOB.01
TP.LOB.02	Lobby pievienošanās	Ievadīts kods	Pārbauda pievienoties pogas funkcionalitāti	1) Ievadīts Lobby kods ievades laukā 2) Nospiesta poga "pievienoties" lobby izveides/pievienošanas paneli	Lobby kods	Pievienošanas lobby telpai	PR.LOB.02
TP.LOB.03	Komandas izvēle	Atvērta lobby telpa	Pārbauda komandu pogu funkcionalitāti	1) Izvēlēties komandas krāsu	Peles kreisais klikšķis	Nomainīta komanda uz attiecīgi izvēlēto zilo vai sarkano	PR.LOB.03

TP.VAD.01	Kustība ar spēlētāju	Spēle aktīva, kāda no režīmiem	Pārbauda WASD vadību	1) Nospiež W	W taustiņš	Spēlētājs kustas uz priekšu	PR.VAD.01
TP.VAD.02	Skriešana	Spēle aktīva, kāda no režīmiem	Pārbauda Skriešanas vadību	1) Nospiež Shift + W	Shift + W	Spēlētājs kustas uz priekšu ātrāk	PR.VAD.02
TP.VAD.03	Ripas sitiens	Pie ripas	Pārbauda sitienu	1) Kreisas klikšķis	Peles kreisais klikšķis	Ripa tiek sista	PR.VAD.03
TP.SPE.01	Vārtu gūšana	Spēle aktīva, kāda no režīmiem	Pārbauda vārtu gūšanu	1) Iemet ripu vārtos	Ripas trāpijums	Punkts pieskaitīts	PR.SPE.01
TP.SPE.02	Rezultātu tabula	Gūti vārti	Pārbauda rezultātu	1) Iegūti vārti	Vārtu iegūšana	Rezultāts mainās	PR.SPE.02
TP.SPE.03	Spēles laiks	Spēle sākta	Pārbauda laiku	1) Sākt spēli	Spēles sākums	Laiks skaitās	PR.SPE.03
TP.SPE.04	Spēles beigas	Laiks beidzas	Pārbauda beigas	1) Sagaidīt beigas	Laika notecēšana	Spēle beidzas	PR.SPE.04
TP.TIK.01	Spēlētāju redzamība	Spēle tīklā	Pārbauda sinhronizāciju	1) Pievienoties spēlei	Savienojums	Visi spēlētāji redzami	PR.TIK.01
TP.TIK.02	Rezultātu sync	Gūti vārti	Pārbauda sinhronizāciju	1) Gūti vārti	Vārtu gūšana	Rezultāts visiem spēlētājiem vienāds	PR.TIK.02
TP.TIK.03	Ripas sync	Sitiens	Pārbauda ripas pozīciju	1) Sitiens pa ripu	Sitiens	Ripas pozīcija visiem vienāda	PR.TIK.03
White Box							

TP.AUT.01	Unity Services autentifikācija	Unity projekts palaists, nav autorizēts	Unity Services autentifikācijas process	1) Izsaukt UnityServices.InitializeAsync()	ServicesInitalization() izsaukums	1) UnityServices.InitalizeASync() izpildās 2) AuthenticationService.Instance tiek inicializēts 3) SignInAnonymouslyASync() izpidlas veiksmīgi 4) AuthenticationService.Instance.IsSignedIn == true 5) Tiek saglabāts PlayerId	PR.DRO.01
TP.REZ.01	2v2 režīma pārbaude	Režīmu izvēlne atvērta	Pārslēdz režīmu uz 2v2	1) Izsauc SelectGameMode(Mode2v2)	GameMode.Mode2v2	1) selectedGameMode == Mode2v2 2) maxPlayers = 4	PR.REZ.01
TP.REZ.01	4v4 režīma pārbaude	Režīmu izvēlne atvērta	Pārslēdz režīmu uz 4v4	1) Izsauc SelectGameMode(Mode4v4)	GameMode.Mode4v4	1) selectedGameMode == Mode4v4 2) maxPlayers = 8	PR.REZ.02
TP.LOB.01	Lobby izveides process	Unity Service inicializēts, autorizēts	Jauna lobby izveides process	1) Izsauc CreateLobby(4, options)	maxPlayers=4, gamemode="2v2"	1)LobbyService.Instance.CreateLobbyAsync() izpildās 2) Tiek izveidots jauns lobby objekts 3) LobbyManager.currentLobby tiek iestatīts 4) Tiek ģenerēts unikāls lobby kods 5) Lobby dati tiek sinhronizēti ar serveri	PR.LOB.01
TP.KOM.01	Komandas maiņa	Lobby aktīvs	Pārslēgt komandu	1) Izsauc SetPlayerTeam()	team="Blue"	1) playerTeams[id]=='Blue' 2) UpdateLobbyData() izsaukts	PR.KOM.01

TP.KOM.02	Komandas limits	Komanda pilna	Mēģināt pievienoties pilnai komandai	1) Izsauc SetPlayerTem()	team="Blue", count=max	1) Returns false 2) Komanda netiek mainīta	PR.KOM.02
TP.PLM.01	Spēlētāju kustību apstrāde	PlayerMovement skripts aktīvs	Spēlētāja kustību vadības loģika	1) Simulēt Input.GetAxisRaw("Horizontal")	Input Vector(1,0,0)	1) HandleMovementInput() metode saņem ievadi 2) Rigidbody.velocity tiek atjaunināts 3) transform.rotation tiek interpolēta 4) Tiek izsaukts UpdateSkatingServerRpc() 5) NetworkVariable<bool> isSkating tiek atjaunināts 6) Animator parametri tiek atjaunināti	PR.VAD.01
TP.TIK.01	Attalinātās procedūras izsaukums	NetworkObject aktīvs	Pārbaudīt RPC	1) Izsaukt ServerRpc()	networkObject!=null	1) ServerRpc nostrādā 2) Dati tiek sinhronizēti	PR.TIK.01
TP.TIK.02	Klienta autoritāte	Client authority ieslēgts	Pārbaudīt tiesības	1) Pārvietot objektu	isOwner=true	1) Kūstība atļauta 2) Pozīcijas sync	PR.TIK.02

TP.NET.01	Pozīciju sinhronizācija	NetworkManager aktīvs, spēlētājs pievienojies	Spēlētāju pozīciju sinhronizācija tīklā	1) Lokālā spēlētāja kustība	Input.GetAxisRaw() vērtības	izmaiņas 2) NetworkVariable tiek atjaunināts 3) Pozīcija tiek sinhronizēta visiem klientiem 4) Interpolācija tiek piemērota 5) Latency < 100ms	PR.TIK.01
TP.VAR.01	Vārtu gūšanas loģika	Spēle aktīva, ripa kustībā	Vārtu gūšanas process	1) Ripa šķērso vārtu trigeri	OnTriggerEnter(Collider)	1) GoalTrigger.OnTriggerEnter() aktivizējas 2) Pārbauda ripas tagu 3) ScoreManager.ScoreGoalServerRpc() tiek izsaukts 4) NetworkVariable<int> score tiek atjaunināts 5) UI elementi tiek atjaunināti 6) Notifikācija tiek parādīta	PR.SPE.01
TP.PCK.01	Ripas pacelšana	Spēlētājs pieet pie ripas	Pārbauda OnTriggerEnter	1) Spēlētājs saskaras ar ripu	collision.tag == "Puck"	1) currentPuck != null 2) ripa.transform.parent == player.transform 3) ripa.GetComponent<Rigidbody>().isKinematic == true	PR.VAD.03
TP.PCK.02	Ripas sitiens	Spēlētājam ir ripa	Pārbauda sitiena mehāniku	1) Nospiest kreiso peles pogu	Input.GetMouseButtonDown(0)	1) ripa.GetComponent<RigidBody>.velocity == transform.forward * shootForce 2) ripa.transform.parent == null 3) currentPuck == null	PR.VAD.03

TP.SPM.01	Ripas izveide	Spēle sākas	Pārbauda SpawnPuck metodi	1) Izsaukt OnNetworkSpawn()	isServer=true	1) currentPuck !=null 2) NetworkObject.IsSpawned == true 3) transform.position == puckSpawnPoint.position	PR.SPE.01
TP.SPM.02	Dubulta ripa	Eksistē ripa	Pārbauda dubultu izveidi	1) Mēģināt izveidot jaunu ripu	existingPuck !=null	1) Vecā ripa iznīcināta 2) Jauna ripa izveidota 3) NetworkObject.IsSpawned == true	PR.SPE.01
TP.STA.01	Enerģijas limits	Enerģija pilna	Pārbauda maksimumu	1) Atjaunot enerģiju	currentStamina>maxStamina	1) currentStamina == maxStamina 2) canSprint == true	PR.VAD.04
TP.STA.02	Enerģijas izsīkums	Skriešana aktīva	Pārbauda minimumu	1) Turēt sprintu 10s	currentStamina<0	1) currentStamina == 0 2) canSprint == false	PR.VAD.04

TP.STA.03	Atjaunošanas ātrums	Enerģija zema	Pārbauda regenerāciju	1) Gaidīt 5s	sprint == false	1) currentStamina += staminaRegenRate * 5 2) stamina < maxStamina	PR.VAD.04
TP.RPF.01	Ripas ātrums	Spēcīgs sitiens	Pārbauda max ātrumu ripai	1) Sist ar max spēku	force = maxForce	1) rb.velocity.magnitude <= maxSpeed 2) Ātrums samazinās	PR.VAD.03
TP.RPF.02	Ripas berzēšana	Ripa slīd	Pārbauda berzi	1) Piemēro spēku	time > 2s	1) rb.velocity.magnitude samazinās 2) Apstājās pēc ~3s	PR.VAD.03
TP.RPF.03	Vārtu kolīzija	Ripa ieiet vārtos	Pārbauda vārtu trigger	1) Ripa šķērso vārtus	triggerEnter == true	1) OnTriggerEnter izsaukts 2) Punkts pieskaitīts	PR.SPE.01

TP.TNK.01	Host migrācija	Host atvienojas	Pārbauda pārslēgšanos	1) Host atvienojas	NetworkManager.Disconnect	1) Jauns host izvēlēts 2) Spēle turpinās	PR.TIK.04
TP.TNK.02	Klienta sync	Augsta latency	Pārbauda interpolāciju	1) Simulē lagu	latency > 100ms	1) Pozīcijas tiek interpolētas 2) Kustība gluda	PR.TIK.01
TP.TNK.03	Ripas autoritāte	Sitiens tīklā	Pārbauda sinhronizāciju	1) Klienta sitiens	Input.GetMouseButton(0)	1) Ripas pozīcija sync visiem 2) Fizika strādā korekti	PR.TIK.03
TP.CAT.01	Ziņas garums	Gara ziņa	Pārbauda limitus	1) Sūta ziņu > 100 simbolim	message.Length >100	1) Ziņa saīsināta 2) Rāda ... Beigās	PR.LOB.05

TP.CAT.02	Formātēšana	HTML tags	Pārbauda drošību	1) Sūta ziņu ar tagiem	"Test"	1) Tags escaped 2) Rāda kā tekstu	PR.LOB.05
TP.CAT.03	Vēstures limits	>50 ziņas	Pārbauda bufera izmēru	1) Sūtīt 51 ziņu	messages.Count > 50	1) Vecāka ziņa dzēsta 2) Count == 50	PR.LOB.05
TP.IES.01	Skaņas saglabāšana	Iestatījumi atvērti	Mainīt skaņu	1) SetVolume(0.5f)	volume=0.5f	1) PlayerPrefs.SetFloat() izsaukts 2) AudioListener.volume==0.5f	PR.IES.01
TP.IES.02	Vārda saglabāšana	Iestatījumi atvērti	Mainīt vārdu	1) SetPlayerName("Test")	name="Test"	1) PlayerName=='Test' 2) PlayerPrefs atjaunināts	PR.IES.02

TP.IES.03	Nekorektas vērtības	Invalid input	Pārbauda validāciju	1) volume = -1	volume < 0	1) volume = 0 2) Kļūda ielogota	PR.IES.02
TP.IES.04	Datu persistance(saglabāšana)	Restartē spēli	Pārbauda saglabāšanu	1) Maina iestatījumus 2) Restartē spēli	Application.Quit()	1) Iestatījumi saglabāti 2) Pareizi ielādēti	PR.IES.06
TP.ANM.01	Skriešanas animācija	Spēlētājs kustas	Pārbauda blend tree	1) WASD+Shift	velocity > walkSpeed	1) Animator.Speed == 1 2) Pareizs blend	PR.VAD.02
TP.ANM.02	Sitiena animācija	Sitiena ar ripu	Pārbauda trigger	1) Nospiests Fire1	GetMouseButtonDown(0)	1) Animator trigger izsaukts 2) Animācija atskaņojas	PR.VAD.03

TP.MEH.01	Ripas fizika	Spēle aktīva	Ripas kustība	1) addForce(Vector3.right)	force=10f	1) rb.velocity atjaunojas 2) Pareiza fizika	PR.MEH.01
TP.MEH.02	Spēlētāja sadursme	Spēle aktīva	Pārbauda sadursmi	1) OnCollisionEnter()	collision.tag="Player"	1) Pareizs atlēciens 2) Ātrums mainās	PR.MEH.02

6.3. Testēšanas žurnāls

4. tabula

Testēšanas ID	Datums	Testpiemēra ID	Testpiemera nosaukums	Testētājs	Statuss	Kļūdas ziņojums	Klūdas ziņojuma Nr.
Black Box							
TZ.B.01	06.05.2025	TP.SAK.01	Sākt spēles pogas pārbaude	Tomijs Būmerts	Veiksmīgs		
TZ.B.02	06.05.2025	TP.SAK.02	Iestatījumi pogas pārbaude	Tomijs Būmerts	Veiksmīgs		
TZ.B.03	06.05.2025	TP.SAK.03	Iziet pogas pārbaude	Tomijs Būmerts	Veiksmīgs		
TZ.B.04	06.05.2025	TP.IES.01	Regulēt skaņas skaļumu	Tomijs Būmerts	Neveiksmīgs	Nav pievienots neviens skaņu effekts	KZ.IES.01
TZ.B.05	06.05.2025	TP.IES.02	Lietotāja vārda maiņa	Tomijs Būmerts	Veiksmīgs		
TZ.B.06	06.05.2025	TP.IES.03	Saglabāt izmainītos iestatījumus	Tomijs Būmerts	Veiksmīgs		
TZ.B.07	06.05.2025	TP.REZ.01	Iesākt treniņu režīmu	Tomijs Būmerts	Veiksmīgs		
TZ.B.08	06.05.2025	TP.REZ.02	Iesākt 2v2 režīmu	Tomijs Būmerts	Neveiksmīgs	Nav pievienota spēles aina build	KZ.REZ.02
TZ.B.09	06.05.2025	TP.REZ.03	Iesākt 4v4 režīmu	Tomijs Būmerts	Neveiksmīgs	Nav pievienota spēles aina build	KZ.REZ.03
TZ.B.10	06.05.2025	TP.LOB.01	Izvedot jaunu lobby(telpu)	Tomijs Būmerts	Veiksmīgs		
TZ.B.11	06.05.2025	TP.LOB.02	Pievienošanās izveidotajam lobby	Tomijs Būmerts	Veiksmīgs		
TZ.B.12	06.05.2025	TP.LOB.03	Komandas izvēle starp zilo un sarkano spēles telpā	Tomijs Būmerts	Veiksmīgs		
TZ.B.13	06.05.2025	TP.VAD.01	Spēlētāja kustības pārbaudīšana ar taustiņiem WASD, kādā no spēles ainām	Tomijs Būmerts	Veiksmīgs		
TZ.B.14	06.05.2025	TP.VAD.02	Spēlētāja kustības pārbaude, skriešana	Tomijs Būmerts	Veiksmīgs		
TZ.B.15	06.05.2025	TP.VAD.03	Ripas sitiena pārbaude, kad pacelta ir ripa	Tomijs Būmerts	Veiksmīgs		

TZ.B.16	06.05.2025	TP.SPE.01	Vārtu iesišana	Tomijs Būmerts	Veiksmīgs		
TZ.B.17	06.05.2025	TP.SPE.02	Rezultāta maiņa atšķirība no vārtiem	Tomijs Būmerts	Veiksmīgs		
TZ.B.18	06.05.2025	TP.SPE.03	Laika atskaites sākšana iesākoties spēlei	Tomijs Būmerts	Neveiksmīgs	Nav pievienota, laika atskaite	
TZ.B.19	06.05.2025	TP.SPE.04	Laika iztecēšana, spēles beigas	Tomijs Būmerts	Neveiksmīgs	Laika atskaite, nav pievienota, spēles beigas nevar tādēļ noteikt	
TZ.B.20	06.05.2025	TP.TIK.01	Citu spēlētāju redzēšana, tīkla spēlē	Tomijs Būmerts	Veiksmīgs		
TZ.B.21	06.05.2025	TP.TIK.02	Rezultātu sinhronizācija	Tomijs Būmerts	Veiksmīgs		
TZ.B.22	06.05.2025	TP.TIK.03	Ripas sinhronizācija tīkla	Tomijs Būmerts	Veiksmīgs		
TZ.B.23	06.05.2025	TP.TIK.04	Atvienošanās	Tomijs Būmerts	Neveiksmīgs	Atvienošanas iespēja vēl nav pievienota	
White box							
TZ.W.01	06.05.2025	TP.AUT.01	Unity servisa autorizācija	Tomijs Būmerts	Veiksmīgs		
TZ.W.02	06.05.2025	TP.REZ.01	Režīma pārslēgšanās uz 2v2	Tomijs Būmerts	Veiksmīgs		
TZ.W.04	06.05.2025	TP.LOB.01	Lobby telpas izveide ar ģenerēto pievienošanas kodu	Tomijs Būmerts	Veiksmīgs		
TZ.W.05	06.05.2025	TP.KOM.01	Komandas maiņa lobby telpā	Tomijs Būmerts	Veiksmīgs		
TZ.W.06	06.05.2025	TP.KOM.02	Komandas maiņa lobby telpā pret pilnu komandu	Tomijs Būmerts	Neveiksmīgs	Automātiski piešķir zilo komandu, pie ieejas, lai arī tā ir pilna	KZ.KOM.02
TZ.W.07	06.05.2025	TP.PLM.01	Spēlētāju kustību apstrāde	Tomijs Būmerts	Veiksmīgs		
TZ.W.08	08.05.2025	TP.TIK.01	Attalinātās procedūras jeb Rpc izsaukšana	Tomijs Būmerts	Veiksmīgs		
TZ.W.09	08.05.2025	TP.TIK.02	Klienta autoritāte	Tomijs Būmerts	Veiksmīgs		
TZ.W.10	08.05.2025	TP.NET.01	Pozīciju sinhronizēšana	Tomijs Būmerts	Veiksmīgs		
TZ.W.11	16.05.2025	TP.VAR.01	Vārtu gūšana	Tomijs Būmerts	Veiksmīgs		

TZ.W.12	16.05.2025	TP.PCK.01	Ripas pacelšana	Tomijs Būmerts	Veiksmīgs		
TZ.W.13	16.05.2025	TP.PCK.02	Sitiens pa ripu	Tomijs Būmerts	Veiksmīgs		
TZ.W.14	16.05.2025	TP.SPM.01	Ripas izveide tīkla spēlēm	Tomijs Būmerts	Veiksmīgs		
TZ.W.15	16.05.2025	TP.SPM.02	Dubulto ripu izdzēšana izveides brīdī	Tomijs Būmerts	Veiksmīgs		
TZ.W.16	08.05.2025	TP.STA.01	Enerģijas limita noteikšana	Tomijs Būmerts	Veiksmīgs		
TZ.W.17	08.05.2025	TP.STA.02	Enerģijas izsīkums	Tomijs Būmerts	Veiksmīgs		
TZ.W.17	08.05.2025	TP.STA.03	Enerģijas atjaunošanās ātrums	Tomijs Būmerts	Veiksmīgs		
TZ.W.18	16.05.2025	TP.RPF.01	Ripas ātruma noteikšana un tā samazināšana	Tomijs Būmerts	Veiksmīgs		
TZ.W.19	16.05.2025	TP.RPF.02	Ripas berzes pievienošana	Tomijs Būmerts	Veiksmīgs		
TZ.W.20	16.05.2025	TP.RPF.03	Vārtu kolīzija	Tomijs Būmerts	Neveiksmīgs	Nav vel izveidota kolīzija ar vārtiem un ripu	KZ.RPF.01
TZ.W.21	16.05.2025	TP.TNK.01	Hosta migrācija	Tomijs Būmerts	Neveiksmīgs	Nav vēl implementa	KZ.TNK.01
TZ.W.22	16.05.2025	TP.TNK.02	Klientu sinhronizācija ar host	Tomijs Būmerts	Veiksmīgs		
TZ.W.23	16.05.2025	TP.TNK.03	Ripas autoritāte	Tomijs Būmerts	Veiksmīgs		
TZ.W.24	16.05.2025	TP.CAT.01	Ziņas garuma maksimāla noteikšana	Tomijs Būmerts	Neveiksmīgs	Nav izveidota čata sistēma	KZ.CAT.01
TZ.W.25	16.05.2025	TP.CAT.02	HTML tagu nonemšana	Tomijs Būmerts	Neveiksmīgs	Nav izveidota čata sistēma	KZ.CAT.02
TZ.W.26	16.05.2025	TP.CAT.03	Vēstures limita noteikšana	Tomijs Būmerts	Neveiksmīgs	Nav izveidota čata sistēma	KZ.CAT.03
TZ.W.27	16.05.2025	TP.IES.01	Izvēlētās skaņas saglabāšana	Tomijs Būmerts	Veiksmīgs		
TZ.W.28	16.05.2025	TP.IES.02	Vārda saglabāšana	Tomijs Būmerts	Veiksmīgs		
TZ.W.29	16.05.2025	TP.IES.03	Nekorektu vērtību novēršana	Tomijs Būmerts	Veiksmīgs		
TZ.W.30	16.05.2025	TP.IES.04	Datu saglabāšana	Tomijs Būmerts	Veiksmīgs		
TZ.W.31	08.05.2025	TP.ANM.01	Skriešanas animācijas attēlošana	Tomijs Būmerts	Veiksmīgs		
TZ.W.32	08.05.2025	TP.ANM.02	Sitiena animācijas attēlošana pec spēlētāja peles klikšķa nospiešanas	Tomijs Būmerts	Neveiksmīgs	Animācija ne vienmēr tiek parādīta visiem pievienojošajiem spēlētājiem	KZ.ANM.02

TZ.W.33	16.05.2025	TP.MEH.01	Ripas fizika	Tomijs Būmerts	Veiksmīgs		
TZ.W.34	16.05.2025	TP.MEH.02	Spēlētāju savstarpēja	Tomijs Būmerts	Veiksmīgs		

Secinājumi

Kopējais rezultāts:

Izstrādātā datorspēle "HockeyShot" ir sasniegusi izvirzītos mērķus gan funkcionāli, gan tehniski. Projekta gaitā iegūtās zināšanas un sasniegtais rezultāts ir sniedzis lielu gandarijumu projekta Izstrādātājam.

Sasniegumi un Izaicinājumi:

Veiksmīgi realizētās projekta ieceres:

- Izveidota reāllaika tīkla spēle ar vairākiem spēlētājiem
- Nodrošināta precīza spēlētāju kustību sinhronizācija
- Panākta aizraujoša spēle, ko rada iespēja spēlēt ar draugiem, vai citiem spēlētājiem

Nerealizētās ieceres projektā:

- Modernāka lietotāja saskarne (UI) dizaina izveide
- Papildus kontroles mehānismu ieviešana spēlētājiem, padarot spēli nedaudz sarežģītāku

Galvenie izaicinājumi:

- Hokeja spēles mehānikas izveide
- Tīkla funkcionalitātes ieviešana
- Optimāla risinājuma izvēle no daudziem pieejamajiem risinājumiem, lai izveidotu projektu.

Iepriekšējās pieredzes priekšrocības:

- Skolas projektu pieredze ar Unity dzini, tādejādi atvieglojot projekta izstrādi, neveicot pilnīgu apguvi
- Zināšanas no iepriekšējā "Cirka spēles" projekta
- Praktiskā pieredze līdzīgu uzdevumu risināšanā

Nākotnes plāni projektam:

Ja tiktu veikti plāni nākotnei, tie iekļautu vairākus uzlabojumus kurus varētu veikt, lai spēle justos dabiskāka, piemēram, uzlabotu fiziku spēles objektiem radot spēli vairāk fiziski bāzētu. Tiktu pievienotas papildus kontroles spēlētājiem, lai spēle būtu mazliet advancētāka, un ilgstspējīgāka, tiktu arī uzlabots UI dizains. Un vēl papildus vārstarga spēlētāja pozīciju, lai spēlētāji arī varētu uzņemties šādu lomu spēlē.

Secinājumi:

Projekts demostrē veiksmīgu iepriekš apgūto zināšanu pielietošanu projekta prasību sasniegšanai. Izstrādātājs ir apmiernāts ar sasniegto rezultātu un redz potenciālu, ja spēle tiktu turpināta arī nākotnē.

Lietoto terminu un saīsinājumu skaidrojumi

5. tabula

Saīsinājums	Skaidrojums
Izstrādātājs	Indivīds vai organizācija, kas veido programmatūru pēc pasūtītāja iniacatīvas un prasībām
Lietotājs	Persona vai personas, kas lieto programmatūru ar noteiktu uzdevumu
UML	Vizuālā modelēšana valoda, kas paredzēta sistēmu projektēšanas standartizēšanai
Aina	Viens no pamata elementiem spēļu projektā, kā piemēram, Unity

Literatūras un informācijas avotu saraksts

1. How to use Unity Relay, Multiplayer through FIREWALL! (Unity Gaming Services) - Adrese(youtube.com)
2. Ice Hockey Player modelis - Adrese (Unity Asset Store)
3. Ice Hockey Customizable Characters laukums - Adrese (Unity Asset Store)

Pielikumi

1. pielikums

```
private void HandleMovementInput()
    {
        float horizontal = Input.GetAxis("Horizontal"); // A/D tikai rotācijai
        float vertical = Input.GetAxis("Vertical");     // W/S tikai kustībai
        bool sprint = Input.GetKey(KeyCode.LeftShift);
        bool quickStop = Input.GetKey(KeyCode.Space);   // JAUNS: Space taustiņš ātrai apturēšanai

        currentSprintState = sprint;        if (IsOwner)
        {
            // Saglabā pašreizējo ātrumu pirms jebkādām izmaiņām
            Vector3 currentVel = rb != null ? rb.linearVelocity : Vector3.zero;
            float currentHorizontalSpeed = new Vector3(currentVel.x, 0f, currentVel.z).magnitude;

            // LEDUS FIZIKA: Vienmērīgāka rotācija ar impulsa saglabāšanu
            if (Mathf.Abs(horizontal) > 0.01f)
            {
                // Rotācija notiek neatkarīgi no kustības stāvokļa - saglabā impulsu
                float rotationAmount = horizontal * rotationSpeed * Time.deltaTime;
                // Samazina rotācijas ātrumu, kad kustas ātri, lai būtu reālistiskāki ledus pagriezieni
                if (currentHorizontalSpeed > 30f)
                {
                    rotationAmount *= Mathf.Lerp(1f, 0.6f, (currentHorizontalSpeed - 30f) / 70f);
                }
                transform.Rotate(0f, rotationAmount, 0f);
            }

            // PRIORITĀTE 1: Ātrā apstāšanās ar atstarpi (pārspēj visu)
            if (quickStop && rb != null)
            {
                // LEDUS FIZIKA: Pakāpeniskāka apturēšana (joprojām ātra, bet jūtas vairāk ledaina)
                Vector3 stoppedVel = currentVel * 0.7f; // Mazāk agresīva nekā 0.5f
                rb.linearVelocity = new Vector3(stoppedVel.x, currentVel.y, stoppedVel.z);
            }
            // PRIORITĀTE 2: Aktīvā kustības ievade (W/S nospiests)
            else if (Mathf.Abs(vertical) > 0.1f)
            {
                // LEDUS FIZIKA: Pakāpeniska paātrināšanās pretstatā tūlītējam ātrumam
                float targetSpeed = sprint ? sprintSpeed : moveSpeed;
                Vector3 targetDirection = transform.forward * vertical;
                Vector3 targetVelocity = targetDirection * targetSpeed;
                  if (rb != null)
                {
                    Vector3 currentHorizontalVel = new Vector3(currentVel.x, 0f, currentVel.z);
                    Vector3 velocityDiff = targetVelocity - currentHorizontalVel;

                    // Piemēro paātrinājuma spēku reālistiskākai ledus sajūtai
                    float accelForce = acceleration * Time.deltaTime;
                    Vector3 newVelocity;

                    if (velocityDiff.magnitude > accelForce)
                    {
                        // Pakāpeniska paātrināšanās
                        newVelocity = currentHorizontalVel + velocityDiff.normalized * accelForce;
                    }
                    else
```

```
                    {
                        // Pietiekami tuvu mērķim
                        newVelocity = targetVelocity;
                    }

                    rb.linearVelocity = new Vector3(newVelocity.x, currentVel.y, newVelocity.z);
                }
            }
            // PRIORITĀTE 3: Impulsa uzturēšana - LEDUS FIZIKA: Dabiska slīdēšana
            else if (currentHorizontalSpeed > 0.1f)
            {
                // LEDUS FIZIKA: Piemēro pakāpenisku palēnināšanos, kad nav ievades
                if (rb != null)
                {
                    Vector3 horizontalVel = new Vector3(currentVel.x, 0f, currentVel.z);
                    float decelAmount = deceleration * Time.deltaTime;

                    if (horizontalVel.magnitude > decelAmount)
                    {
                        Vector3 decelVel = horizontalVel - horizontalVel.normalized * decelAmount;
                        rb.linearVelocity = new Vector3(decelVel.x, currentVel.y, decelVel.z);
                    }
                    else
                    {
                        // Gandrīz apstājies
                        rb.linearVelocity = new Vector3(0f, currentVel.y, 0f);
                    }
                }
            }
            // PRIORITĀTE 4: Pilnīga apstāšanās (nav nepieciešama ātruma regulēšana)
            // Ļauj dabiskai palēnināšanai apstrādāt galīgo apstāšanos            // LABOTS: Atjaunina
animācijas TIKAI, balstoties uz aktīvu W/S ievadi (nevis impulsu vai ātrumu)
            if (animator != null)
            {
                // SVARĪGI: Animācija tiek atskaņota TIKAI aktīvi spiežot W/S, NEVIS impulsa uzturēšanas
laikā
                bool isActivelyMoving = Mathf.Abs(vertical) > 0.1f && !quickStop;
                animator.SetBool("IsSkating", isActivelyMoving);
                animator.speed = sprint ? 1.5f : 1.0f;

                // Atkļūdošanas ieraksts animācijas stāvokļa pārbaudei
                Debug.Log($"DIAGNOSTIKA: Animācija - vertikālā ievade: {vertical:F2}, aktīvi kustīgs:
{isActivelyMoving}, ātrā apture: {quickStop}");
            }
        }

        // Tīkla sinhronizācija: Nosūta visas ievades, ieskaitot ātro apstāšanos
        bool inputChanged = Mathf.Abs(horizontal - lastSentHorizontal) > 0.05f ||
                            Mathf.Abs(vertical - lastSentVertical) > 0.05f ||
                            sprint != lastSentSprint;

        bool shouldSend = inputChanged || (Time.time - lastInputSendTime) > inputSendRate;
```

3. pielikums

```
        if (shouldSend && NetworkManager.Singleton != null && IsSpawned)
        {
            MoveServerRpc(horizontal, vertical, sprint, quickStop, transform.position,
transform.rotation);
            lastSentHorizontal = horizontal;
            lastSentVertical = vertical;
            lastSentSprint = sprint;
            lastInputSendTime = Time.time;
        }
    }

    [ServerRpc]
    private void MoveServerRpc(float horizontal, float vertical, bool sprint, bool quickStop, Vector3
clientPosition, Quaternion clientRotation)
    {
        if (rb == null) return;

        currentSprintState = sprint;

        if (IsOwner)
        {
            transform.position = clientPosition;
            transform.rotation = clientRotation;
        }       else // NE-ĪPAŠNIEKS (attālie klienti) - piemēro servera puses kustību
        {
            Vector3 currentVel = rb.linearVelocity;
            float currentHorizontalSpeed = new Vector3(currentVel.x, 0f, currentVel.z).magnitude;

            // LEDUS FIZIKA: Piemēro tādu pašu rotācijas loģiku attālajiem klientiem
            if (Mathf.Abs(horizontal) > 0.01f)
            {
                float rotationAmount = horizontal * rotationSpeed * Time.fixedDeltaTime;
                if (currentHorizontalSpeed > 30f)
                {
                    rotationAmount *= Mathf.Lerp(1f, 0.6f, (currentHorizontalSpeed - 30f) / 70f);
                }
                transform.Rotate(0f, rotationAmount, 0f);
            }           if (quickStop)
            {
                Vector3 stoppedVel = currentVel * 0.7f; // Tāds pats kā īpašniekam
                rb.linearVelocity = new Vector3(stoppedVel.x, currentVel.y, stoppedVel.z);
            }
            else if (Mathf.Abs(vertical) > 0.1f)
            {
                // LEDUS FIZIKA: Tāda pati pakāpeniskā paātrināšanās attālajiem klientiem
                float targetSpeed = sprint ? sprintSpeed : moveSpeed;
                Vector3 targetDirection = transform.forward * vertical;
                Vector3 targetVelocity = targetDirection * targetSpeed;

                Vector3 currentHorizontalVel = new Vector3(currentVel.x, 0f, currentVel.z);
                Vector3 velocityDiff = targetVelocity - currentHorizontalVel;
```

```
            Vector3 targetVelocity = targetDirection * targetSpeed;

            Vector3 currentHorizontalVel = new Vector3(currentVel.x, 0f, currentVel.z);
            Vector3 velocityDiff = targetVelocity - currentHorizontalVel;

            float accelForce = acceleration * Time.fixedDeltaTime;
            Vector3 newVelocity;

            if (velocityDiff.magnitude > accelForce)
            {
                newVelocity = currentHorizontalVel + velocityDiff.normalized * accelForce;
            }
            else
            {
                newVelocity = targetVelocity;
            }

            rb.linearVelocity = new Vector3(newVelocity.x, currentVel.y, newVelocity.z);
        }           else if (currentHorizontalSpeed > 0.1f)
        {
            // LEDUS FIZIKA: Tāda pati pakāpeniskā palēnināšanās attālajiem klientiem
            Vector3 horizontalVel = new Vector3(currentVel.x, 0f, currentVel.z);
            float decelAmount = deceleration * Time.fixedDeltaTime;

            if (horizontalVel.magnitude > decelAmount)
            {
                Vector3 decelVel = horizontalVel - horizontalVel.normalized * decelAmount;
                rb.linearVelocity = new Vector3(decelVel.x, currentVel.y, decelVel.z);
            }
            else
            {
                rb.linearVelocity = new Vector3(0f, currentVel.y, 0f);
            }
        }           // TIKAI ATTĀLAJIEM KLIENTIEM: Bloķē Y poziciju un ātrumu
        Vector3 pos = transform.position;
        if (Mathf.Abs(pos.y - 0.71f) > 0.001f)
        {
            pos.y = 0.71f;
            transform.position = pos;
            rb.position = pos;
            rb.linearVelocity = new Vector3(rb.linearVelocity.x, 0f, rb.linearVelocity.z);
        }
    }


    networkPosition.Value = transform.position;
    if (IsOwner)
    {

        networkVelocity.Value = rb != null ? rb.linearVelocity : Vector3.zero;
    }
    else
    {

        networkVelocity.Value = rb != null ? rb.linearVelocity : Vector3.zero;
    }
    isSkating.Value = Mathf.Abs(vertical) > 0.1f && !quickStop;
```