Kauno technologijos universitetas

Informatikos fakultetas

Turinio generavimo metodų, skirtų šaudyklės žanro žaidimui, tyrimas

Baigiamasis magistro studijų projektas

Lukas Kaladė

Projekto autorius

Assoc. Prof. Dr. Tomas Blažauskas

Vadovas

Kaunas, 2023

Kauno technologijos universitetas

Informatikos fakultetas

Turinio generavimo metodų, skirtų šaudyklės žanro žaidimui, tyrimas

Baigiamasis magistro studijų projektas

Programų sistemų inžinerija (6211BX011)

Lukas Kaladė

Projekto autorius

Assoc. Prof. Dr. Tomas Blažauskas

Vadovas

Dr. Šarūnas Packevičius

Recenzentas

Kaunas, 2023

Kauno technologijos universitetas

Informatikos fakultetas

Lukas Kaladė

Turinio generavimo metodų, skirtų šaudyklės žanro žaidimui, tyrimas

Akademinio sąžiningumo deklaracija

Patvirtinu, kad:

1. baigiamąjį projektą parengiau savarankiškai ir sąžiningai, nepažeisdama(s) kitų asmenų autoriaus ar kitų teisių, laikydamasi(s) Lietuvos Respublikos autorių teisių ir gretutinių teisių įstatymo nuostatų, Kauno technologijos universiteto (toliau – Universitetas) intelektinės nuosavybės valdymo ir perdavimo nuostatų bei Universiteto akademinės etikos kodekse nustatytų etikos reikalavimų;

2. baigiamajame projekte visi pateikti duomenys ir tyrimų rezultatai yra teisingi ir gauti teisėtai, nei viena šio projekto dalis nėra plagijuota nuo jokių spausdintinių ar elektroninių šaltinių, visos baigiamojo projekto tekste pateiktos citatos ir nuorodos yra nurodytos literatūros sąraše;

3. įstatymų nenumatytų piniginių sumų už baigiamąjį projektą ar jo dalis niekam nesu mokėjęs (-usi);

4. suprantu, kad išaiškėjus nesąžiningumo ar kitų asmenų teisių pažeidimo faktui, man bus taikomos akademinės nuobaudos pagal Universitete galiojančią tvarką ir būsiu pašalinta(s) iš Universiteto, o baigiamasis projektas gali būti pateiktas Akademinės etikos ir procedūrų kontrolieriaus tarnybai nagrinėjant galimą akademinės etikos pažeidimą.

Lukas Kaladė

Patvirtinta elektroniniu būdu

Kauno technologijos universitetas

Informatikos fakultetas

Baigiamojo magistro projekto užduotis

Projekto tema: Turinio generavimo metodų, skirtų šaudyklės žanro žaidimui, tyrimas.

Reikalavimai ir sąlygos (tikslinti pavadinimą pagal poreikį):

Vadovas / Vadovė

(vadovo pareigos, vardas, pavardė, parašas) (data)

Kaladė Lukas. Turinio generavimo metodų, skirtų šaudyklės žanro žaidimui, tyrimas. Magistro baigiamasis projektas / vadovas Assoc. Prof. Dr. Tomas Blažauskas; Kauno technologijos universitetas, Informatikos fakultetas.

Studijų kryptis ir sritis (studijų krypčių grupė): Informatikos mokslai, programų sistemos.

Reikšminiai žodžiai: vaizdo žaidimai, žaidimų kūrimas, Unity žaidimų variklis, turinio generavimas, atsitiktinis turinys, šaudyklės.

Kaunas, 2023. 72 p.

Santrauka

Šiame darbe aprašomas „Toxic Wasteland“ žaidimas, skirtas žaisti tinkle nuo vieno iki keturių žaidėjų. Žaidimas buvo įgyvendintas kartu su kolega magistrinio darbo metu ir susideda iš 6 pagrindinių posistemių: grafinė vartotojo sąsaja, žaidėjo duomenų valdymas, žaidėjo valdymas, žaidimo eigos valdymas, tinklo valdymas bei garsų valdymas. Žaidimas yra pirmojo asmens šaudyklė, kurios lygiai, daiktai bei priešai yra atsitiktinai sugeneruojami iš naujo kiekvieną kartą pradedant žaidimą.

Analizės dalyje analizuojami įvairūs būdai kurti atsitiktiniams žaidimų lygiams ar žemėlapiams, aprašomi jų privalumai bei trūkumai ir pasirenkamas geriausias metodas suprojektuotam žaidimui įgyvendinti.

Tyrimo dalyje pasirinktos 5 metrikos, pagal kurias ištiriama žaidimo atsitiktinio turinio generavimo algoritmų kokybė, pasiūlomi, įgyvendinami ir aprašomi algoritmų patobulinimai.

Eksperimentinėje dalyje apskaičiuojamos tyrimo dalyje pasirinktos 5 kokybės metrikos ir įvertinama atsitiktinių turinio generavimo algoritmų kokybė po įgyvendintų algoritmų patobulinimų. Palyginamos metrikos prieš ir po patobulinimų ir daromos išvados ar patobulinimai buvo efektyvūs.

Kaladė Lukas. A Study of Content Generation Techniques for a Shooter Game. Master's Final Degree Project / supervisor Assoc. Prof. Dr. Tomas Blažauskas; The Faculty of Informatics, Kaunas University of Technology.

Study field and area (study field group): Computer science, program systems.

Keywords: videogames, game development, Unity game engine, content generation, random content, shooters.

Kaunas, 2023. 72 p.

Summary

This paper describes the ‘‘Toxic Wasteland‘‘ game, which can be played by one and up to four players. The game was implemented with a colleagues during master‘s thesis work and consists of 6 main subsystems: graphical user interface, player data management, game management, game progress management, network management and sound management. The game is a first-person shooter with levels, items, and enemies that are randomly generated every time the game is started.
The analysis part analyzes the ways of procedurally creating game levels or maps, describes their advantages and disadvantages and chooses the best method to implement into the designed game.
The research part examines the quality of game random content generation algorithms based on selected 5 metrics, proposes, implements, and describes improvements to the algorithms.
The experimental part calculates the 5 quality metrics selected in the research part and evaluates the quality of random content generation algorithms after the implemented algorithm improvements. Comparing metrics before and after improvements and concluding whether the improvements were effective.

Turinys

Paveikslų sąrašas

Santrumpų ir terminų sąrašas

ASCII – 7 bitų koduotė, įteisinta JAV standartu ir užregistruota kaip tarptautinio standarto modifikacija ISO 646-IRV;
Atsitiktinių skaičių generatorius – programos komponentas atsitiktiniams skaičiams gauti;
Atsitiktinio turinio generavimas – procesas, kurio metu yra sukuriamas turinys naudojant algoritmus ir modelius su atsitiktiniais rezultatais;
Bitas – mažiausias informacijos kiekio vienetas kompiuteryje, koduojamas vienu dviejų būsenų elementu;
„Epic Store“ – internetinė žaidimų prekybos platforma, kurioje galima įsigyti kompiuterinius žaidimus;
„FizzySteamworks“ – atviro kodo biblioteka, kuri suteikia galimybę naudoti „Steam“ integruotas žaidimų funkcijas programinės įrangos projektuose, sukurtuose naudojant „Unity“ žaidimų kūrimo variklį;
Gauso Filtras – matematinis filtras, kuris naudojamas siekiant sumažinti triukšmą ir išryškinti signalų reikšmingą informaciją;
GHz – skaitinis matavimo vienetas, nurodantis milijardo ciklų per sekundę kiekį.
Kadrai per sekundę – rodiklis, skaičiuojantis kiek kompiuteris sugeba sugeneruoti kadrų per sekundę, naudojamas žaidimų greitaveikai matuoti;
Medianinis filtras – signalų apdorojimo technika, kuri naudojama triukšmo mažinimui ir signalų išvalymui;
„Mirror“ – „Unity“ paketas, turintis biblioteką kurti žaidimams tinkle;
ms – laiko matavimo vienetas, kuris atitinka tūkstantąją sekundės dalį;
RAM – kompiuterio atminties rūšis, kuri yra naudojama laikinai saugoti informaciją, kuri reikalinga kompiuterio darbui;
„Roguelike“ – žaidimų žanras, kurio žaidimuose dažniausiai yra atsitiktinai sugeneruojami žaidimo lygiai, galimybė prarasti visus savo pažangumus ir pradėti žaidimą iš naujo po mirties, bei atsitiktiniai daiktų ir varžovų pasiskirstymai;
„Roguelite“ – žaidimų žanras, dažniausiai yra mažiau griežtas už „Roguelike“, paprastai neturi galimybės prarasti visus savo pažangumus ir pradėti žaidimą iš naujo po mirties, arba ši galimybė yra ribota;
Shannon diversija – būdas apskaičiuoti įvairovę, matuojamas bitais;
„Steam“ – žaidimų įkėlimo platforma, skirta platinti ir parduoti žaidimus;
„Steam Friends“ – „Steam“ platformos funkcija, kuri leidžia naudotojams valdyti ir susisiekti su savo „Steam“ platformoje esančiais draugais;
„Steamworks“ – „Steam“ įrankių bei paslaugų rinkinys, padedantis kurti žaidimus;
„Unity“ – įvairiaplatformis žaidimų kūrimo įrankis, kurį sudaro žaidimų variklis ir integruota kūrimo aplinka;
„Unity Asset Store“ – internetinė platforma, skirta „Unity“ programinės įrangos naudotojams pirkti ir parduoti šablonus, bibliotekas, įrankius, modelius, vaizdinius, garsus ir kitus išteklius, kurie gali būti naudojami „Unity“ projektuose.
Žaidimo lygis – žaidimo etapas, kuris gali apimti tam tikrą užduočių rinkinį arba specifinę žaidimo teritoriją;
Žaidimo variklis – programinė įranga, skirta kurti žaidimams;
Žaidimo žemėlapis – virtualus vaizdas, kuriame yra pavaizduota žaidimo erdvė;

Įvadas

Žaidimų kūrimo procesas reikalauja specifinių žinių apie visus žaidimo komponentus, kadangi nuo to priklauso tolimesnė žaidimo kūrimo eiga. Kuriamo žaidimo variklis, sudėtingumas, esmė, mechanikos, tėkmė, išvaizda ir visi kiti elementai yra glaudžiai susiję su žaidimo lygiais, nes tai yra grandis, kurioje visi šie elementai funkcionuoja ir yra atvaizduojami. Ypač sudėtingi ir detalūs žaidimai gali reikalauti atskiro lygių kūrėjo dviem, ar net vienam lygiui sukurti [1]. Negana to, kad lygių kūrimas yra sunkus, užima daug laiko ir reikalauja didelės žmonių komandos, taip pat iškyla problema, kad vienodi, nekintantys žaidimo lygiai tampa monotoniški ir žaidimas praranda tikslą būti žaidžiamas dar kartą [2]. Šioms problemoms išspręsti galima pasitelkti atsitiktinio, dar vadinamo, procedūrinio žaidimo turinio generavimo algoritmus. Šie algoritmai padeda sumažinti žaidimo kūrimo laiką ir kainą, kadangi tam tikrą žaidimo turinį generuoja pats žaidimas, o ne lygių kūrėjas [3]. Atsitiktinio turinio generavimo algoritmai leidžia žaidimams kurti įvairų žaidimo turinį patiems su minimaliu (arba jokiu) žmogaus įsitraukimu. Generuojamas turinys gali būti labai įvairus: lygiai, žaidimų taisyklės, tekstūros, istorijos, daiktai, užduotys, muzika, ginklai, žaidimų išvaizda, veikėjai, atlygiai, sunkumai ir kita [4]. Šie algoritmai dažniausiai naudoja atsitiktinių skaičių generatorius, kad sukurtų nenuspėjamus rezultatus. Atsitiktinio turinio generavimo algoritmai yra svarbūs tiek žaidimų kūrėjams, nes jie mažina laiką ir piniginius kaštus reikalingus norint sukurti žaidimą bei leidžia atnaujinti žaidimą tiesiog tobulinant algoritmą arba duodant jam daugiau elementų iš kurių jis gali kurti žaidimo turinį, tiek ir žaidėjams, nes žaidėjams yra suteikiama nesibaigianti, unikali ir daug kartų pakartojama žaidimo patirtis, kadangi kiekvieną kartą žaidžiant žaidimą, jo komponentai bus sukurti kitokia tvarka ir kiekvienas žaidėjas gali patirti unikalius žaidimo potyrius [5]. Vis dėlto, naudojant atsitiktinio turinio generavimo algoritmus, iškyla nauja problema žaidimų kūrėjams: nesibaigiančio turinio žaidimą pilnai ištestuoti yra praktiškai neįmanoma. Galima ištestuoti algoritmo veikimą ir žaidimo eigą dalyje pateiktų rezultatų, tačiau visada išlieka tikimybė, jog mažoje dalyje algoritmo pateiktų rezultatų, žaidimas turės stambių klaidų, kurios gali neigiamai paveikti žaidimo patirtį žaidėjams. Sprendžiant šią problemą, galima naudoti dygsniuotojo turinio atsitiktinio generavimo algoritmus. Tai atsitiktinio turinio generavimas naudojant tam tikrus, žmonių sukurtus komponentų blokus, kuriuos atsitiktinio turinio algoritmai sujungia į vieną lygį. Vienas iš labai populiarių ir sėkmingų žaidimų, naudojančių dygsniuoto turinio atsitiktinį generavimą, yra „Diablo 2“ [3]. Šio žaidimo žemėlapiai yra sugeneruojami iš žaidimo lygių dizainerių sukurtų dalių ir sudedami į vieną didelį žemėlapį. Tai padeda išvengti nenuspėjamo turinio, kurį galėtų sukurti algoritmas, o žaidimas tampa lengviau ištestuojamas ir daug kokybiškesnis. Tai yra puikus kompromisas žaidimų kūrėjams, kurie nori naudoti atsitiktinį turinio generavimą, tačiau privalo užtikrinti, kad visi žaidimo lygiai neturės kritinių, žaidimą griaunančių, klaidų.

Darbo naujumas ir aktualumas

Atsitiktiniai generuojamas turinys žaidimuose atsirado jau senai. Šių metodų naudojimas ankščiau dažniausiai buvo naudojamas norint išvengti labai limituotos atminties sistemose. Maža sistemos atmintis reikšdavo, kad žaidimai negalėjo turėti daug kokybiško turinio įpakuoto į patį žaidimą, todėl turinys būdavo generuojamas per algoritmus. Vienas iš senų ir geriausiai žinomų žaidimų, kurie naudojo atsitiktinai sugeneruotą turinį, buvo „Rogue“. Šis žaidimas naudojo atsitiktinai sugeneruotas grafikas, kurias piešdavo naudojant ASCII simbolius. Žaidimas taip pat turėjo atsitiktinai išdėstytus lygius ir žaidimo objektus, todėl žaidimas kiekvieną kartą buvo skirtingas ir unikalus. Žaidimo sėkmė buvo tokia didelė, kad jo vardu buvo pavadintas visas žaidimų žanras „Roguelike“ [6].

1 pav. „Rogue“ žaidimas [7].

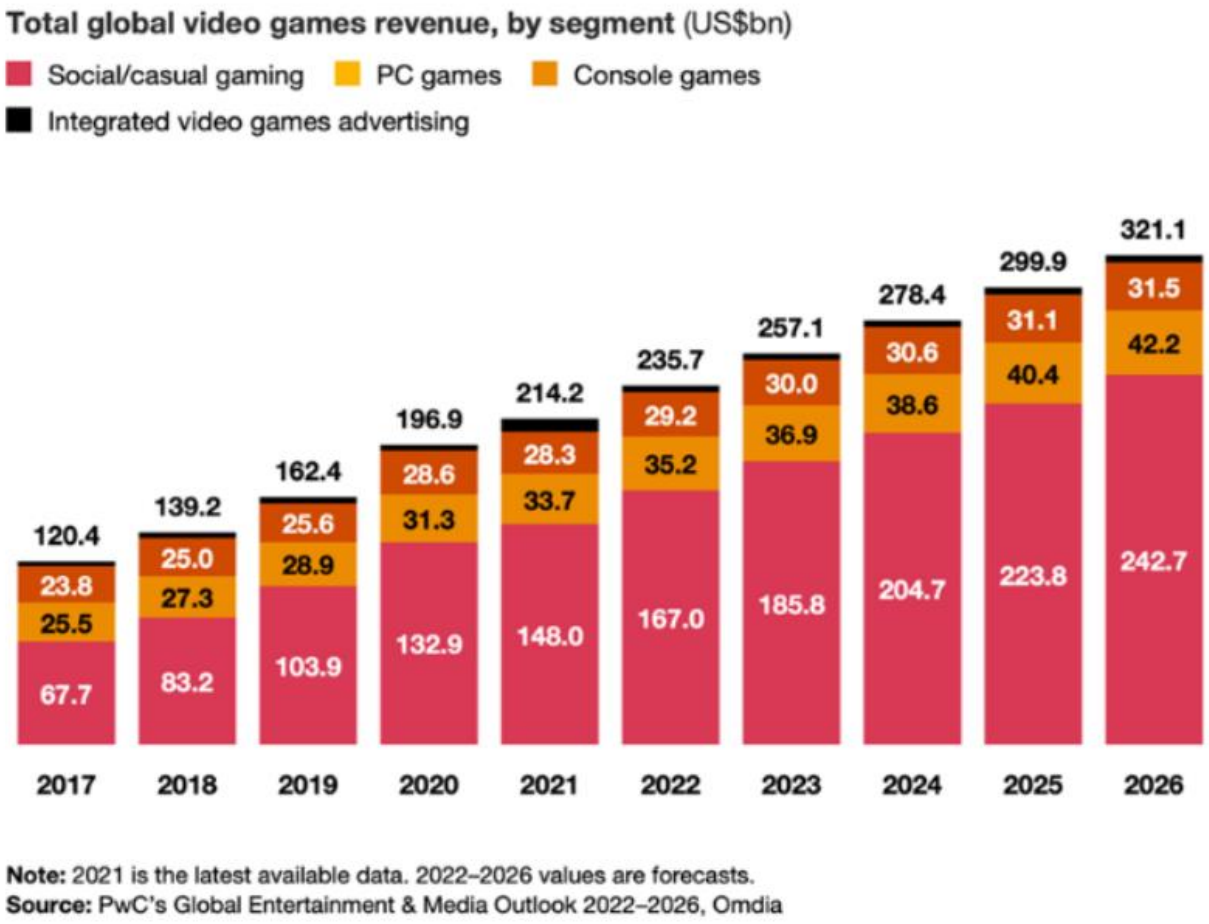
Nors atsitiktinio turinio generavimo metodai žaidimuose buvo pradėti naudoti jau senai, jie vis dar yra labai populiarus žaidimų elementas. Vien „Steam“ parduotuvėje galima rasti virš 3000 „Roguelike“ ar „Roguelite“ žanro žaidimų [8]. „Roguelite“ yra „Roguelike“ žaidimų subžanras, kuriame, priešingai nei „Roguelike“ žaidimuose, žaidėjas turi tam tikrų žaidimų elementų, kurie yra išsaugomi tarp žaidimų ir net mirus žaidimo veikėjui ar žaidimui pasibaigus, koks nors žaidimo elementas (pvz. pinigai) yra išsaugomi ir yra galimi panaudoti kitam žaidimo raundui palengvinti, tuo tarpu „Roguelike“ žaidimai neišsaugo nieko iš už būtent dabar žaidžiamo raundo ribų. Visi sukaupti žaidimo pinigai, daiktai, patirtis ar kiti elementai yra prarandami numirus žaidimo veikėjui ar kitaip pasibaigus žaidimui [9]. Vienas iš pirmųjų žaidimų, kurio reklamavimosi metu pirmą kartą buvo panaudotas žodis „Roguelite“ yra žaidimas „Rogue Legacy“. Šiame žaidime kiekvienas naujas veikėjas yra praeito veikėjo palikuonis. Žaidimas turi elementų, tokių kaip per žaidimą statoma pilis, kurie yra išsaugomi tarp žaidimo raundų, todėl žaidimas yra laikomas „Roguelite“ [10].

2 pav. „Rogue Legacy“ žaidimas [11].

„Roguelike“ žanras išlieka ne tik populiarus, tačiau ir pelningas. 10 populiariausių „Roguelike“ ar „Roguelite“ žaidimų buvo išleisti tarp 2014 ir 2021 metų ir yra verti nuo apie 18 milijonų dolerių iki apie 99 milijonų dolerių [12]. Šiuo metu vertingiausiu iš jų yra laikomas žaidimas „Hades“.

3 pav. „Hades“ žaidimas [13].

Populiarumą ir pelningumą rodo ne tik „Roguelike“ žanro žaidimai, tačiau ir visa žaidimų industrija, kuri, priešingai nei kitos, augo tik dar labiau pandemijos laikotarpiu. Tarp 2019 ir 2021 metų žaidimų industrija išaugo 26%. 2021 metais industrijos vertė buvo 214 milijardų dolerių, o 2026 metais yra spėjama, kad šie skaičiai sieks net 320 milijardus dolerių [14].



4 pav. Žaidimų industrijos augimas [14].

Personalinių kompiuterių vaizdo žaidimų rinkoje dominuoja dvi žaidimų parduotuvės, „Epic Store“ bei „Steam“. Iš šių dviejų, pranašesnė yra „Steam“, kurios naudotojų skaičius yra 50% didesnis nei „Epic“ store, yra senesnė, labiau išvystyta ir dokumentuota [15]. Todėl projektui pasirinkta „Steam“ žaidimų parduotuvės platforma ir sąsaja dėl patogumo ir didesnio potencialių žaidėjų kiekio.

5 pav. „Steam“ parduotuvės naudotojų statistika [16].

Darbo tikslas ir uždaviniai

Šiame dokumente aprašomo darbo tikslas yra ištirti atsitiktinio turinio generavimo technologijas vaizdo žaidimuose, pritaikyti vieną iš analizuotų technologijų savo projekte, išnagrinėti pritaikytą technologiją, rasti jos problemas ir pasiūlyti bei realizuoti sprendimus ar patobulinimus.

Norint įgyvendinti tikslą darbas buvo išskaidytas į šiuos uždavinius:

1. Atlikti atsitiktinio turinio generavimo metodų, kurie yra skirti žaidimų žemėlapių ar lygių kūrimui, analizę, įvertinti jų teigiamas ir neigiamas savybes bei pasirinkti vieną iš jų projektuojamos sistemos taikymui.
2. Suprojektuoti ir įgyvendinti žaidimą, kuris naudotų pasirinktą išanalizuotą atsitiktinio turinio generavimo metodą skirtą žaidimo lygių kūrimui. Sukurti ar įgyvendinti papildomus atsitiktinio turinio generavimo metodus skirtus kitoms žaidimo dalims.
3. Ištirti suprojektuotos ir įgyvendintos žaidimo sistemos kokybę. Išanalizuoti žaidimo tobulinimo galimybes bei įgyvendinti ir pagrįsti tobulinimus.
4. Atlikti ir aprašyti sukurtos ir įdiegtos žaidimo sistemos bei jos patobulinimų eksperimentinį tyrimą.

Darbo struktūra

Šis dokumentas susideda iš įvado dalies, kurioje yra aprašoma problema ir bendrai apibūdinami analizuojami metodai ir jų svarba, taip pat analizės dalies, kurioje atliekama įvairių problemos sprendimų analizė ir pateikiami populiariausių ar tinkamiausių atsitiktinio turinio generavimo metodų aprašymai, jų savybės, pliusai bei minusai. Projektinėje dalyje aprašomas suprojektuotas ir įgyvendintas projektas, kuris naudoja vieną iš analizuotų atsitiktinio turinio generavimo metodų, pateikiamos sistemos funkcijos, panaudojimo atvejai, bei kitos sistemos projektavimo diagramos. Dokumento tyrimo dalyje aprašomas įgyvendinto žaidimo kokybės tyrimas ir analizuojamos sistemos tobulinimo galimybės bei jų įgyvendinimas. Eksperimentinėje dalyje aprašomas žaidimo bei įgyvendintų patobulinimų eksperimentinis tyrimas. Išvadų dalyje yra pateikiami eksperimentų, analizių bei tyrimų rezultatai bei apibendrinamas darbas.

1. Analitinė dalis

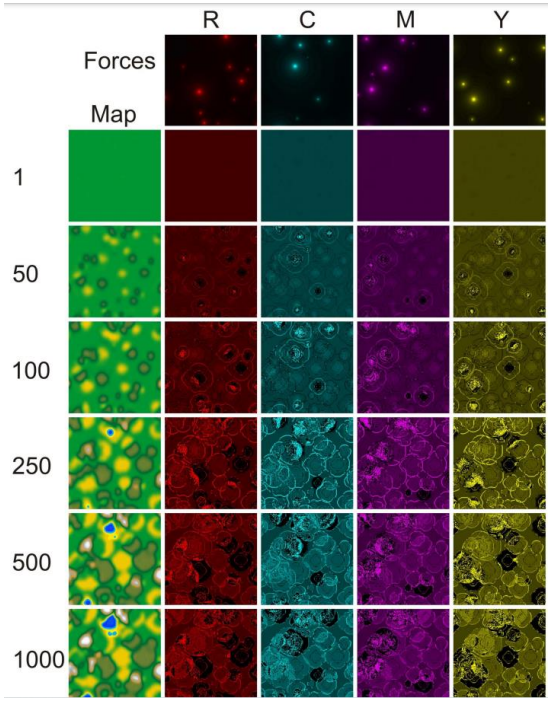
Analizės metu nagrinėjami įvairūs atsitiktinio turinio generavimo būdai, siekiant rasti būdą, kuris tenkintų daugiausiai mūsų norų ir būtų tinkamas įdiegti į projektuojamą žaidimą. Atsitiktinio turinio generavimas turėtų būtu lengvai prieinamas ir gerai dokumentuotas, išplečiamas, galėtų prisitaikyti prie žaidėjo norų, būtų efektyvus ir generuotų lygius, kurie yra suprantami, gali būti sukurti pakartotinai, ištestuojami, peržaidžiami tačiau ne vienodi. Toks atsitiktinio generavimo metodas leistų turėti atsitiktinį turinį žaidime, tačiau nebūtų visiškai nenuspėjamas, todėl būtų įmanomas ištestuoti ir įsitikinti, kad žaidimas būtų pražaidžiamas ir be didelių, žaidimą griaunančių klaidų.

1.1. Pažengusio vaizdų apdorojimo metodas

Nors vaizdų apdorojimas dažniausiai naudojamas kompiuterių moksle kaip apsaugos, duomenų analizavimo, vaizdavimo ar apskaičiavimo įrankis, šiuolaikiniai kompiuteriai yra pakankamai greiti, kad vaizdų apdorojimas galėtų būti naudojamas ir pramoginėms veikloms, tokios kaip vaizdo žaidimų turinio generavimui Vaizdų apdorojimas leidžia turėti dinaminį žemėlapį, kuris kinta priklausomai nuo žaidėjo ir jo veiksmų. Tai puikiai tinka vaidmenų žaidimų žemėlapių kūrimui, nes tokie žaidimai dažniausiai turi didžiulį, tačiau statinį žaidimo žemėlapį [17].

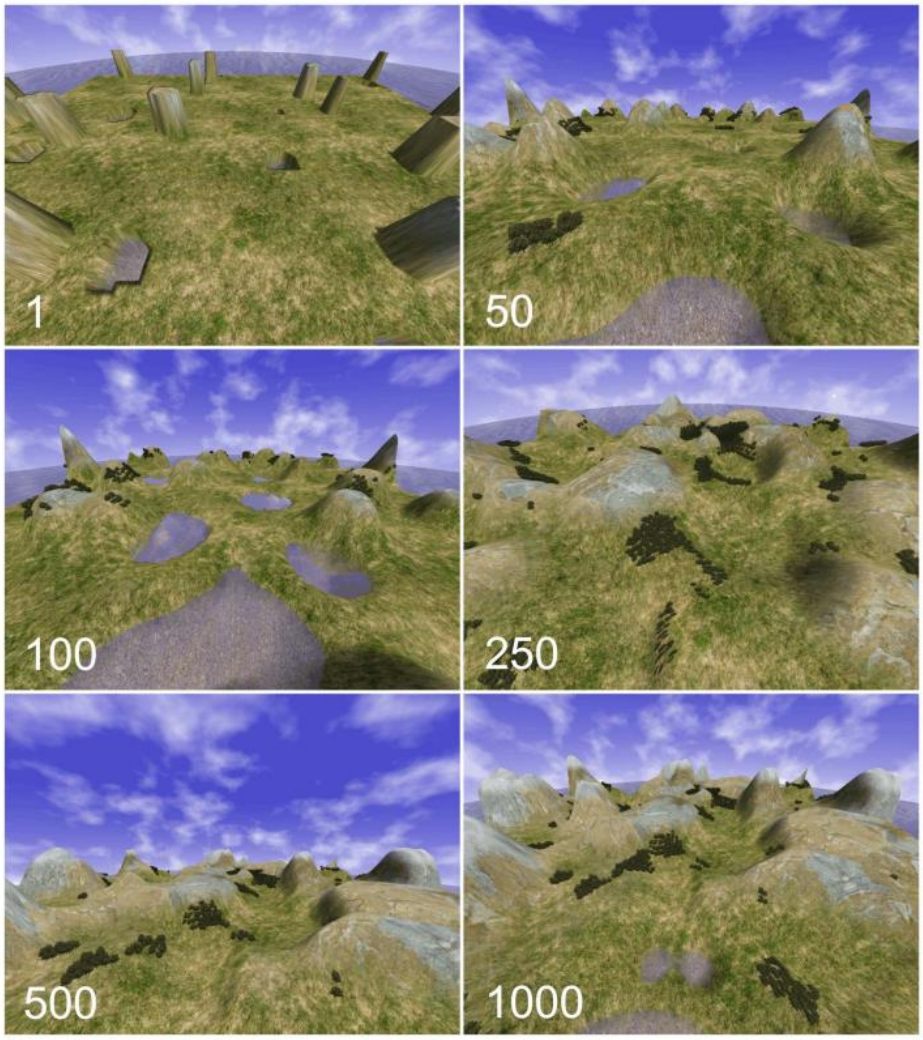
1.1.1. Pažengusio vaizdų apdorojimo metodo veikimas

Vaizdų apdorojimo žemėlapių kūrime esmė — jėgų šaltiniai esantys pasaulyje gali keisti aplink juos supančią aplinką. Šie šaltiniai yra objektai, kurie gali keisti poziciją ir būseną. Aplink juos esanti aplinka gali reaguoti ir keisti žemėlapį ir jo reljefą. Algoritme yra elementai R, C, M, Y bei jėgos FR, FC, FM, FY, kurios gali keisti žemėlapio reljefą. Po to įtraukiami greitos aproksimacijos Gauso filtrai bei medianiniai filtrai, kad rezultatai atrodytų natūralesni ir panašesni į tikro pasaulio reljefą [17].



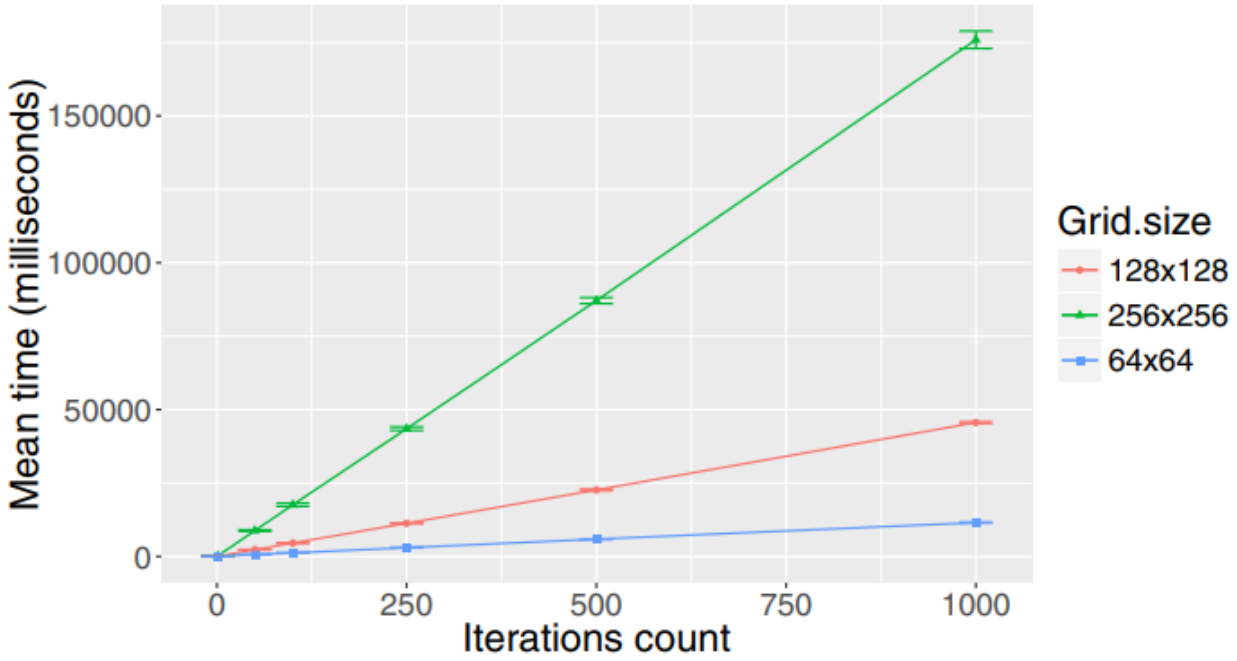
6 pav. Žemėlapio kitimas priklausomai nuo jėgų ir iteracijų [17]

1.1.2. Paženguso vaizdų apdorojimo metodo rezultatai



7 pav. Žaidimo žemėlapis gautas naudojant pažengusį vaizdų apdorojimą [17]

Naudojant pažengusį vaizdų apdorojimą kuriant žaidimų žemėlapius gaunamas atsitiktinis žemėlapio reljefas kuris kyla arba leidžiasi į viršų. Šiuo metodu yra sukuriamas tik reljefas, o ne objektais užpildytas pilnas žemėlapis. Vis dėlto, norint išgauti pakankamai tikrovišką reljefą, metodas turi praeiti nemažą kiekį iteracijų, kas gali užtrukti nemažai laiko, ypač didelio masto žemėlapiams.



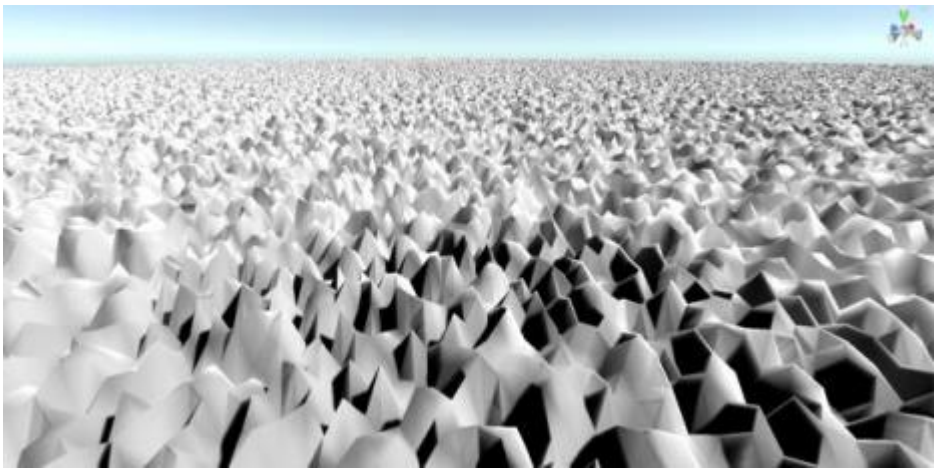
8 pav. Žemėlapio generavimo laikas priklausomai nuo dydžio ir iteracijų [17]

Metodas taip pat yra pakankamai naujas, todėl nėra lengva rasti informacijos kaip jį tobulinti ar dokumentacijos kaip jį reikia įgyvendinti ar konfigūruoti. Nors tai ir leidžia turėti dinamiškai kintantį žemėlapį, šaudyklės žanro žaidimams tai nėra ypač aktualu, o sąlyginai ilgas generavimo laikas trukdytų žaidimo eigą.

1.2. Atsitiktinių aukščių metodas

Atsitiktinių aukščių metodas yra paprastas algoritmas, kuris kiekvienam žemėlapio taškui, priklausomai nuo jo dydžio, sugeneruoja atsitiktinį aukštį pasirinktose ribose. Baigus skaičiuoti aukščius, jie yra nusiunčiami reljefo objektui kuris sukuria žemėlapį pagal pateiktus aukščius [18].

1.2.1. Atsitiktinių aukščių metodo rezultatai



9 pav. Žemėlapis sukurtas pagal atsitiktinių aukščių metodą [18]

Kadangi algoritmas yra paprastas ir greitas, jis sukuria gan nepatrauklų žemėlapį. Sugeneruotas žemėlapis neturi jokios logikos ar natūralumo, nes kiekvienas žemėlapio taškas yra tiesiog atsitiktinai

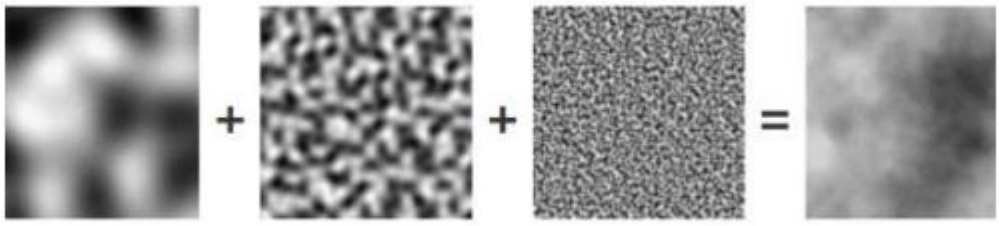
sugeneruotas aukštis. Tokio tipo žemėlapį būtų sunku pritaikyti žaidimui, kuriame yra valdomas veikėjas, nes reljefas yra nepatogus naviguoti. Algoritmas taip pat negeneruoja objektų ar kitaip užpildo žaidimo žemėlapį.

1.3. Perlin triukšmo metodas

Perlin triukšmo žemėlapiai yra sukurti iš banguojančių duomenų, žemėlapiui suteikiant glotnumo ir nuspėjamumo. Algoritmas naudoja šiuos parametrus:

- X ir Y skalės: bangų aukščiui ir dažniui nustatyti;
- Oktavos: bangų skaičiui nustatyti;
- Išsilaikymas: kaip stipriai turėtų kisti dydžiai tarp bangų;
- X ir Y atsvara: rezultato kintamumui nustatyti;
- Aukščio skalė: kaip stipriai pabrėžti rezultatų mastą.

Naudojant Perlin metodą vieną kartą, gaunamas glotnus žemėlapis. Tai galima pakeisti Perlin metodą naudojant daugiau nei vieną kartą su skirtingais masteliais, taip gaunant Fraktalinį triukšmą [18].



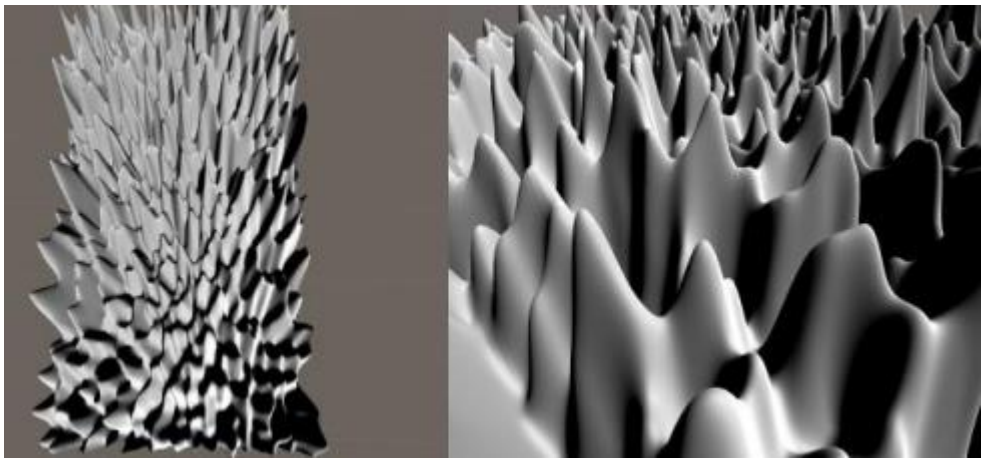
10 pav. Fraktalinio triukšmo generavimas [18]

1.3.1. Perlin triukšmo metodo rezultatai

Kaip jau minėta ankščiau, panaudojus Perlin metodą vieną kartą, gaunamas glotnus ir nuspėjamas žemėlapio reljefas, kuris neturi daug variacijos.

11 pav. Glotnus žemėlapis gautas naudojant Perlin metodą vieną kartą [18]

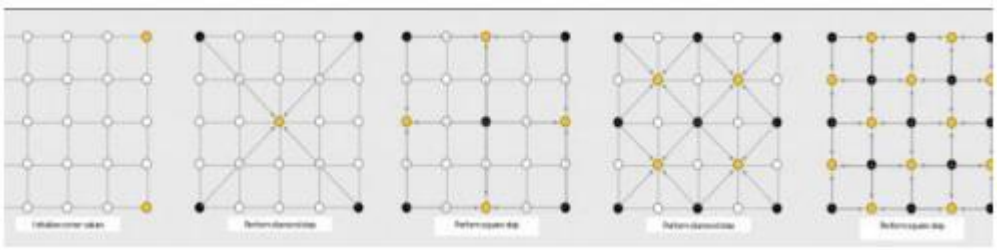
Panaudojus Perlin metodą daugiau nei vieną kartą, taip sukuriant Fraktalinį triukšmą, gaunamas daug ryškesnis ir įvairesnis reljefas, tačiau iškyla kita bėda, kad žemėlapis beveik nebeturi nei trupučio lygių vietų, kur galėtų laisvai vykti žaidimo veiksmas. Žemėlapis tapo per daug spygliuotas, nei vienas žemėlapio pikselis neliko nepakeistas ir nėra tinkamas pirmojo asmens šaudyklės žaidimui ar vaikščiojantiems veikėjams žaidime.



12 pav. Perlin metodo rezultatas panaudojus metodą daugiau negu vieną kartą [18]

1.4. Vidurio taško perkėlimo metodas

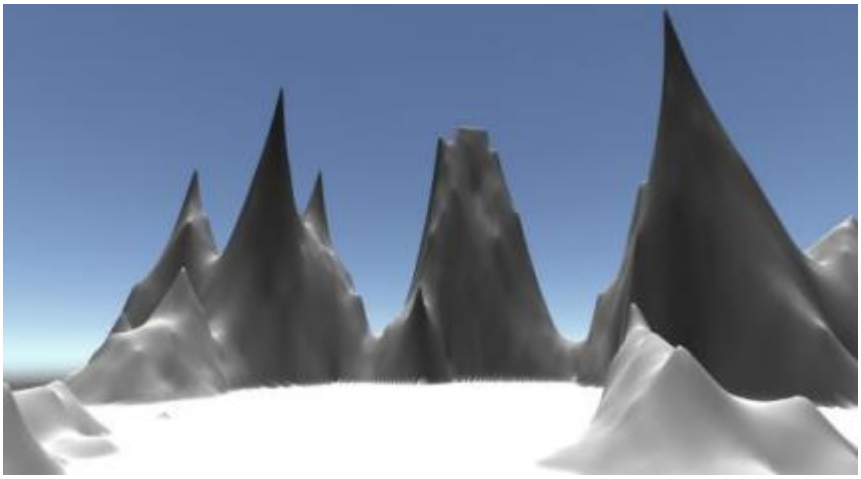
Dar kitaip žinomas kaip deimanto-kvadrato metodas, šis metodas paima dviejų dimensijų aukščio žemėlapį ir atsitiktiniai sugeneruoja aukščio reikšmes iš 4 pozicijų pateiktų ant aukščio žemėlapio. Šios pozicijos gali būti išdėstytos kvadrato arba deimanto formos rekursyviai, taip kiekvieną žingsnį didinant aukščio reikšmę. Kiekvienam taškui aukščio reikšmė yra apskaičiuojama paimant praeitų 4 taškų vidurkio ir atsitiktinai sugeneruoto skaičiaus sumą [18].



13 pav. Deimanto-kvadrato metodo žingsniai [18]

1.4.1. Vidurio taško perkėlimo metodo rezultatai

Naudojant vidurio taško perkėlimo metodą, sugeneruojamas žemėlapis, kuris turi daug plokščios vietos, kurioje gali vykti žaidimo veiksmas, tačiau su spygliuotais kalnais kurie būtų kaip žaidimo ribos, kraštovaizdis ar priedanga nuo priešų šūvių. Naudojant maksimalią aukščio reikšmę bei šiurkštumo reikšmę galima toliau konfigūruoti gautus rezultatus ir išgauti dar geresnį vaizdą, kuris tiktų šaudyklės žanro žaidimui. Vis dėlto, toks metodas gali išgauti atsitiktinius rezultatus, kuriuos ne visada galima būtų ištestuoti ir žinoti, kad žaidimas visada bus peržaidžiamas ir be rimtų klaidų.



14 pav. Vidurio taško perkėlimo metodo rezultatas [18]

1.5. Procedūrinis labirinto metodas

Šis metodas sukuria labirintus iš jau paruoštų įvairių kambarių objektų. Iš masyvo atsitiktine tvarka paimant vieną iš masyve esančių kambarių objektų, po vieną kambarį lipdomas labirintas. Kiekvienas kambarys turi vieną arba daugiau durų, ties kuriomis gali būti toliau lipdomas labirintas. Labirinto dydį nusako apibrėžtas dydžio skaičius arba iki kol labirintas nebeturi kur padėti sekančio kambario. Tokiu atveju, kambario durys yra užveriamos ir pro jas žaidėjas negali praeiti, kad nebūtų galimybės išeiti iš labirinto ribų [19].

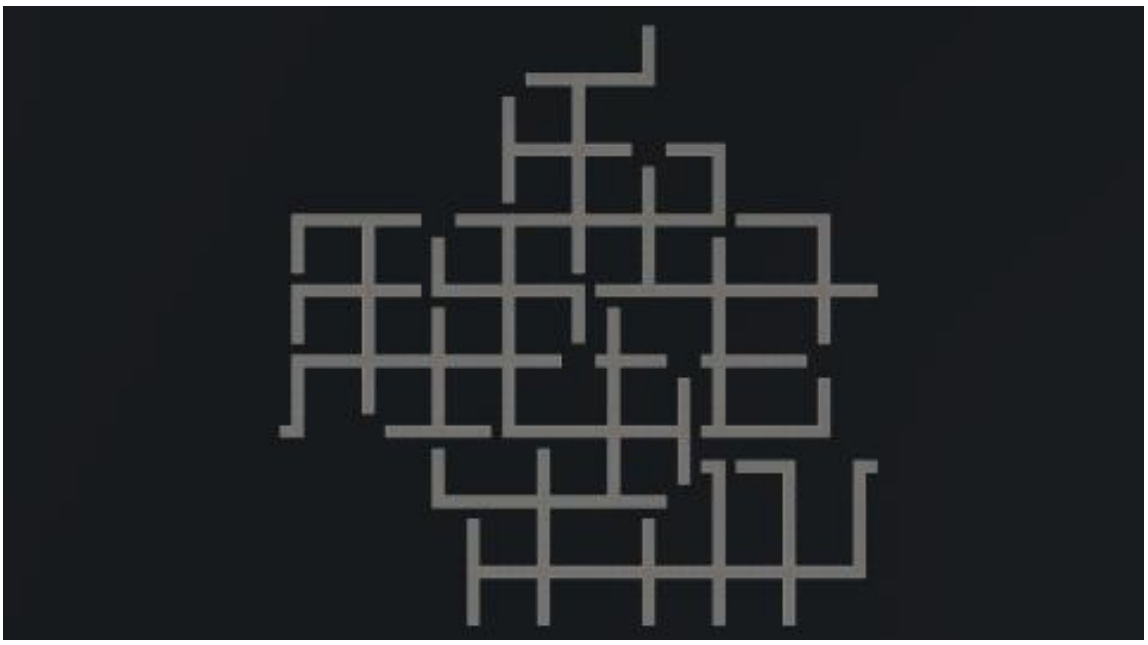
Kiekvieno masyve esančio kambario durys turi būti tokio pačio dydžio, kad durys galėtų būti sujungtos su sekančiu kambariu ir žaidėjas negalėtų matyti už labirinto ribų. Tokie masyvai gali sugeneruoti lygį prieš žaidimui prasidedant, arba generuoti kambarius žaidimo veikėjui keliaujant per jau sugeneruotus kambarius. Taip galima sukurti ir niekada nesibaigiantį labirintą. Norint, kad žaidimo lygis turėtų vertikalumo, į masyvą reikia įdėti kambarių, kurie turėtų laiptus ar kitą būdą

veikėjui keliauti vertikaliai, o kambario durys turi būti aukščiau arba žemiau negu kambario pradinis taškas.

15 pav. Įvairių formų ir durų skaičiaus kambariai [19]

1.5.1. Procedūrinis labirinto metodo rezultatai

Kadangi metodas naudoja jau sukurtus kambarius ir negeneruoja reljefo pats, tai yra puikus metodas kurti atsitiktinius žaidimo lygius šaudyklės žanro žaidimui, nes kiekvienas kambarys gali būti ištestuojamas atskirai, įsitikinant, kad jis yra peržaidžiamas ir neturi klaidų. Labirintą taip pat yra lengva atnaujinti papildomais kambariais, kadangi viskas ką tereikia padaryti, tai sukurti papildomų kambarių ir įkelti juos į masyvą. Kiekvienas kambarys taip pat gali turėti savo specifinę tematiką, pavyzdžiui, vienas iš kambarių yra žaidimo boso kambarys, kurį įveikus laimimas žaidimas. Taip pat kiekvienas kambarys gali turėti savo lokalų turinio generavimą, kuris išdėstytų priešus, objektus ar daiktus, reikalingus žaidėjui, kambaryje.



16 pav. Procedūrinio labirinto metodu sukurtas lygis [19]

1.6. Analizės išvados

Atlikus turinio generavimo metodų analizę, buvo pasirinktas procedūrinis labirinto metodas projektuojamos sistemos lygių generavimui. Visi kiti metodai yra geriau pritaikomi didelių žemėlapių reljefo kūrimui. Tai padeda sutaupyti laiko kuriant didžiulius žemėlapius, tačiau po to juos vis dar reikia užpildyti objektais, o tam būtų reikalingi papildomi kompleksiški algoritmai. Taip pat atsitiktinai sugeneruotas reljefas turi daugiau galimybių būti neperžaidžiamas ar turėti stambių klaidų žaidime, jo generavimą yra sunkiau ištestuoti. O procedūrinis labirinto metodas yra paprastas, greitas ir reikalaujantis mažiau išteklių testavimui. Kadangi į masyvą yra paduodami jau sukurti kambariai, kiekvieną kambarį galima sukurti pagal savo norus, kiekvienam iš jų pritaikyti papildomą turinio generavimo algoritmą, kuris dar labiau paįvairintų žaidimo lygius, o atsitiktinio turinio generavimo testavimas reikalautų tik patikrinti kiekvieną kambarį ir ar jų durys lygiai susijungia su kitų kambarių durimis. Žaidimą būtų lengvą atnaujinti, į masyvą pridedant papildomų kambarių. Taip pat yra lengva valdyti žaidimo lygio dydį, nusakant kiek kambarių reikėtų sugeneruoti. Taigi procedūrinis labirinto metodas yra puikus kompromisas tarp atsitiktinai sugeneruoto turinio ir jo testavimo bei valdymo galimybių.

2. Projektinė dalis

Magistrinio darbo metu buvo realizuotas pirmojo asmens daugelio žaidėjų tinkle šaudyklės žaidimo projektas. Žaidimas buvo sukurtas naudojant „Unity“ žaidimų variklį. Projektas buvo įgyvendintas kaip grupinis darbas kartu su kolega Domu Raulinaičiu, darbus išsiskirsčius posistemis. Žaidimas buvo realizuotas naudojant „Mirror“ bei „Steamworks“ tinklo bibliotekas, todėl žaidimas funkcionuoti gali tik naudojant „Steam“ platformą kaip žaidimo paleidimo programą.

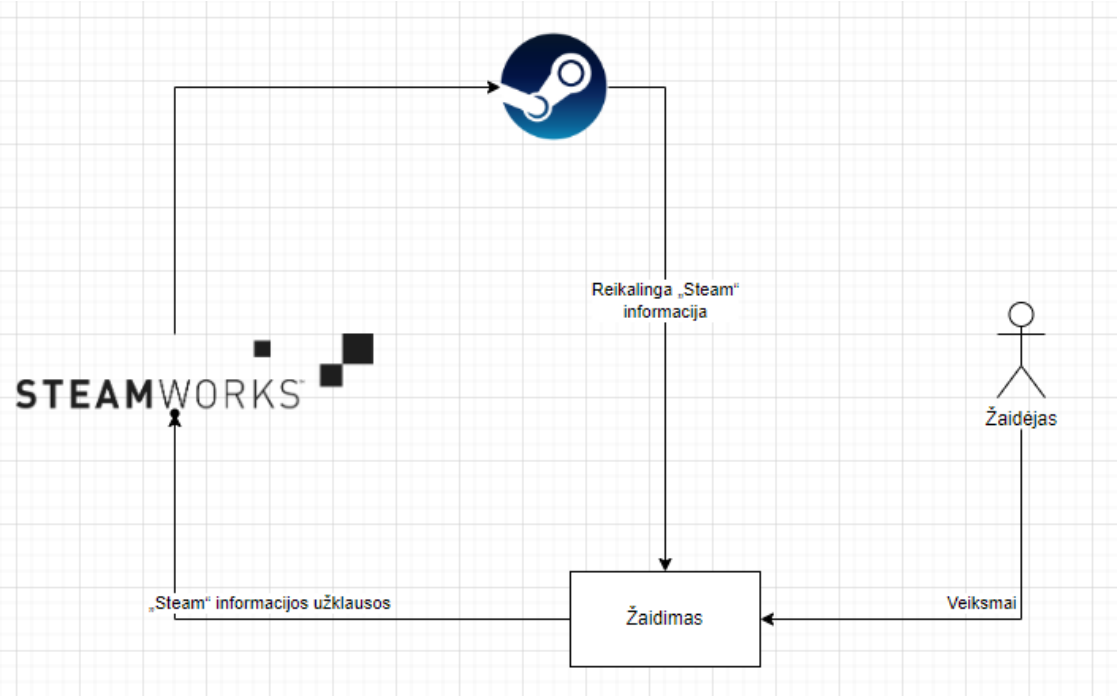
17 pav. Projekto metu sukurtas žaidimas

2.1. Esama projekto padėtis

Šiuo metu žaidimo projektas yra funkcionaliai baigtas, visos suprojektuotos funkcijos buvo įgyvendintos. Žaidimas yra uždaroje klaidų testavimo ir tvarkymo stadijoje, prieinamas tik projektą kūrentiems žmonėms, dar kitaip, alfa versijoje. Žaidimas šiuo metu nėra patalpintas ir yra neprieinamas „Steam“ platformos parduotuvėje kaip parduodamas žaidimas. Žaidimas naudoja „Steam“ suteiktą testavimo identifikatorių, kad būtų galima ištestuoti su „Steam“ platforma susijusias funkcijas, žaidimui nebūnant patalpintam platformoje.

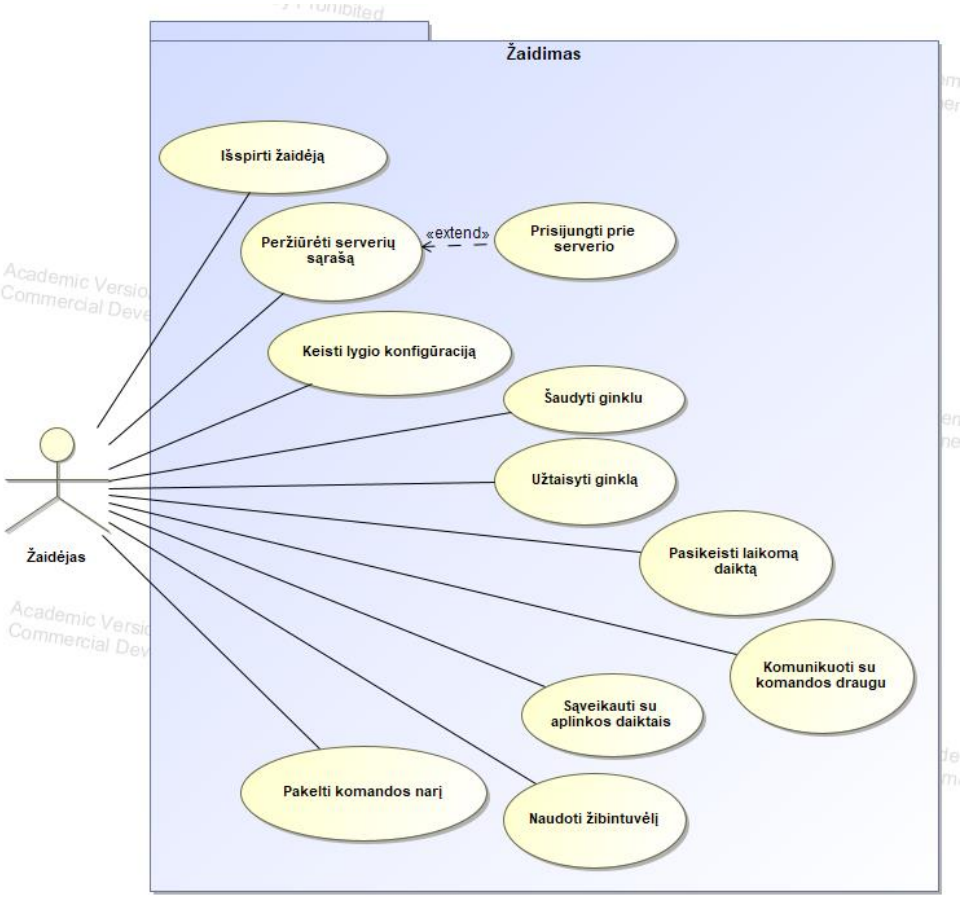
2.2. Veiklos kontekstas

Žaidimas atpažįsta žaidėjo veiksmus kaip įvestį, apskaičiuoja objektų ir viso pasaulio fizikines savybes, koordinates ir būseną. Reikalinga informacija ir užklausos yra siunčiamos naudojant „SteamWorks.net“ į „Steam“ platformą, kuri žaidimui atgal siunčia atnaujintą reikalinga informaciją. Tai gali būti draugų sąrašai, serverių sąrašai, naudotojų vardai ir kita.



18 pav. Veiklos konteksto diagrama

2.3. Panaudojimo atvejai



19 pav. Panaudojimo atvejų diagrama

Įsijungus žaidimą naudotojas patenka į pradinio menių sceną. Šioje scenoje mano įgyvendinti panaudojimo atvejai yra:

- *Išspirti žaidėją* – naudotojas būdamas savo sukurtame serveryje turi galimybę išspirti nepageidaujamus žaidėjus iš serverio fojė paspausdamas mygtuką esantį šalia žaidėjo. Tik serverį sukūręs asmuo turi galimybę matyti ir naudoti šį mygtuką.
- *Peržiūrėti serverių sąrašą* – naudotojas būdamas menių scenoje gali paspausti mygtuką, kuris įjungs visų žaidimo serverių sąrašo langą ir naudotojas galės pasirinkti prie kurio serverio nori prisijungti.
- *Prisijungti prie serverio* – naudotojas būdamas visų žaidimo serverių sąrašo lange gali paspausti prisijungimo mygtuką esantį prie kiekvieno serverio, taip nusiųsdamas užklausą serveriui, kad bandoma prisijungti.
- *Keisti lygio konfigūraciją* – naudotojas būdamas savo sukurtame serveryje turi galimybę keisti lygio konfigūraciją prieš pradedamas žaidimas. Į tai įeina serverio pavadinimas, atsitiktinumo sėklos skaičių seka, lygio dydis, žaidimo režimas, lygio sunkumas bei papildomų žaidimo modifikavimų kaip „Tik galvos šūviai“ ir „Tamsusis režimas“ įjungimas.

Prisijungus prie serverio ir patekus į žaidimo sceną, naudotojas gali atlikti šias funkcijas:

- *Šaudyti ginklu* – naudotojas, žaidimo veikėjui rankose turint ginklą su pakankamai amunicijos, gali šauti ginklu į taikomą vietą ir žaloti priešus.
- *Užtaisyti ginklą* – naudotojas, žaidimo veikėjui rankose turint ginklą kurio apkaboje nėra maksimalus amunicijos kiekis, gali užsitaisyti ginklą, taip paleidžiant užtaisymo animaciją ir papildant ginklo apkabą amunicija.
- *Pasikeisti laikomą daiktą* – naudotojas, žaidimo veikėjui turint daugiau nei vieną ginklą ar kitą daiktą, gali keisti žaidimo veikėjo laikomą daiktą jo rankose ir tada juo naudotis.
- *Komunikuoti su komandos draugu* – naudotojas, būnant žaidimo eigos scenoje, gali paspausti komunikavimo mygtuką, kuris atveria garso kanalą ir pradeda siųsti mikrofonu įrašomą garsą kitiems žaidėjams serveryje.
- *Sąveikauti su aplinkos daiktais* – naudotojas, būnant žaidimo eigos scenoje, gali sąveikauti su aplinkos daiktais paspausdamas specifinį mygtuką. Tai yra sąveikavimas su mygtukais žaidime, norint atidaryti duris, amunicijos ar gyvybių paėmimas nuo žemės.
- *Naudoti žibintuvėlį* – naudotojas, būnant žaidimo eigos scenoje, gali paspausti mygtuką ir įjungti arba išjungti žibintuvėlį, jeigu žibintuvėlis turi pakankamai energijos.
- *Pakelti komandos narį* – naudotojas, kurio žaidimo veikėjas yra šalia kito veikėjo, kuriam baigėsi gyvybės, gali paspausti mygtuką ir pakelti komandos narį, taip atstatant jo gyvybes ir leidžiant jam vėl vaikščioti.

2.4. Sistemos reikalavimai

2.4.1. Reikalavimai sistemos išvaizdai

- Sistemos tekstai privalo neturėti rašybos ar gramatinių klaidų, nes jos gali parodyti žaidimą kaip neprofesionalų ar net klaidinti žaidėjus.

2.4.2. Naudojimosi paprastumo reikalavimai

- Žaidėjai, turintys bent minimalią pirmo asmens šaudyklių patirtį, turi sugebėti naudotis sistema, norint pritraukti didžiausią įmanoma auditoriją.

2.4.3. Prieinamumo neįgaliesiems reikalavimai

- Sistema turi galėti naudotis kurtieji naudotojai, nes žaidimas gali funkcionuoti pilnai vien tik vizualiais indikatoriais.

2.4.4. Reikalavimai užduočių vykdymo greičiui

- Bet kokia sąveika tarp naudotojo ir sistemos turi užtrukti maksimaliai 10 sekundžių, nes per ilgas sistemos atsako laikas gali atbaidyti naudotoją.

2.4.5. Atsparumo trukdžiams ir klaidoms reikalavimai

- Dingus interneto ryšiui, žaidėjams turi būti pranešta apie trūkstamą internetą ir žaidimas turi būti stabdomas, nes užstrigusi be aiškios priežasties sistema gali pykdyti žaidėjus.

2.4.6. Numatomos fizinės aplinkos reikalavimai

- Sistema numatoma naudotis atsisėdus prie stalo, su kompiuteriu, interneto ryšiu, pele, klaviatūrą, garso aparatūrą ar ausinėmis, monitoriumi, mikrofonu.

2.4.7. Prieigos reikalavimai

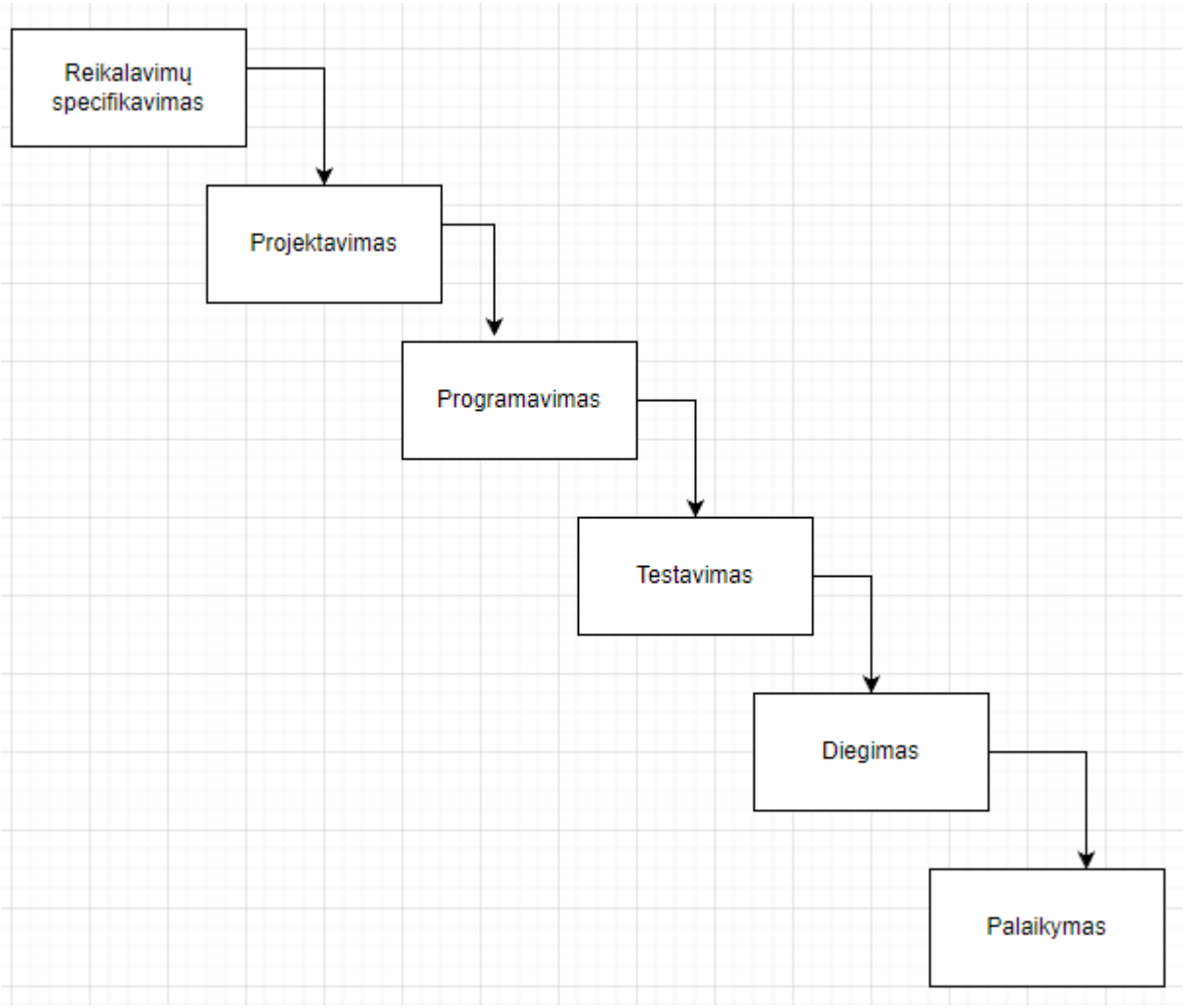
- Žaidimą gali žaisti tik tie naudotojai, kurie turi „Steam“ paskyrą, nes žaidimas reikalauja „Steam“ platformos funkcijų, kad pilnai funkcionuotų.

2.4.8. Atitikties standartams reikalavimai

- Sistema privalo atitikti „Steam“ keliamus standartų reikalavimus, kadangi žaidimas planuojamas būti patalpintas į „Steam“ žaidimų parduotuvę.

2.5. Sistemos kūrimo procesas

Sistemos kūrimui buvo pasirinktas krioklio metodas, nes sistema buvo aiškiai specifikuota ir projektuota proceso pradžioje, buvo aiškus galutinis rezultatas ir nebuvo laiko ar reikalo eksperimentuoti. Kiekvienas žingsnis turėjo būti atliktas ir atsiskaitytas iki tam tikros datos, todėl ir projektas buvo kuriamas nuosekliai.



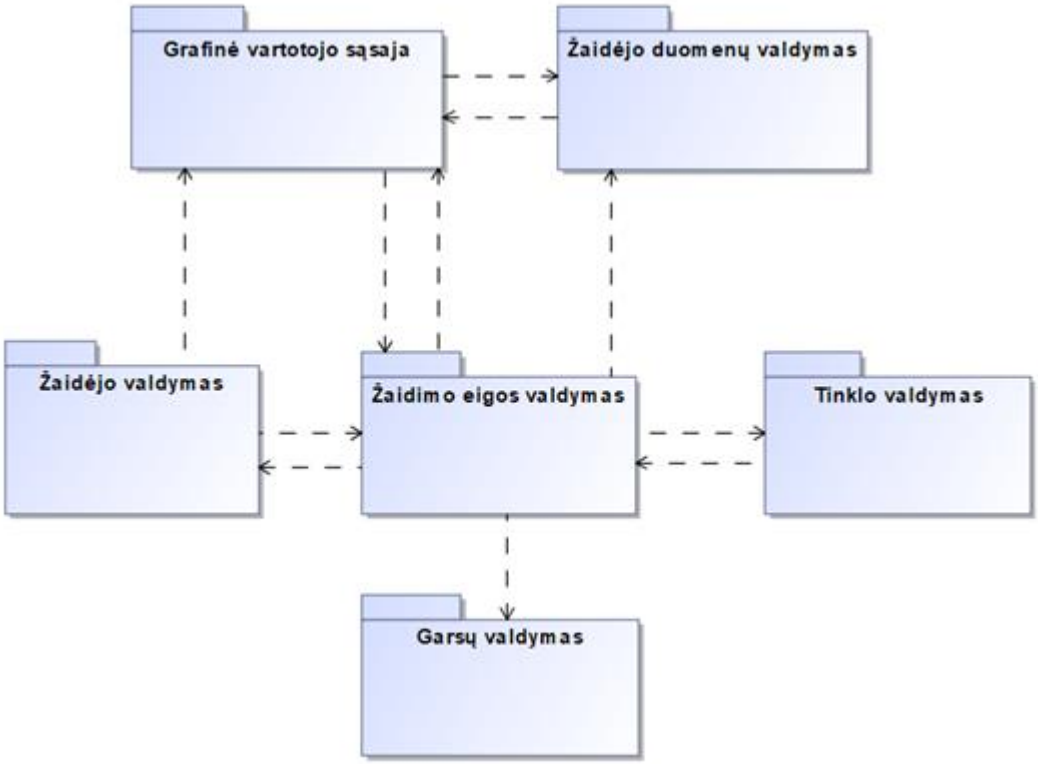
20 pav. Krioklio kūrimo procesas

2.6. Paketų diagramos

2.6.1. Visos sistemos paketų diagrama

Žaidimo sistema susideda iš 6 paketų. Iš šių 6 paketų, 3 iš jų buvo paskirti man kaip pagrindiniai paketai ties kuriais reikia dirbti:

- Žaidėjo valdymas;
- Žaidimo eigos valdymas;
- Garsų valdymas;



21 pav. Visos sistemos paketų diagrama

2.6.2. Paketų detalizavimas

22 pav. Žaidėjo valdymo paketo klasių diagrama

Žaidėjo valdymo paketas atsakingas už žaidėjo veiksmų valdymą žaidimo eigos metu – žaidėjo judėjimą, pašokimą, šaudymą, daiktų panaudojimą, komunikaciją su komandos draugu, ginklų užtaisymą, sąveikavimą su aplinkos daiktais, daiktų rankose keitimą bei komandos draugų pakėlimą.

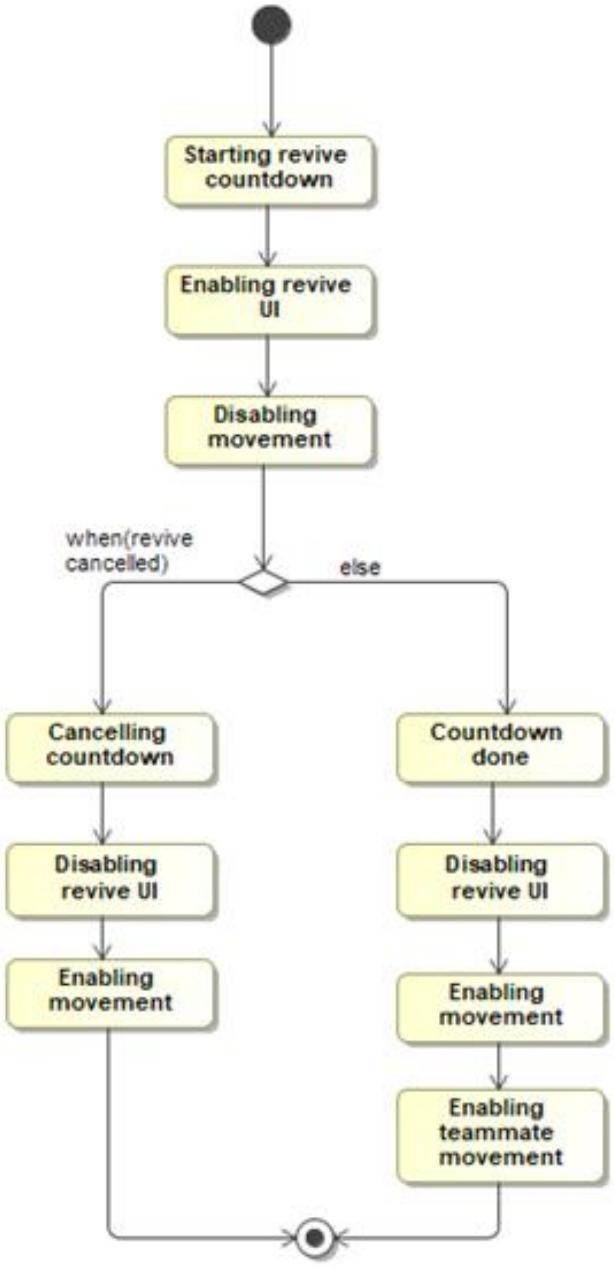
23 pav. Žaidimo eigos valdymo paketo klasių diagrama

Žaidimo eigos valdymo paketas atsakingas už lygių ir aplinkos daiktų generavimą, priešų valdymą bei žaidėjų, priešų ir misijų būsenas ir statistiką.

24 pav. Garsų valdymo paketo klasių diagrama

Garsų valdymo paketas atsakingas už žaidime leidžiamus garsus, garso efektus ir foninę muziką bei garso kanalus tarp komunikuojančių komandos narių.

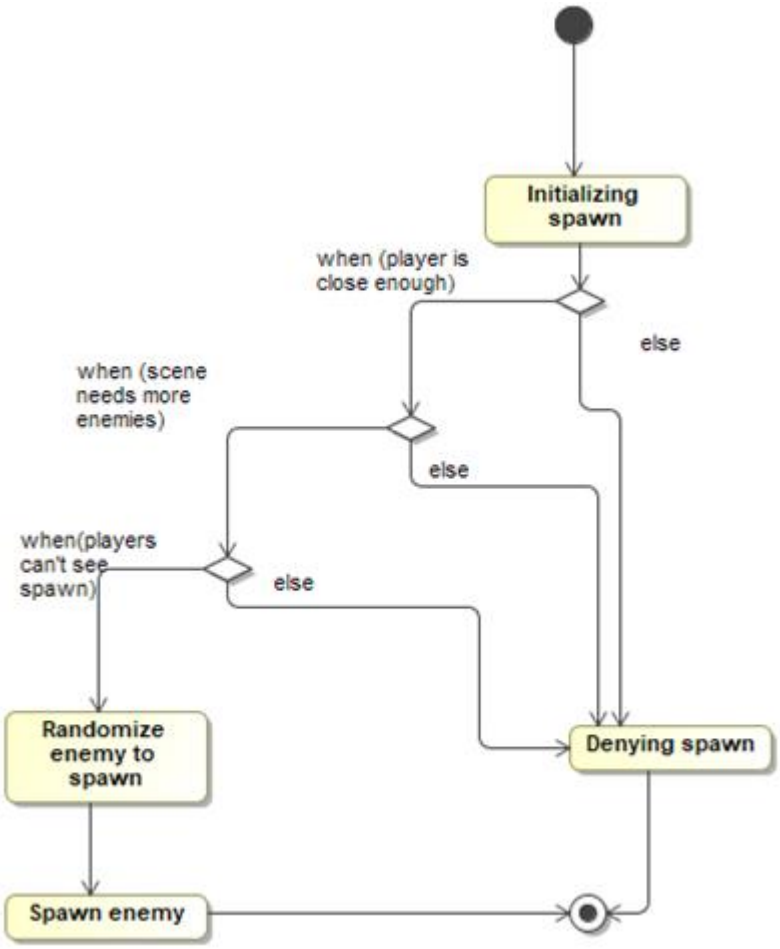
2.7. Būsenų diagramos



25 pav. Komandos nario pakėlimo būsenos diagrama

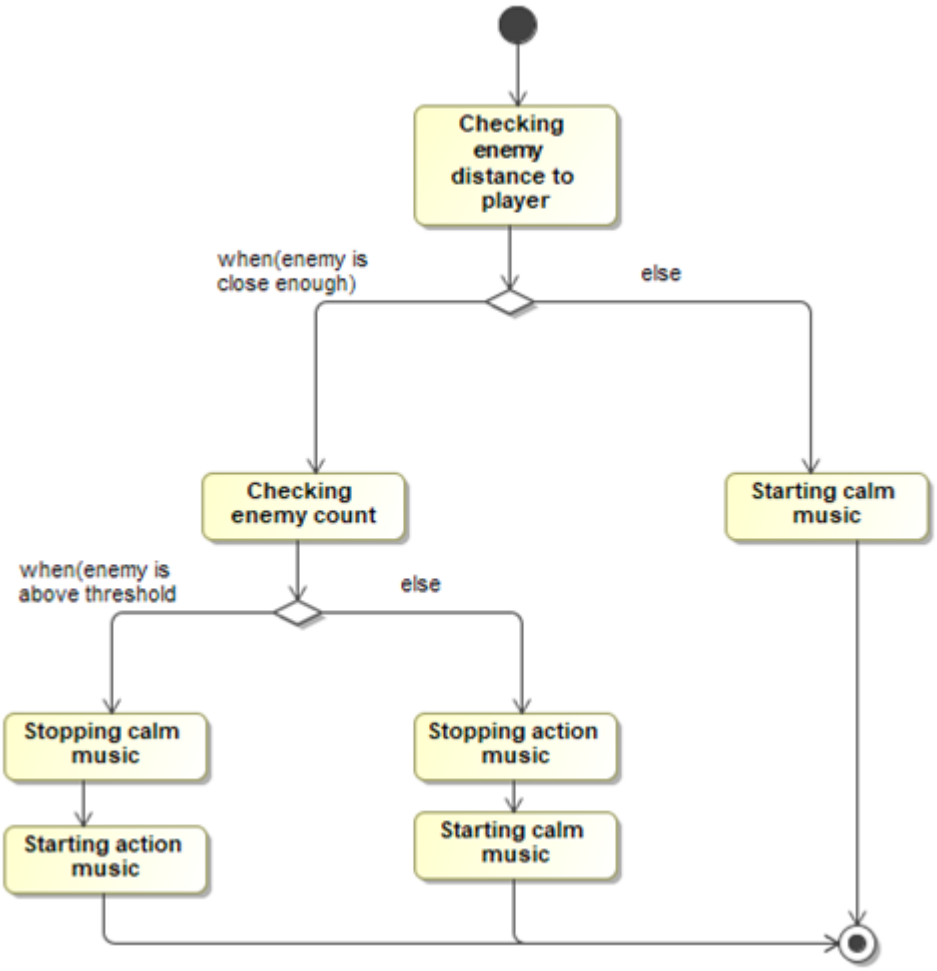
Komandos nario pakėlimo metu pradedamas skaičiavimas iki komandos nario pakėlimo, įjungiamas grafinis indikatorius, išjungiamas veikėjo judėjimo kontroliavimas. Atšaukus pakėlimą iki jam pasibaigus, skaičiavimas yra atšaukiamas, išjungiamas indikatorius ir įjungiamas judėjimas.

Priešingu atveju, pasibaigus skaičiavimui, abu žaidėjai gauna kontrolę, išjungiamas indikatorius ir baigiamas skaičiavimas.



26 pav. Priešų pridėjimo būsenos diagrama

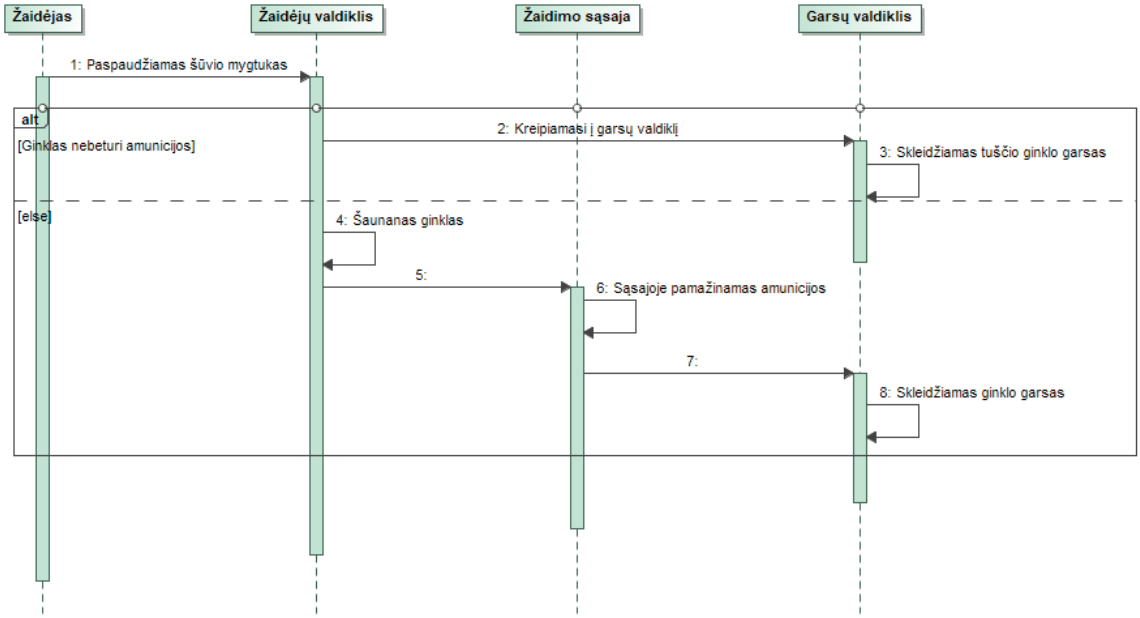
Priešų pridėjimo į žaidimą metu vyksta patikrinimai ar žaidėjas yra pakankamai arti, kad priešai nebūtų per toli nuo veiksmo, ar scenoje reikalinga daugiau priešų, kad būtų priešų limitas ir būtų palaikomas žaidimo sunkumas ir ar žaidėjai negali matyti atsirandančių priešų, kad nebūtų sugadinta žaidimo atmosfera. Atsitiktinai pasirenkamas priešas ir jis sukuriamas scenoje. Priešingais atvejais atšaukiamas priešo kūrimas.



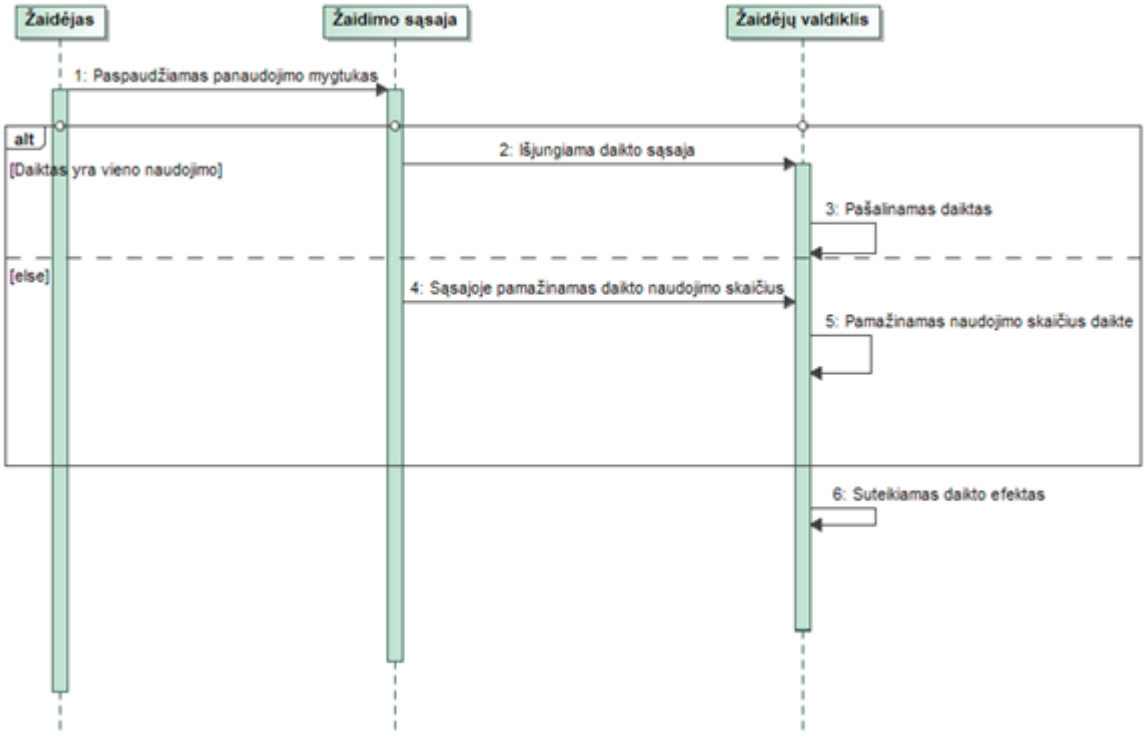
27 pav. Muzikos grojimo būsenos diagrama

Muzikos grojimas žaidime prasideda nuo priešų atstumų tikrinimo iki žaidėjo. Priešams esant toli, grojama rami muzika. Priešams esant arti, tikrinama kiek priešų yra. Jeigu priešų yra nedaug, stabdoma veiksmo muzika ir grojama rami muzika. Jeigu priešų yra pakankamai, stabdoma rami muzika ir grojama veiksmo muzika.

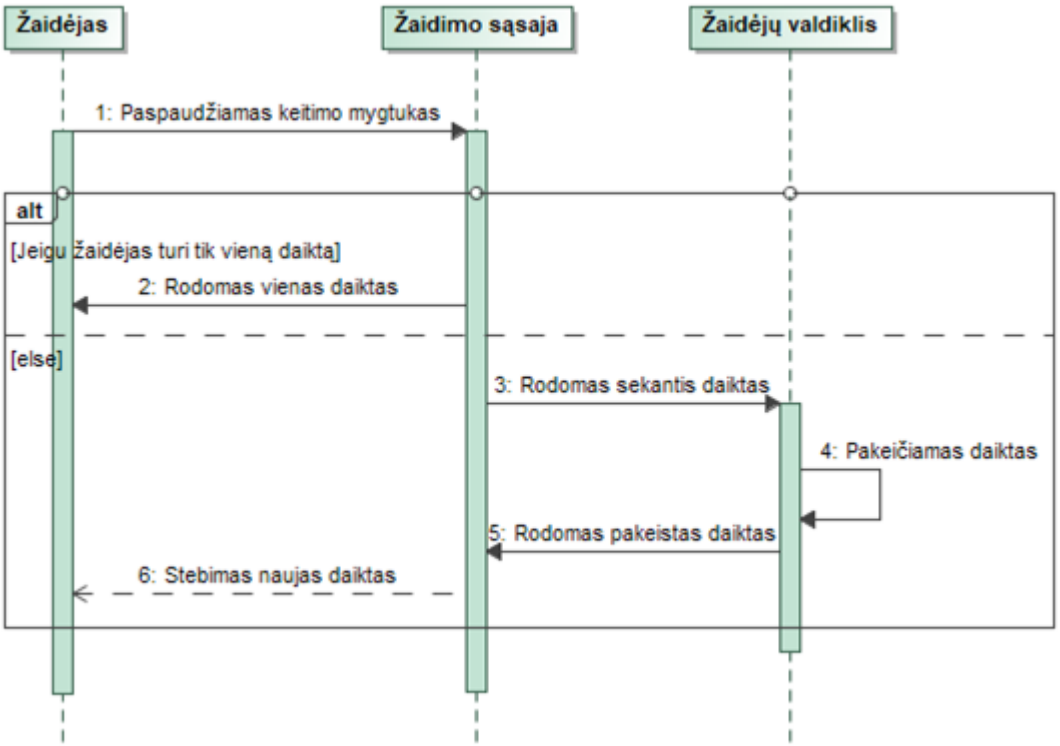
2.8. Sekų diagramos



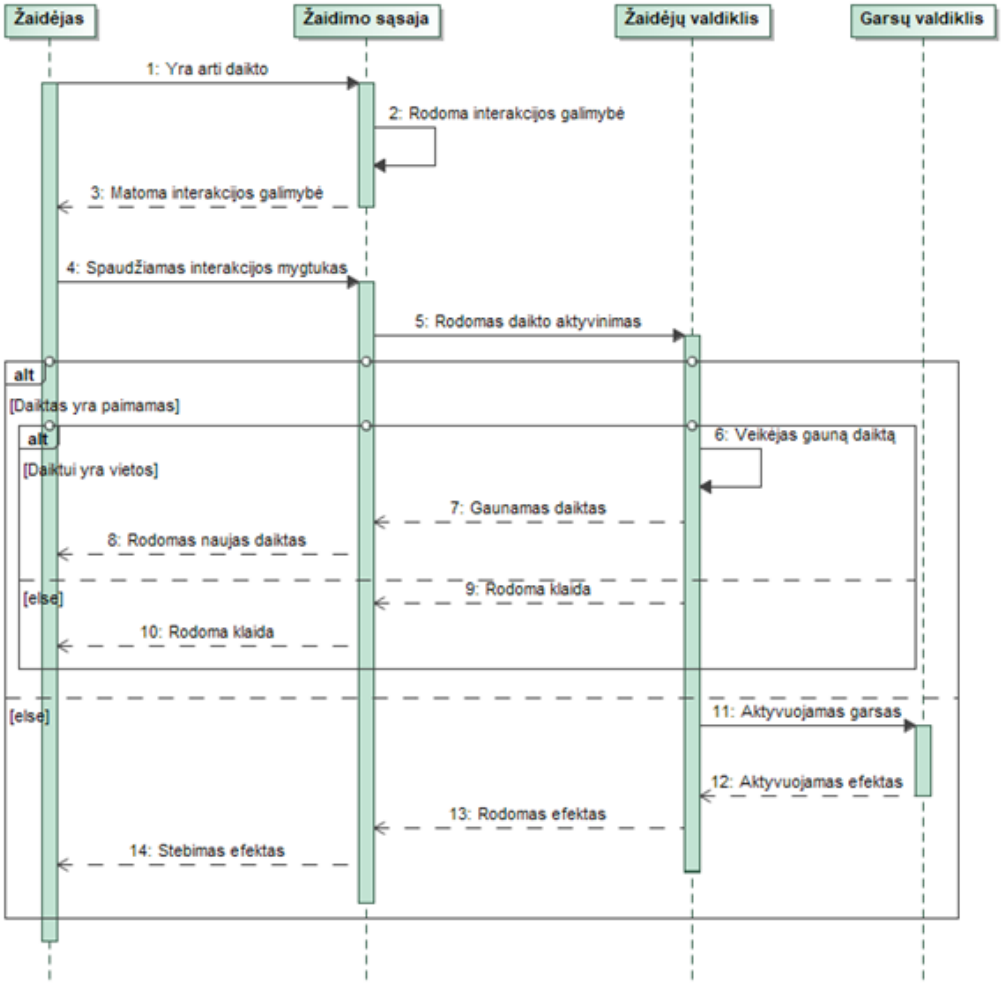
28 pav. Šaudymo ginklu sekų diagrama



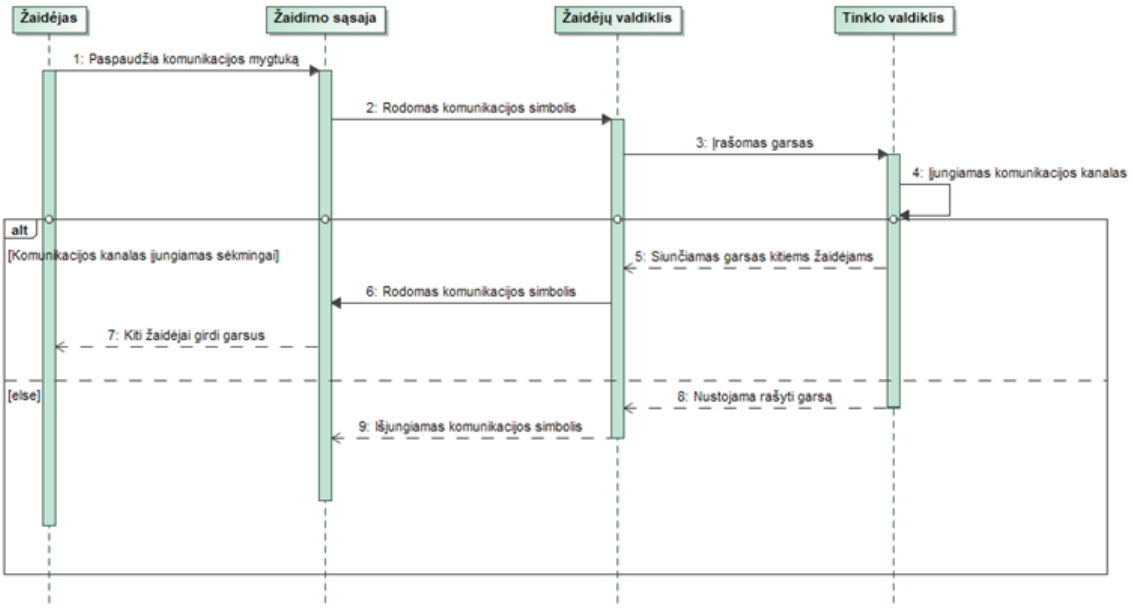
29 pav. Daikto paėmimo sekų diagrama



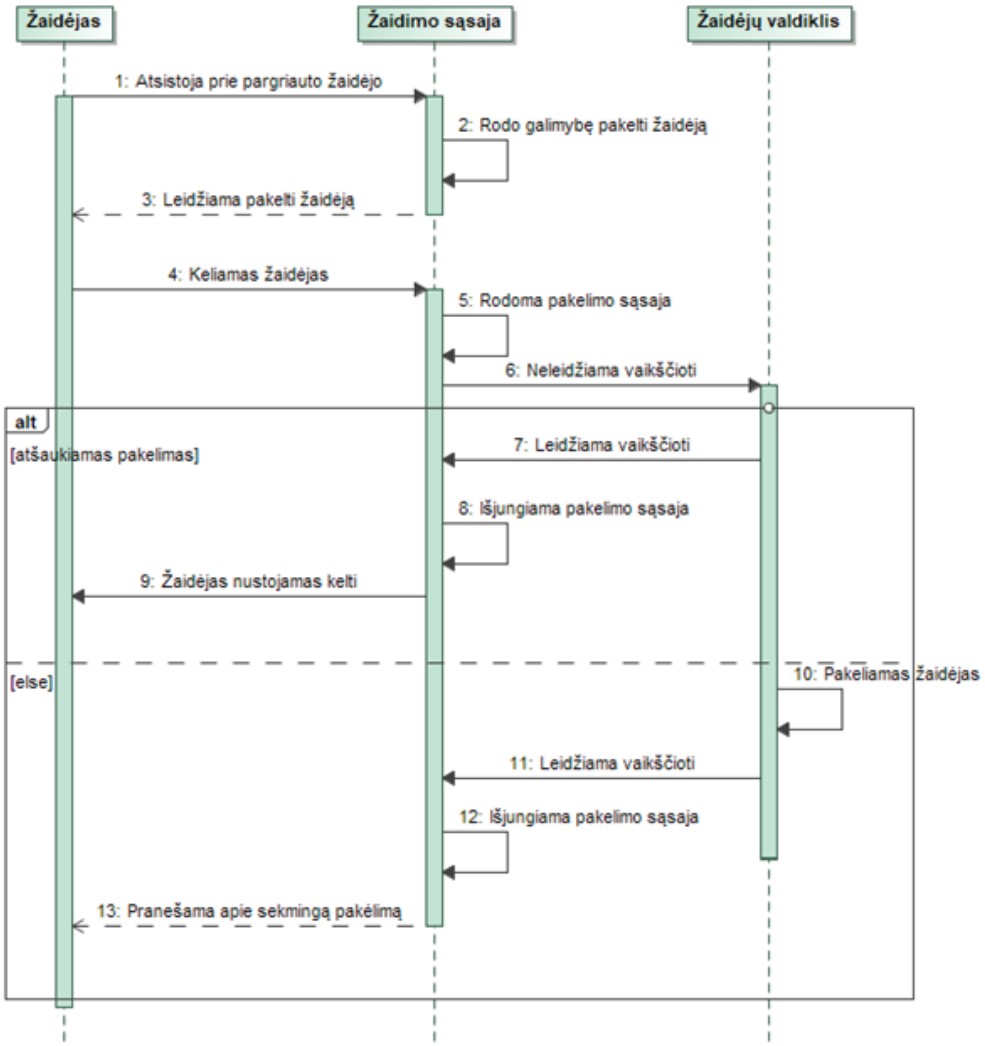
30 pav. Laikomo daikto pakeitimo sekų diagrama



31 pav. Sąveikavimo su aplinkos daiktais sekų diagrama

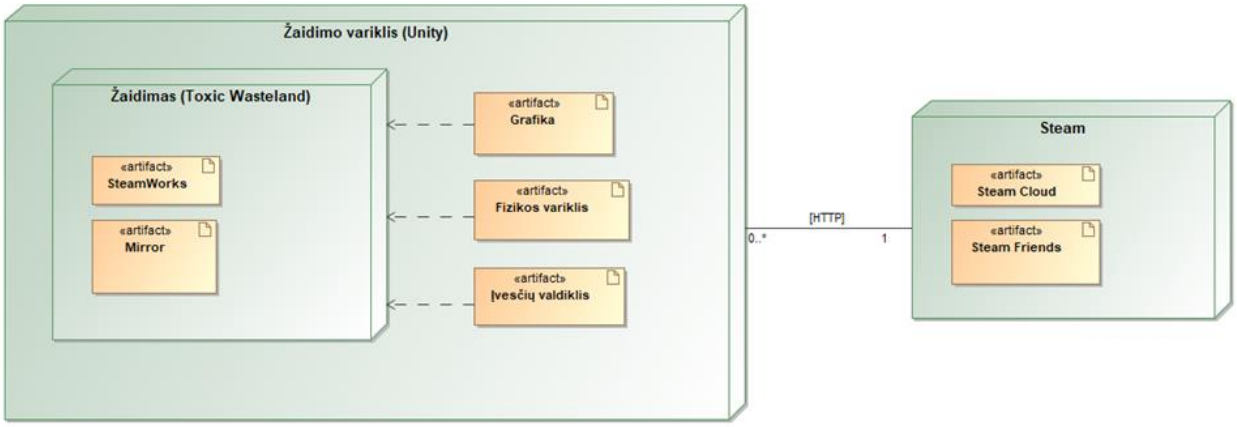


32 pav. Komunikavimo su komandos nariu sekų diagrama



33 pav. Komandos nario pakėlimo sekų diagrama

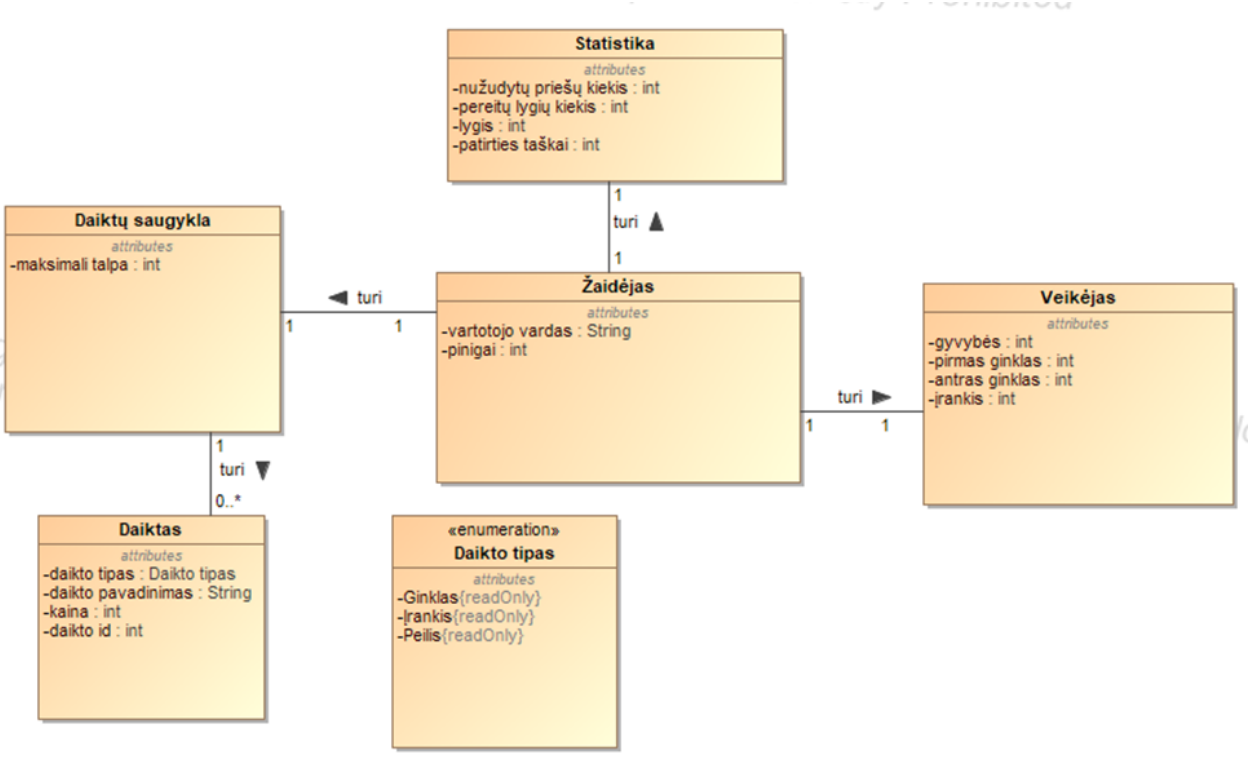
2.9. Išdėstymo vaizdas



34 pav. Išdėstymo diagrama

Žaidime naudojamos „SteamWorks“ ir „Mirror“ bibliotekos, tam kad žaidimas galėtų veikti ir komunikuoti su „SteamCloud“ ir „Steam Friends“ dalimis esančiomis „Steam“ platformoje. „Steam“ komunikuoja su „Unity“ varikliu, kurio grafikos, fizikos ir įvesčių valdikliai palaiko žaidimą.

2.10. Duomenų diagrama



35 pav. Sistemos duomenų modelis

2.11. Naudoti papildiniai

Sistemos kūrimui buvo naudoti šie papildiniai:

- „*Aurora Engine – Mirror Network*“. Šaudymo ir veikėjo kontrolių papildinys su modeliais ir animacijomis.
- „*Procedural Level Generator*“. Atsitiktinių lygių ir kambarių kūrimo įrankis.
- „*Underground Facility Pack*“. Kambarių lygiams modelių bei tekstūrų rinkinys.
- „*Volumetric Blood Fluids*“. Kraujo efektų paketas.
- „*Fantastic Creature #1*“. Vieno iš zombių modelis ir animacijos.
- „*True Horror – Crawler*“. Vieno iš zombių modelis ir animacijos.
- „*Mirror*“. Tinklo sąsajos biblioteka.
- „*FizzySteamworks*“. „*Mirror*“ sujungimo su „*SteamWorks*“ biblioteka.
- Likusieji zombių modeliai paimti iš „*Mixamo*“.
- Likusios animacijos paimtos iš „*Mixamo*“.
- Žaidimo garsai panaudoti iš „*FreeSound*“.
- Žaidimo muzika paimta iš „*Pixabay*“.

3. Tyrimo dalis

Tyrimo dalyje tiriama suprojektuoto žaidimo atsitiktinai generuojamo turinio sistemos kokybė. Atlikus tyrimą ir radus būdų pagerinti atsitiktinai generuojamo turinio sistemą, pateikiami ir realizuojami siūlomi patobulinimai.

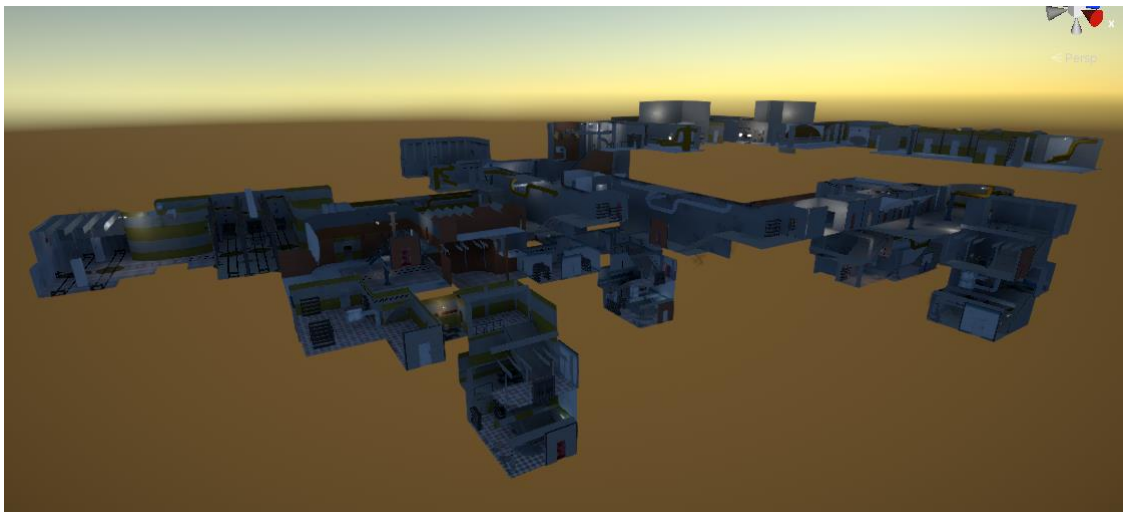
3.1. Tiriamos sistemos aprašymas

Tyrimui naudojama sistema yra žaidimas „Toxic Wasteland“ sukurtas su „Unity“ žaidimų varikliu komandoje kartu su kolega Domu Raulinaičiu magistro studijų metu. Žaidimą galima žaisti tinkle nuo vieno iki keturių žaidėjų, todėl atsitiktinai sugeneruotas turinys turi sutapti visiems žaidėjams esantiems serveryje.

Žaidimo tikslas yra įvykdyti vieną iš dviejų misijų skirtinguose žaidimo režimuose. Žaidimo režime „Išvalymas“ žaidėjai turi išnaršyti visą žaidimo lygį. Tai padarius, atsiranda sekanti misijos dalis, kuri reikalauja, kad žaidėjai nukautų visus lygyje likusius priešus. Šiame žaidimo režime priešai gali atsirasti tik dar žaidėjams nebuvusiuose kambariuose, o atsiradę priešai tampa agresyvūs tik tada, kai jie pamato žaidėjus. Žaidimo režime „Išgyvenimas“ žaidėjų tikslas yra nukauti tam tikrą kiekį priešų. Priešai gali atsirasti visuose kambariuose, kur tik nėra žaidėjų. Priešai yra agresyvūs ir bando pasiekti žaidėjus vos tik priešams atsiradus lygyje.

Žaidimo progresas vyksta žaidėjams pereinant lygius, taip gaunant pinigų, už kuriuos galima nusipirkti ginklus sekančioms misijos, ir patirtį, kuri atrakina galingesnius žaidimo ginklus parduotuvėje. Mirus žaidimo eigoje žaidėjai praranda veikėjų turimus daiktus ir turi juos pirkti dar kartą.

Žaidimas naudoja atsitiktinį turinio generavimą trims pagrindiniams žaidimo elementams generuoti: žaidimo lygiai, priešai ir žaidėjams naudingi daiktai lygyje. Žaidimo lygių generavimui buvo panaudota nemokama „Procedural Level Generator“ sistema iš „Unity Asset Store“. Ši sistema paima žaidimo kūrėjų sukurtus kambarius ir atsitiktine tvarka juos sudeda į vieną norimo dydžio labirintą [20]. Priešams bei naudingiems daiktams lygyje generuoti buvo panaudoti paprasti atsitiktinės tvarkos pasirinkimo iš masyvų algoritmai.



36 pav. Sugeneruotas žaidimo lygis

3.2. Sistemos kokybės įvertinimo metrikos

Atsitiktinio turinio generavimo sistemų kokybei tirti galima panaudoti šias metrikas [21]:

- Įvairovė: ar įvairus yra sukurtas turinys? Generuojamas turinys turėtų atrodyti unikaliai ir skirtis nuo kito sugeneruoto turinio.
- Greitaveika: kaip sugeneruotas turinys paveikia žaidimo greitaveiką? Generuojamas turinys turėtų būti sugeneruojamas pakankamai greitai, kad netrikdytų žaidimo eigos ir nesukeltų nepatogumų žaidėjui. Taip pat sugeneruotas turinys neturėtų stipriai pakeisti žaidimo greitaveikos.
- Prisitaikymas: ar generuojamas turinys gali prisitaikyti prie žaidimo konteksto, režimų, sudėtingumų? Generuojamo turinio sistemos turėtų atkreipti dėmesį į žaidimo kontekstą ir generuoti turinį pagal tam tikrus žaidimo nustatymus.
- Balansas: ar generuojamas turinys turi galimybę prisitaikyti prie žaidimo eigos ir ją balansuoti? Generuojamo turinio sistemos turėtų atkreipti dėmesį į žaidimo sunkumą ir generuoti turinį pagal tai ar žaidėjams turėtų būti sunkiau, ar lengviau.
- Išplečiamumas: ar generuojamas turinys turi galimybę keisti savo dydį ir ar didesniu mastu generuojamas turinys yra sukuriamas pakankamai greitai? Generuojamo turinio sistemos turėtų atkreipti dėmesį į žaidėjo norus ir generuoti norimo dydžio lygį, bei veikti pakankamai greitai generuojant tiek mažus, tiek didelius lygius.

3.3. Įvairovė

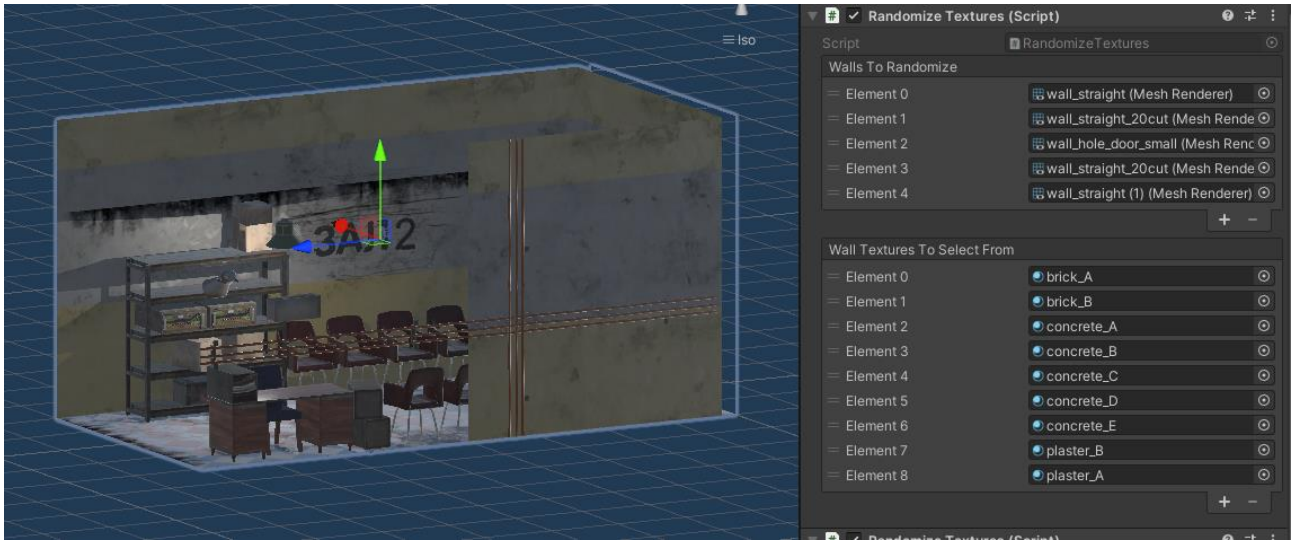
Įvairovės tyrimo metu matuojama ar sukurtas turinys yra įvairus. Atsitiktinio turinio generavimo algoritmai turėtų sukurti pakankamai įvairų turinį, kuris turėtų pakankamai daug skirtingų detalių ir atsitiktinumo, kad sugeneruoti lygiai neatrodytų per daug panašus ir žaidžiant žaidimą daugiau negu vieną kartą būtų daugiau įvairovės.

3.3.1. Įvairovės tyrimas

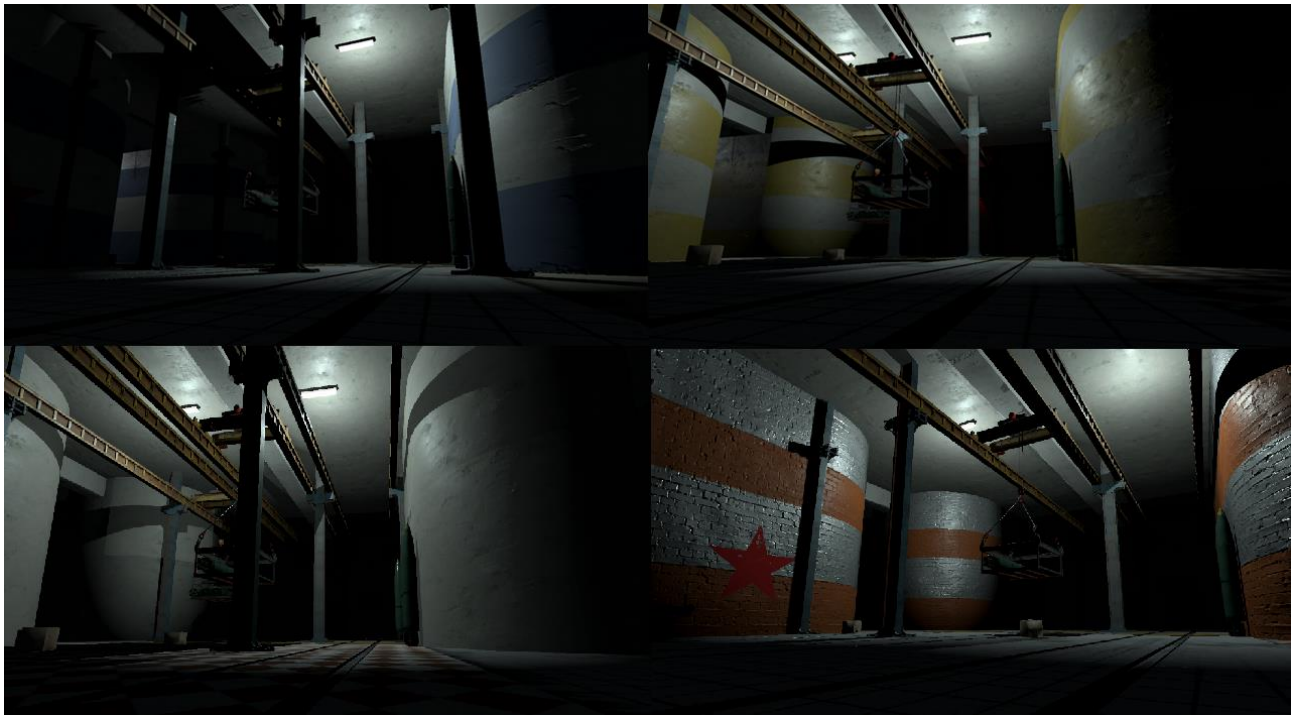
Įvairovei išmatuoti naudojama Shannon diversijos metrika. Ji nurodo skirtingo turinio pasiskirstymą ir kiek bitų reikia norint išsaugoti informaciją apie sugeneruotą turinį. Kuo didesnė diversijos reikšmė, tuo daugiau bitų reikia norint išsaugoti informaciją apie sugeneruotą turinį, ir tuo įvairesnis ir nenuspėjamas yra sugeneruotas turinys [22]. Diversijos formulė: $H(X) = -\sum(p(x)\ log\ p(x))$, kur p(x) yra tam tikro turinio, šiuo atveju, kambario, tikimybė atsirasti lygyje. Žaidimui buvo sukurta 13 skirtingų kambarių, o lygių generavimo algoritmas suteikia vienodą tikimybę kiekvienam kambariui būti sugeneruotam, todėl vieno kambario p(x) yra $\frac{1}{13}$. Taip pat lygis gali būti sugeneruotas iš 10 kambarių, todėl viso lygio p(x) yra $\left(\frac{1}{13}\right)^{10}$. Įstačius p(x) į formulę gaunama $H = -13^{10}\left[\left(\frac{1}{13}\right)^{10}\right] log2\left[\left(\frac{1}{13}\right)^{10}\right] = \sim 37$ bitai, reikalingi išsaugoti 10 kambarių dydžio lygio informaciją. Šį skaičių, o taip pat ir lygio įvairovę, galime padidinti sukurdami sistemos patobulinimą, kuris ne tik atsitiktinai išdėliotų kambarius, tačiau ir padarytų kiekvieną kambarį skirtingu.

3.3.2. Įvairovės patobulinimas

Kadangi lygio įvairovė tiesiogiai priklauso nuo to, iš kiek paruoštų skirtingų kambarių lygių generavimo algoritmas gali pasirinkti generuoti lygį, įvairovę galima padidinti sukuriant patobulinimą, kuris suteiks kiekvieno kambario sienoms skirtingą išvaizdą. Išvaizdą galima atsitiktinai generuoti sukuriant algoritmą, kuris kambario sienoms suteiks atsitiktinai pasirinktą tekstūrą. Norint sumažinti lygio spalvų įvairovę, kiekvienam kambariui turint skirtingas tekstūras, galima į atsitiktinio tekstūros generatorių įkelti kambarių Y koordinatę, taip padarant, kad tekstūros būtų skirtingos ne kiekvienam kambariui, o aukštams. Kiekvieno aukšto kambariai turi savito stiliaus tekstūras ir atrodo skirtingai.

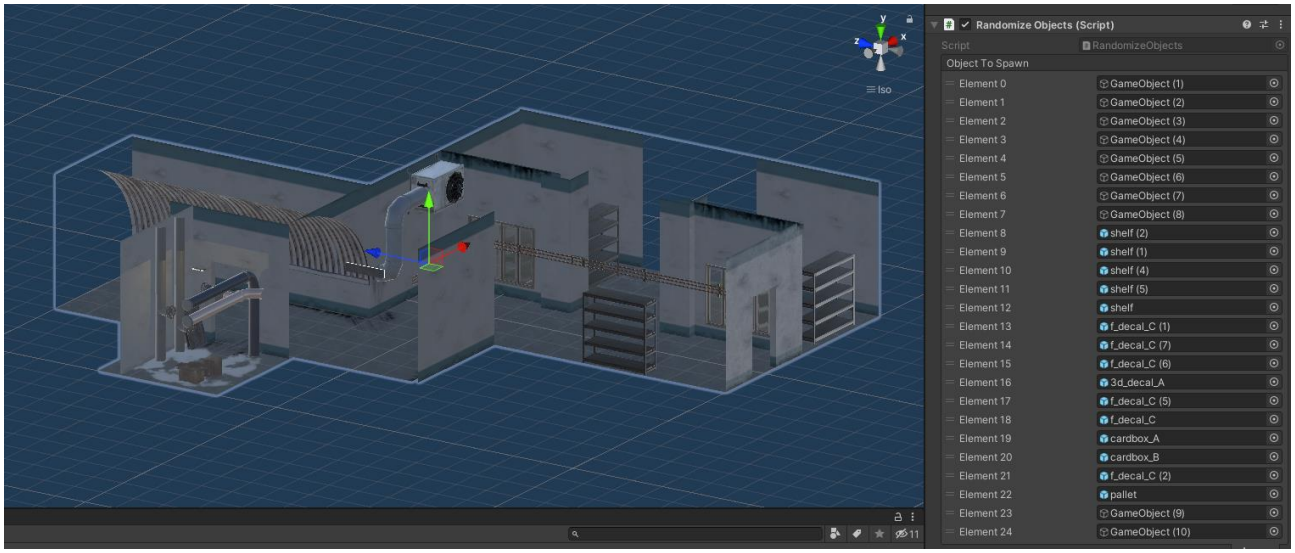


37 pav. Kambario tekstūrų generavimo sistema

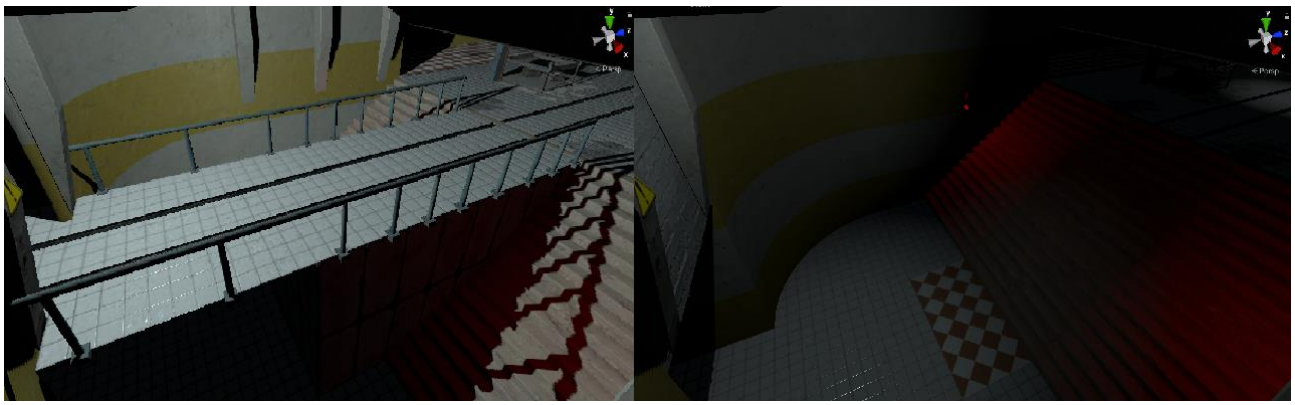


38 pav. Atsitiktinės kambario tekstūros

Kitas būdas suteikti lygiams daugiau įvairovės yra modifikuoti sistemą taip, kad būtų atsitiktinai parenkami ir sugeneruojami objektai kiekviename iš kambarių. Tai kiekvienam, net ir to paties tipo kambariui, suteiks mažą, tačiau ne tik vizualų, bet ir potencialiai žaidimo eigą keičiantį, pakeitimą. Kiekvienas kambarys yra maksimaliai apkraunamas objektais, kurie gali atsirasti kambaryje, įsitikinant, kad žaidėjas vis dar gali per jį pereiti. Tuomet visi šie objektai yra įkeliami į masyvą, kol galiausiai algoritmas atsitiktine tvarka parenka, kuriuos iš šių objektų sugeneruoti kambaryje. Tai gali būti tiek dėžės kambario kampe ar šviesos kurios apšviečia kambarį, tiek vaikštoma kambario dalis ar sienos, kurioms esant sugeneruojamoms, atsiranda papildoma vieta kovai su priešais, kelias iki naudingų daiktų ar pabėgimui nuo priešų. Tai paprasta sistema, kuri suteikia įvairumo tiek žaidimo išvaizdai, tiek žaidimo eigai, tačiau nepakeičia žaidimo ištestavimo galimybių ir nesukelia papildomų problemų.



39 pav. Kambarių objektų generavimo sistema



40 pav. Atsitiktinai sugeneruoti kambario elementai

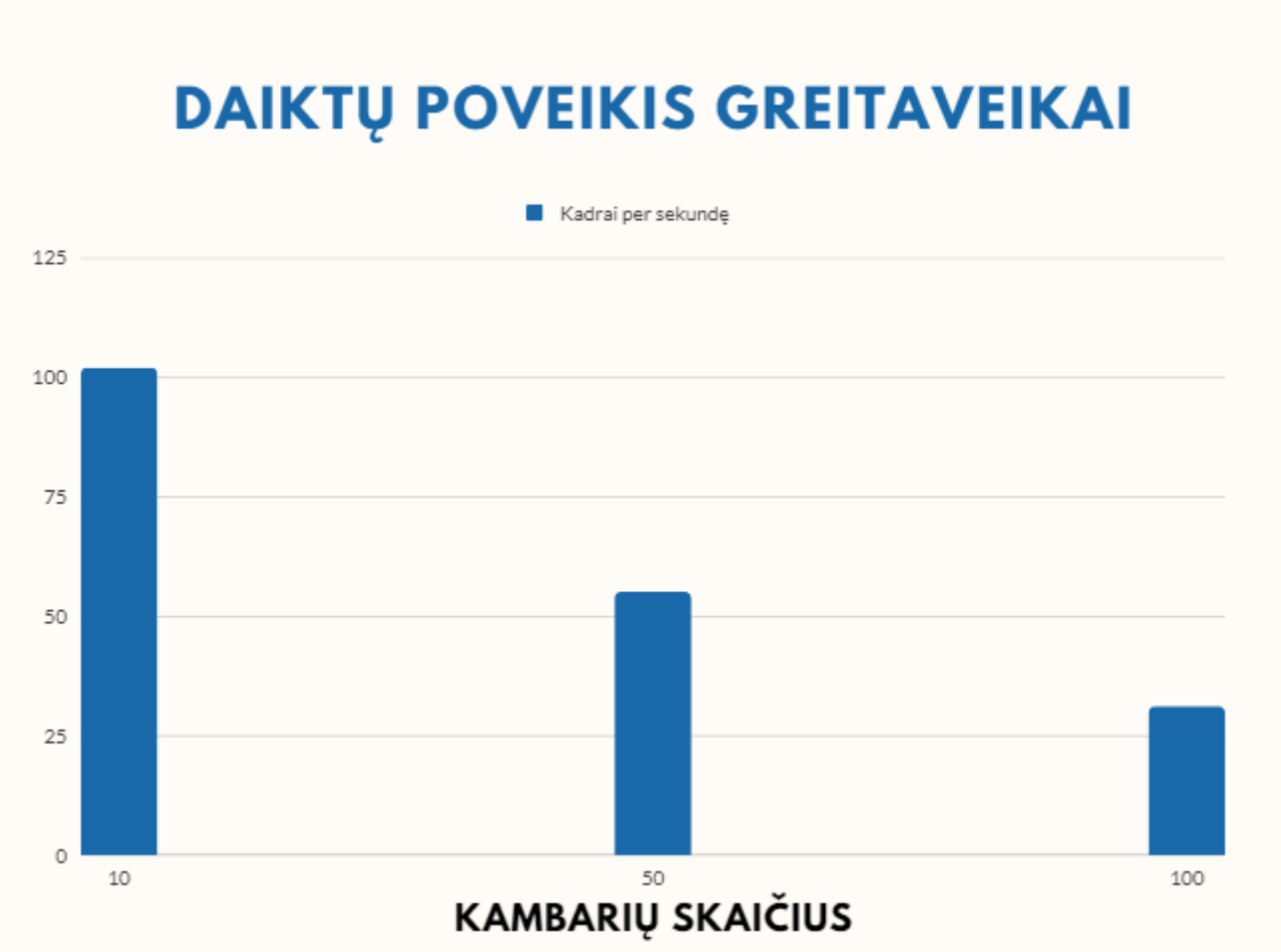
41 pav. Atsitiktinai sugeneruoti kambario elementai

3.4. Greitaveika

Greitaveikos tyrimo metu tikrinama kaip generuojamas turinys paveikia žaidimo greitaveiką. Greitaveikos metrikai naudosime žaidimo kadrus per sekundę. Kuo daugiau kadrų per sekundę kompiuteris gali sugeneruoti žaidimui, tuo yra geriau, nes žaidimas tampa sklandesnis. Tyrimui naudojamas kompiuteris su šiais parametrais: AMD Ryzen 5 3600x 4.25GHz procesorius, GTX 1080 vaizdo plokštė, 16GB RAM. Sklandžiam žaidimui kompiuteris turėtų sugeneruoti bent 60 kadrų per sekundę.

3.4.1. Greitaveikos tyrimas

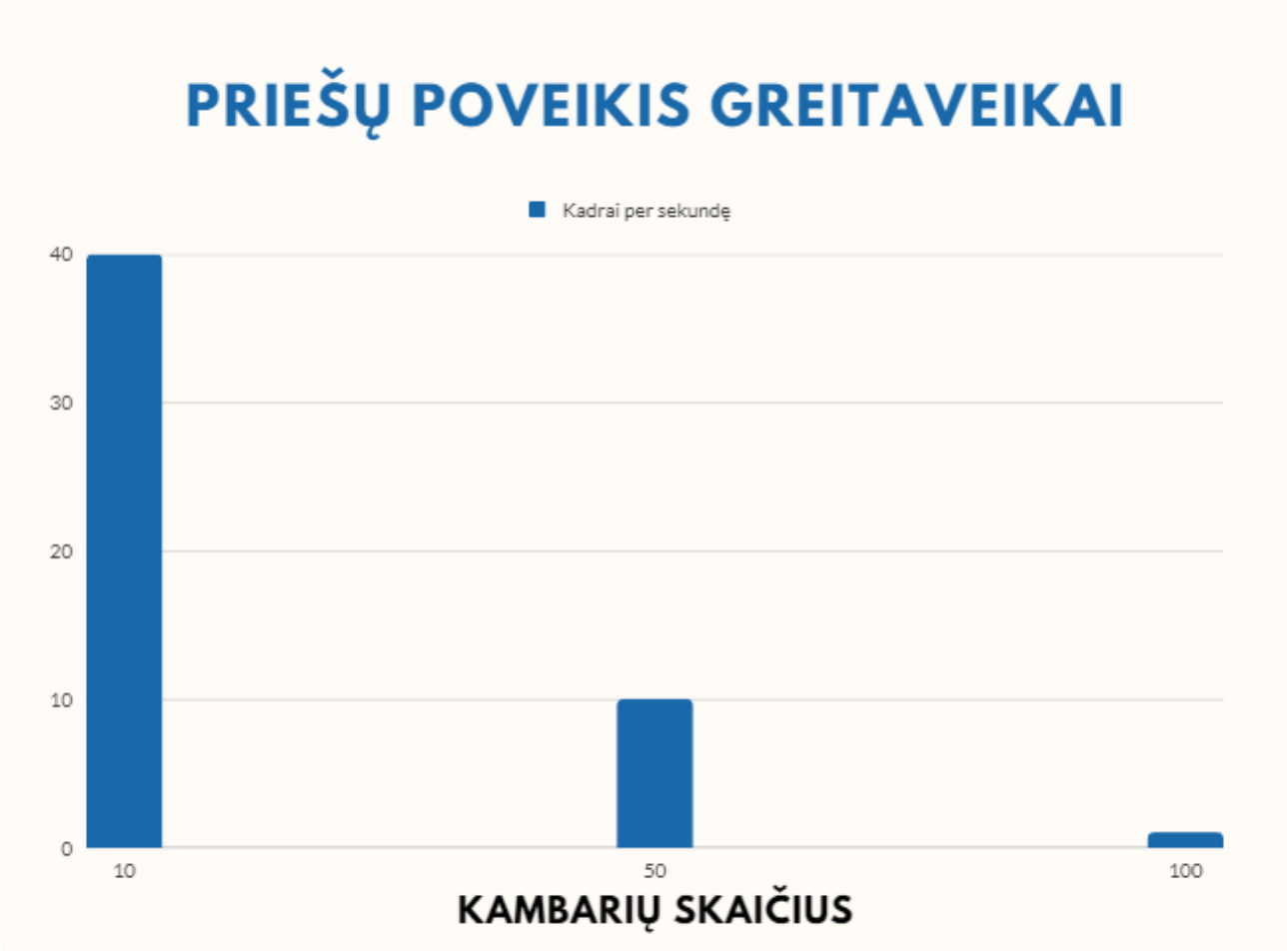
Žaidimo greitaveikai įvertinti naudojamas kadrų per sekundę rodiklis. Kuo kompiuteris gali sugeneruoti daugiau kadrų per sekundę, tuo žaidimas bus sklandesnis. Nemodifikuotas naudingų žaidėjui daiktų generavimo algoritmas generuoja visus daiktus kiekviename kambaryje, vos tik pradėjus krauti žaidimo lygį. Taip žaidimas iš karto sukelia didelę apkrovą kompiuteriui ir kadrų per sekundę skaičius krenta. Kadangi daiktų skaičius lygyje tiesiogiai priklauso nuo lygio dydžio, arba, tiksliau, nuo kambarių skaičiaus, matuosime sugeneruotų daiktų įtaką žaidimo greitaveikai ties 10, 50 ir 100 kambarių. Norint tiksliau ištirti generuojamų daiktų poveikį žaidimo greitaveikai, priešų generavimas tyrimo metu bus išjungtas. Greitaveika nuo žaidime sugeneruotų kambarių skaičiaus nepriklauso dėl žaidime įgyvendintų optimizacijos algoritmų.



42 pav. Kadrų per sekundę skaičiaus priklausomybė nuo kambariuose esančių sugeneruotų daiktų

Rezultatai rodo, kad kuo daugiau kambarių yra su sugeneruotais daiktais, tuo mažesnis kadrų per sekundę skaičius. Žaidžiant vidutinio dydžio lygį, arba lygį su 50 kambarių, kadrai per sekundę nukrenta žemiau 60, kas gali sukelti nepatogumą žaidėjui, nes žaidimas tampa nepakankamai sklandus. Daiktų generavimo algoritmui būtinas patobulinimas, kad žaidimas vėl taptų sklandus.

Sugeneruotas priešų skaičius taip pat tiesiogiai priklauso nuo žaidime esančių kambarių, todėl sugeneruotų priešų daromą poveikį žaidimo greitaveikai taip pat tirsime ties 10, 50 ir 100 kambarių. Norint tiksliau išmatuoti sugeneruotų priešų daromą poveikį žaidimo greitaveikai, daiktų generavimo algoritmas tyrimo metu bus išjungtas. Greitaveika nuo žaidime esančių kambarių skaičiaus nepriklauso dėl įgyvendintų optimizacijos algoritmų.

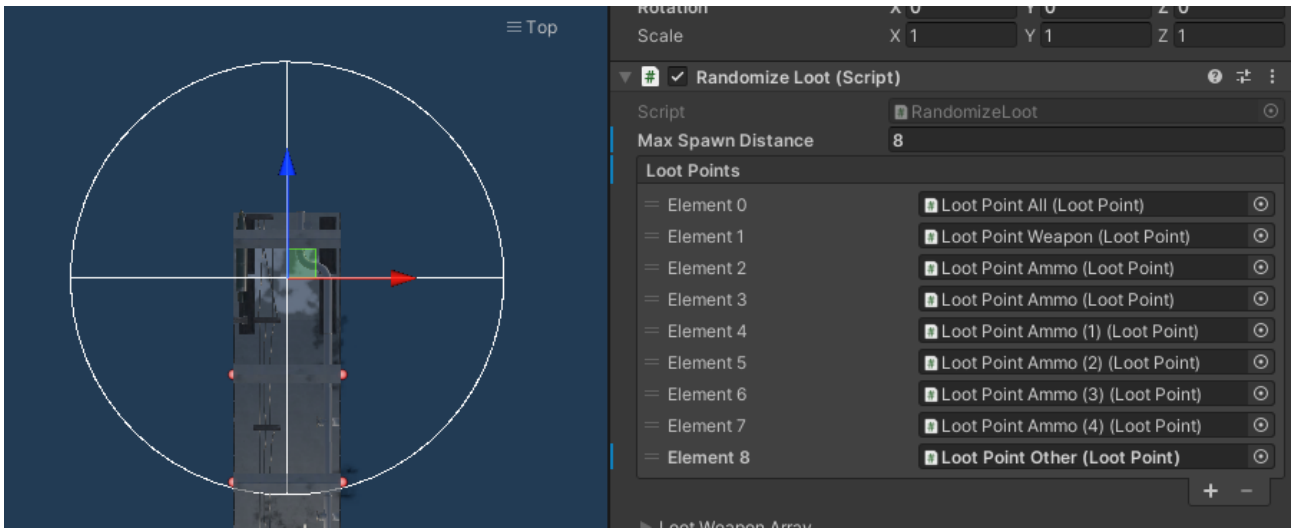


43 pav. Greitaveikos priklausomybė nuo priešų sugeneravimo kambariuose

Rezultatai rodo, kad priešų generavimas turi reikšmingos įtakos žaidimo greitaveikai: kuo didesnio dydžio lygiai, tuo daugiau priešų yra sugeneruojama ir tuo mažiau kadrų per sekundę sugeba sugeneruoti kompiuteris. Ties vidutinio dydžio lygiu priešų sugeneruojama tiek daug, kad žaidimas tampa nebepažaidžiamas, o didelio dydžio lygis tiesiogiai sulaužo žaidimo eigą. Priešų generavimo algoritmui yra būtinas greitaveikos patobulinimas.

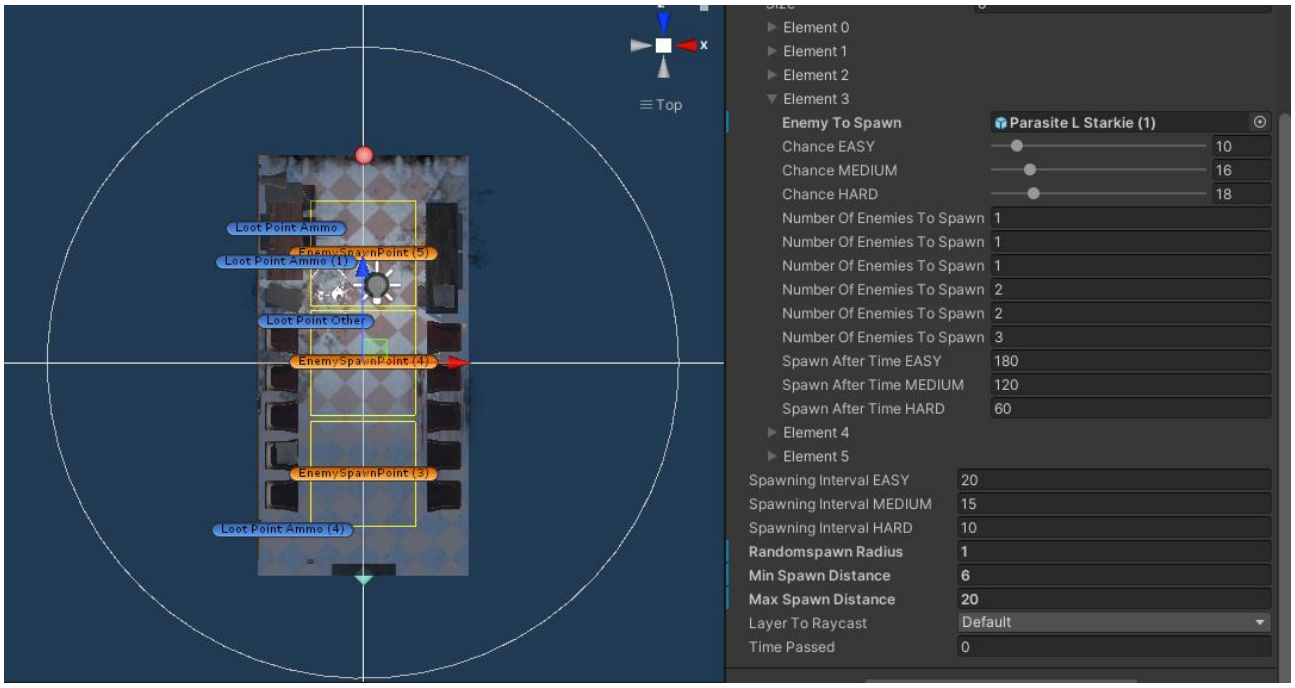
3.4.2. Greitaveikos patobulinimas

Žaidimo greitaveikai pagerinti buvo modifikuotas naudingų žaidėjui daiktų generavimo algoritmas, kad daiktai būtų generuojami tik žaidėjui prisiartinus pakankamai arti prie daiktų atsiradimo vietos. Atstumas, nuo kurio pradedami generuoti daiktai yra nusakomas kintamuoju „Max Spawn Distance“ ir gali būti pakeistas priklausomai nuo kambario. Taip sumažinama žaidimo apkrova kompiuteriui, nes daiktai yra sugeneruojami ne visi iš karto, o dalimis, žaidėjams keliaujant per lygį. Vieno daikto sugeneravimas vidutiniškai užtrunka tarp 1 ir 6 ms. todėl žaidimo eiga generuojant daiktus nėra trikdoma.



44 pav. Daiktų generavimo sistemos patobulinimas

Žaidimo greitaveikai pagerinti taip pat buvo modifikuotas priešų generavimo algoritmas. Pirmasis iš patobulinimų buvo priešų generavimas tik žaidėjui esant pakankamai arti priešų atsiradimo vietos. Atstumas, nuo kurio pradedami generuoti priešai yra „Max Spawn Distance“. Taip pat, norint išvengti, kad priešai atsirastų tiesiai prieš žaidėją, sukurtas ir atstumas „Min Spawn Distance“, kuriame esant, priešai negali atsirasti. Žaidėjams einant per lygį nenugalint priešų, algoritmas vis dar gali sugeneruoti didžiulį kiekį priešų, kas stipriai galėtų paveikti žaidimo greitaveiką, todėl buvo įgyvendintas ir antras algoritmo patobulinimas. Šis patobulinimas skaičiuoja, kiek lygyje iš viso yra priešų ir negeneruoja naujų priešų, jeigu priešų skaičius yra pakankamai didelis. Taip palaikomas priešų kiekis, o kadrų per sekundę rodiklis tampa stabilus.



45 pav. Priešų generavimo algoritmo patobulinimas

3.5. Prisitaikymas

Prisitaikymo tyrimo metu tikrinama kaip gerai generuojamas turinys gali prisitaikyti prie žaidėjų norų ir konteksto. Žaidėjai gali norėti įvairių žaidimo režimų, sudėtingumų ar lygių dydžių. Geras atsitiktinio turinio generavimo algoritmas turėtų ne tik duoti žaidėjui pasirinkimą, kaip turinys turėtų būti sugeneruotas, tačiau ir keisti generuojamą turinį pagal šiuos pasirinkimus, kad žaidimo eiga būtų labiau personalizuota [23].

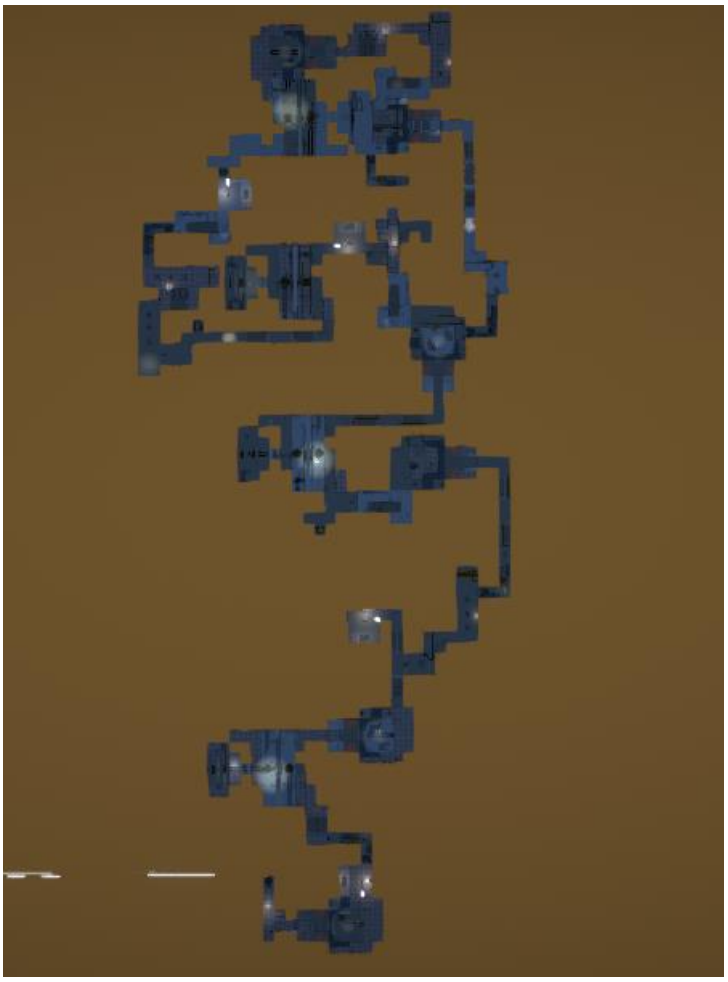
3.5.1. Prisitaikymo tyrimas

Prisitaikymui apskaičiuoti galime pasinaudoti personalizavimo taškų formule. Tai formulė, kuri apskaičiuoja atsitiktinių turinio generavimo algoritmų galimybę prisitaikyti prie žaidėjų norų ir interesų. Formulė: $PS = PM/TP$, kur PS yra personalizavimo taškų kiekis, PM yra kiek skirtingų nustatymų suteikia atsitiktinio turinio generavimo algoritmas, o TP yra visų galimų žaidėjo norų skaičius. Kuo sistemos PS yra didesnis, tuo yra geriau, nes sistema gali labiau prisitaikyti prie žaidėjų norų. Galime paimti TP kaip statinį skaičių, sakydami, kad žaidėjas gali norėti keisti 3 nustatymus, kurių kiekvienas turėtų po 3 pasirinkimus. Tokiu atveju $TP = 3 * 3 * 3 = 27$ skirtingi pasirinkimai, kurių gali norėti žaidėjas. Tuo tarpu, nemodifikuoti projekte įgyvendinti atsitiktinio turinio generavimo algoritmai sugeba generuoti tik vieno tipo atsitiktinius lygius, tik vieno sudėtingumo atsitiktinius priešus ir tik vieno sudėtingumo atsitiktinius žaidėjams naudingus daiktus. Taigi $PM = 1 * 1 * 1 = 1$, rodantis, kad atsitiktinio generavimo sistemos turi tik vieną pasirinkimą. Taigi mūsų nemodifikuotos sistemos $PS = \frac{1}{27}$.

3.5.2. Prisitaikymo patobulinimas

Šiai problemai spręsti galima modifikuoti esamus atsitiktinio turinio generavimo algoritmus, kad jie keistų savo generuojamą turinį pagal žaidėjo pasirinktus nustatymus. Šiuo metu skirtingi žaidimo režimai keičia tik žaidimo priešų elgseną. Tai galime pakeisti pridėdami funkciją, kad lygių generavimo algoritmas atkreiptų dėmesį į pasirinktą žaidimo režimą ir generuotų skirtingus lygius priklausomai nuo pasirinkto žaidimo režimo. Žaidimo režimui „Išvalymas“ realizuotas pakeitimas, kad lygiai būtų generuojami kaip vienas ilgas tunelis, turintis mažai išsišakojimų, o žaidimo režimui „Išgyvenimas“ realizuotas pakeitimas, kad lygiai būtų generuojami kaip labirintas su daug išsišakojimų ir galimų kelių, kuriuose galima pasiklysti, nes šiam žaidimo režimui nebūtina pereiti viso sugeneruoto lygio.

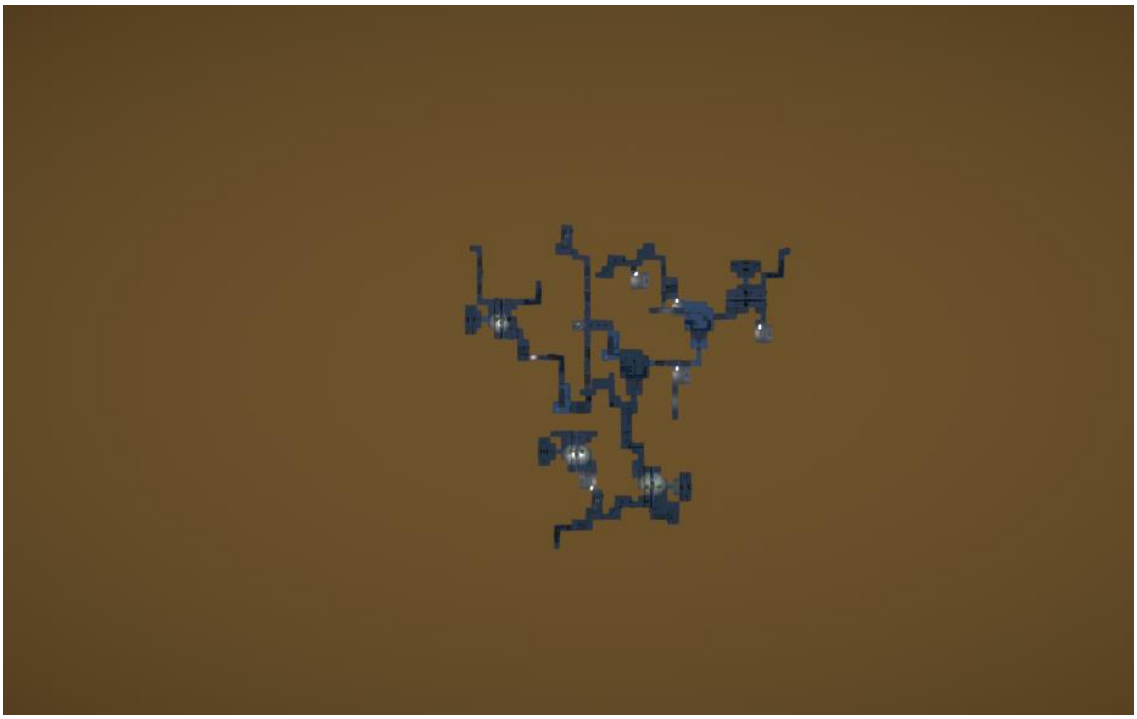
46 pav. Žaidimo režimui „Išgyvenimas“ sugeneruotas labirintas



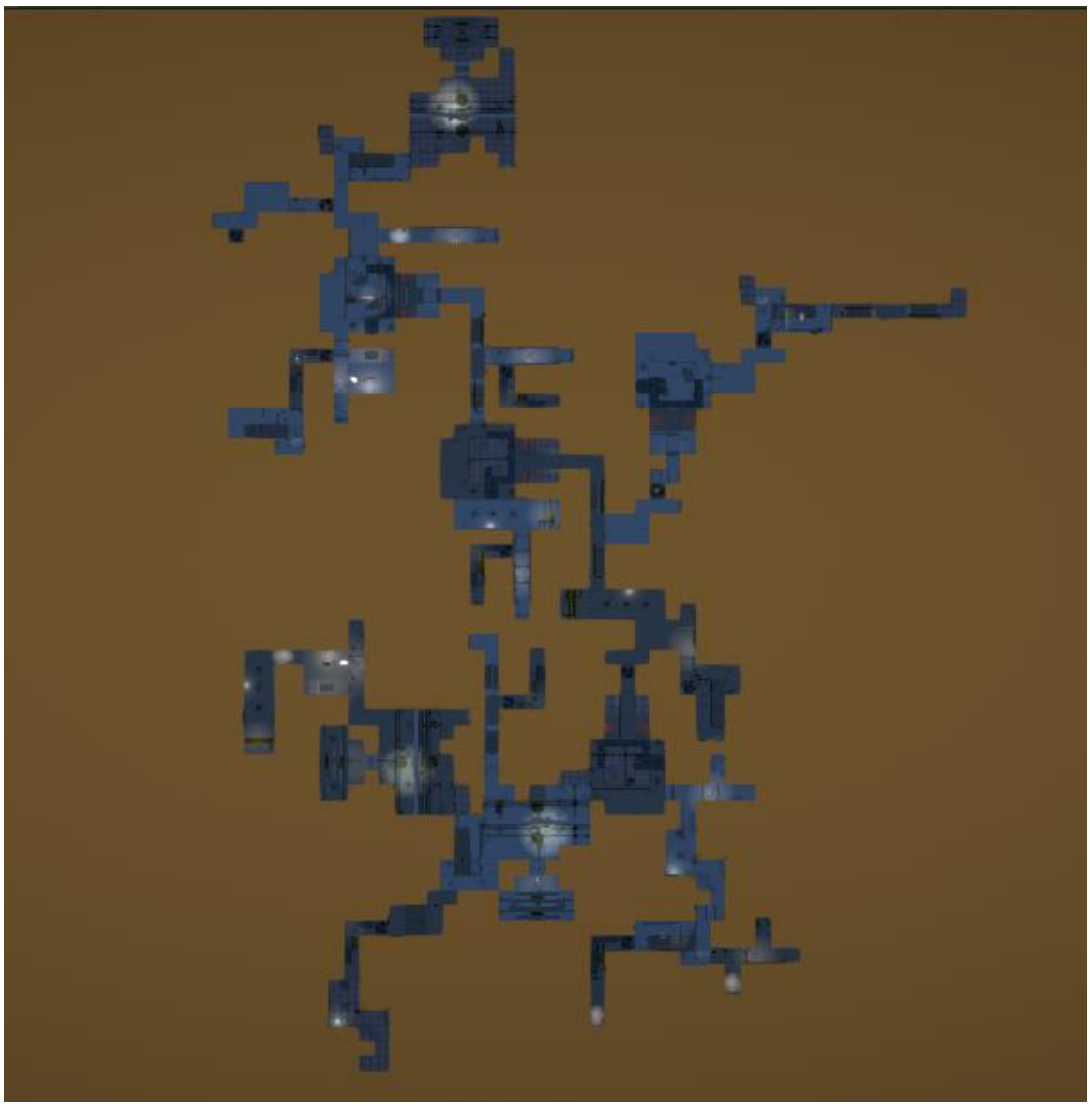
47 pav. Žaidimo režimui „Išvalymas“ sugeneruotas ilgo tunelio lygis

Žaidimo lygio dydžio pasirinkimui galime modifikuoti algoritmą, kad šis atsižvelgtų į žaidėjo pasirinktą lygio dydį. Modifikavus lygio generavimo algoritmą, sistema reaguoja į žaidėjo pasirinktą lygio dydžio nustatymą. Pagal tai lygiai gali būti sugeneruoti iš daugiau ar mažiau kambarių, taip leidžiant žaidėjui pasirinkti žaidimo ilgį. Maži lygiai susideda iš 10 kambarių, vidutiniai iš 50, o dideli iš 100.

48 pav. Mažo dydžio sugeneruotas lygis

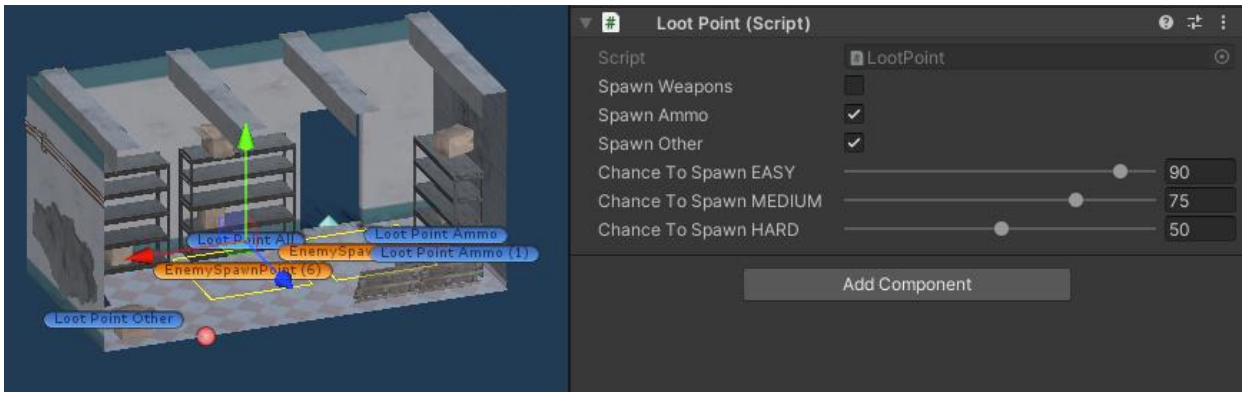


49 pav. Vidutinio dydžio sugeneruotas lygis



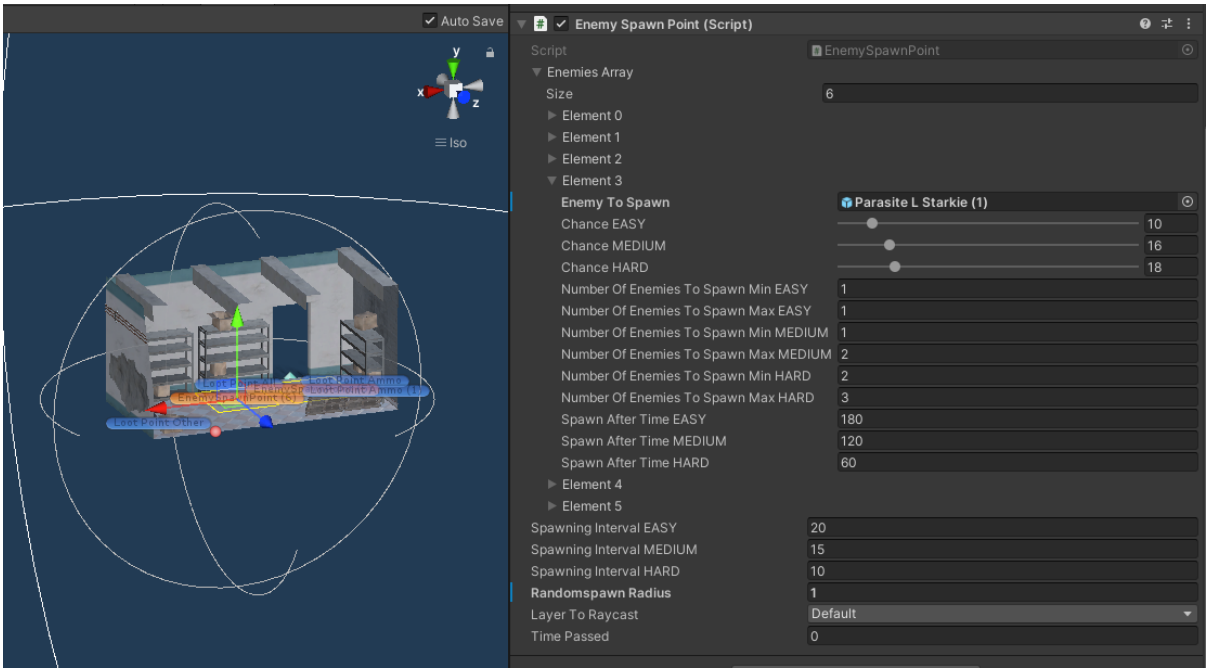
50 pav. Didelio dydžio sugeneruotas lygis

Žaidimo sunkumo pasirinkimui modifikuotas naudingų žaidėjui daiktų generavimo algoritmas, kuris atsižvelgia į pasirinktą žaidimo sunkumą ir turėtų didesnį ar mažesnį šansą generuoti daiktus, kurie padėtų žaidėjui pereiti lygį. Kuo sunkesnis lygis, tuo mažesnis šansas, kad algoritmas sugeneruos žaidėjui reikalingą daiktą.



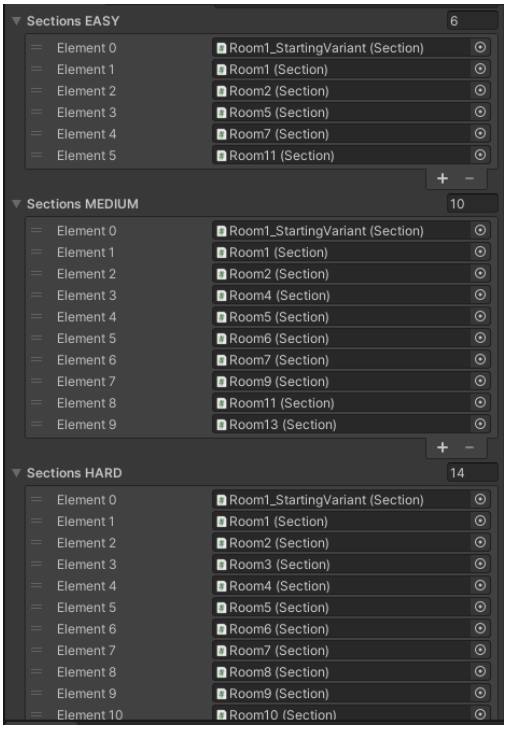
51 pav. Daiktų generavimas pagal pasirinktą sunkumą

Panašią sistemą galime įgyvendinti ir priešų generavimui, suteikiant sudėtingesniems priešams mažesnę tikimybę būti sugeneruotiems lengvesnio žaidimo režimo metu, taip pat sumažinant jų skaičių ir generavimo dažnį.



52 pav. Priešų generavimas priklausomai nuo žaidimo sudėtingumo nustatymo

Galiausiai, galima modifikuoti lygių generavimo sistemą taip, kad ji atsižvelgtų į pasirinktą žaidimo sunkumą ir generuotų lygius iš tam tikrų kambarių priklausomai nuo sunkumo. Lengvas sunkumas reiškia, kad lygis yra sukuriamas iš tiesių, mažų kambarių, o sudėtingiausias sunkumas generuotų lygius iš kompleksiškų, didelių kambarių.



53 pav. Lygių generavimo kambariai pagal sunkumą

3.6. Balansas

Atsitiktinai generuojamo turinio balansas yra svarbi metrika, jeigu atsitiktiniai algoritmai daro įtaką žaidimo eigai. Projekto atveju, priešai ir žaidimui įveikti reikalingi resursai yra atsitiktinai generuojami, todėl balansas yra viena iš metrikų, kurią reikia ištirti ir, esant problemoms, patobulinti. Geras atsitiktinio turinio generavimo balansas reikš, kad žaidimą galima reguliariai įveikti, o žaidėjo pasisekimą žaidime lems jo įgūdžiai, o ne atsitiktinai žaidimo sugeneruotas turinys žaidėjo naudai ar nenaudai [24].

3.6.1. Balanso tyrimas

Žaidimo balansui matuoti yra naudojama žaidimo laimėjimų dažnio formulė kuri yra $laimėjimų\ dažnis = \frac{žaistų\ žaidimų\ skaičius}{sužaistų\ žaidimų\ skaičius}$, tačiau ši metrika gali būti labai subjektyvi neturint didelio žaidėjų ar testuotojų skaičiaus. Todėl vietoj to, galima ištirti ar žaidimo lygis sugeneruoja pakankamai resursų reikalingų lygiui įveikti. Tam panaudojama formulę, kuri apskaičiuoja koeficientą tarp reikalingų resursų kiekio ir vidutiniškai sugeneruojamų resursų kiekio. Jeigu koeficientas mažesnis už 1, reiškia, kad lygis nesugeneruoja pakankamai resursų lygiui įveikti, 1 reiškia, kad generavimas yra subalansuotas ir generuojamų priešų skaičius ir sugeneruojamų resursų skaičius norint jiems įveikti yra lygus. Koeficientui būnant daugiau už vieną, reiškia, kad sugeneruojama daugiau resursų negu reikia, todėl žaidėjas gali padaryti daugiau klaidų ir žaidimas tampa lengvesnis. Formulė: $balanso\ koeficientas = \frac{(galima\ padaryti\ žala)}{(reikalinga\ padaryti\ žala)}$. Kadangi žaidimas turi tris sudėtingumo lygius, koeficientas turėtų būti virš 1 lengvam lygiui, kuo arčiau 1 vidutiniam lygiui ir mažiau už 1 sudėtingam lygiui.

Norint sužinoti reikalingą padaryti žalą, kad lygis būtų pereitas, apskaičiuojama vidutinė galima priešų gyvybių reikšmė. Ji randama taip: $gyvybės =$ (vidutinis priešų atsiradimo taškų skaičius) $*$ (vidutinė gyvybių reikšmė viename taške). Vidutinis priešų atsiradimo taškų skaičius viename kambaryje randamas taip: $vidutinis\ taškų\ skaičius = \frac{(visų\ atsiradimo\ taškų\ skaičius\ kambariuose)}{(kambarių\ skaičius)} *$ $(sugeneruotų\ kambarių\ skaičius)$. Vidutinė gyvybių reikšmė viename taške: $vidutinės\ gyvybės\ taške = (galimo\ sugeneruoti\ priešo\ gyvybės) *$ $(priešo\ sugeneravimo\ tikimybė)$.

Norint sužinoti kiek vidutiniškai žalos galima padaryti su lygio sugeneruotais resursais naudosime formulę: $galima\ padaryti\ žala = \big((pradinis\ amunicijos\ kiekis) +$ $(vidutinis\ sugeneruotas\ amunicijos\ kiekis)\big) * (amunicijos\ daroma\ žala)$. Norint rasti vidutinį sugeneruotą amunicijos kiekį naudosime $sugeneruota\ amunicija =$ $(vidutinis\ amunicijos\ atsiradimo\ taškų\ skaičius) *$ $(amunicijos\ sugeneravimo\ tikimybė) * (taško\ generavimo\ tikimybė) *$ $(amunicijos\ kiekis\ viename\ taške)$. Galiausiai, norint apskaičiuoti vidutinį amunicijos atsiradimo taškų skaičių naudosime: $vidutinis\ amunicijos\ taškų\ skaičius = \frac{(visų\ amunicijos\ taškų\ skaičius)}{(kambarių\ skaičius)} *$ $(sugeneruotų\ kambarių\ skaičius)$.

Skaičiavimui pasirinktas vidutinio dydžio lygis su 50 kambarių. Balansas bus apskaičiuojamas dažniausiai pasitaikantiems, automatiniams ginklams, kurių kulkos daro po 35 žalos. Skirtingų

sunkumo lygių pasirinkimas buvo įgyvendintas tik prisitaikymo tyrimo metu, todėl skaičiuojamas tik vienas balanso koeficientas ištirti situacijai prieš modifikaciją.

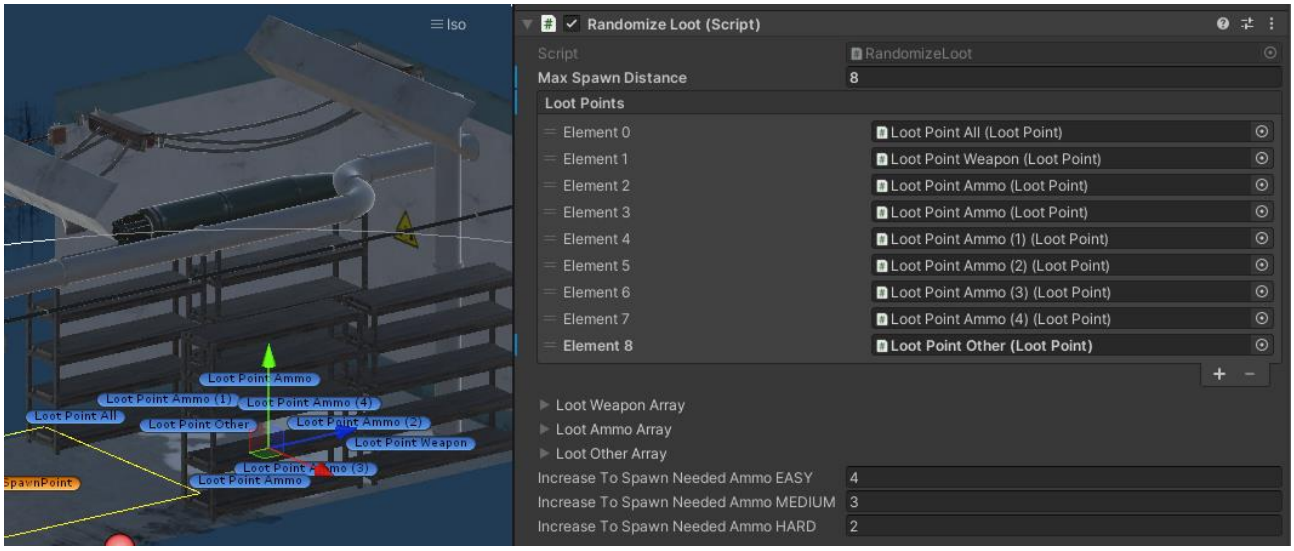
Pirmiausia apskaičiuojamas vidutinis amunicijos atsiradimo taškų skaičius: $\frac{77}{13} * 50 = \sim 296$. Toliau apskaičiuojama vidutiniškai sugeneruota amunicija lygyje: $296 * 0.2 * 0.7 * 30 = 1243.2$. Galima padaryti žala: $(300 + 1243.2) * 35 = 54012$. Kadangi žaidime galima turėti du skirtingus ginklus, šį skaičių galima padvigubinti: $54012 * 2 = 108024$ Vidutinės gyvybės taške: $100 * 0.25 + 300 * 0.2 + 200 * 0.2 + 2000 * 0.03 + 250 * 0.16 + 100 * 0.16 = 241$. Vidutinis taškų skaičius: $\frac{234}{13} * 50 = 900$. Vidutinės gyvybės lygyje: $900 * 241 = 216900$. Galiausiai balanso koeficientas: $\frac{108024}{216900} = \sim 0.50$.

Rezultatai rodo, kad algoritmas sugeneruoja per mažą kiekį resursų susidoroti su visais lygio generuojamais priešais. Generuojamų resursų kiekis turėtų būti apie 2 kartus didesnis, kad amunicijos užtektų nukauti visiems priešams.

3.6.2. Balanso patobulinimas

Patobulinimai, kad žaidimas keistų savo balansą pagal žaidėjo pasirinktą sudėtingumo lygį buvo atlikti prisitaikymo tyrimo metu. Belieka pakeisti reikšmes taip, kad algoritmas sugeneruotų pakankamai naudingų daiktų žaidėjui kovoti su priešais.

Buvo atliktas papildomas balanso gerinimo tobulinimas, daiktų generavimo algoritmą modifikavus taip, kad jis generuotų turinį tinkamesnį tam tikram žaidėjui. Patobulintas algoritmas atkreipia dėmesį į žaidėjo turimus ginklus ir padidina šansą, kad jiems reikalinga amunicija bus sugeneruota. Tai ne tik pagerina žaidimo balansą, tačiau ir padaro žaidimą patogesnį žaidėjui, nes jam reikia praleisti mažiau laiko ieškant daiktų kurie jam yra reikalingi, tačiau nesugriaunama žaidimo atmosfera ir vis dar yra generuojami įvairūs daiktai, sukuriant iliuziją, kad žaidimas vyksta apleistuose tuneliuose.



54 pav. Daiktų generavimo sistemos patobulinimas balansui

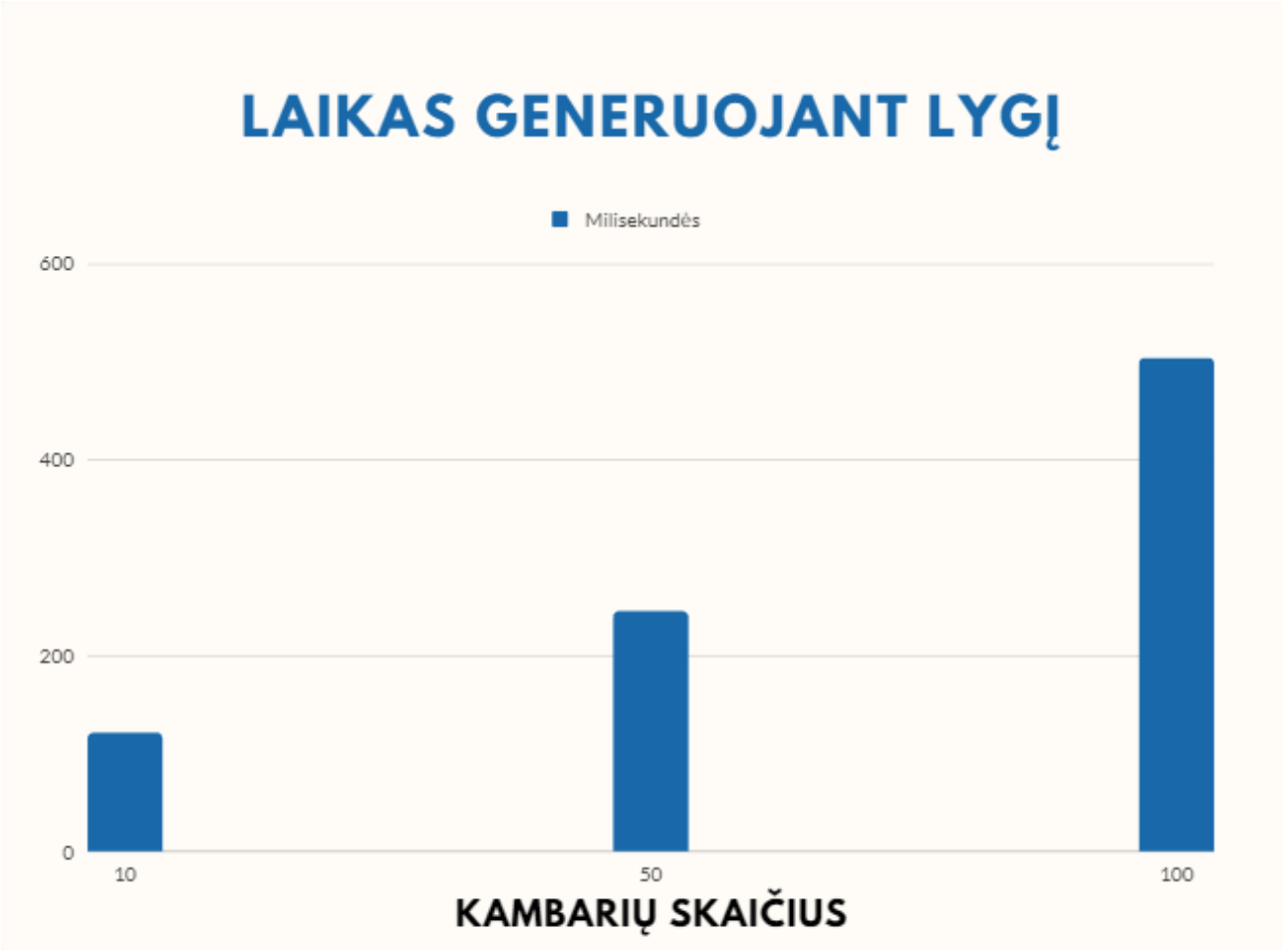
3.7. Išplečiamumas

Išplečiamumo metrika yra naudojama tiek patikrinti, ar algoritmas sugeba generuoti skirtingo dydžio turinį, tiek išmatuoti laiką, sugaištamą sukurti įvairaus dydžio turinį. Kuo mažiau laiko užtrunka turinio generavimas, tuo yra geriau, nes žaidėjas turi laukti trumpiau, o turinys generuojamas ne krovimo ekrano metu, bet žaidimo eigoje, gali stabdyti žaidimą ir sukelti nepatogumų žaidėjui.

3.7.1. Išplečiamumo tyrimas

Prisitaikymo tyrimo metu buvo realizuota galimybė lygių sistemai kurti skirtingų dydžių turinį, todėl galima teigti, kad algoritmas išpildo šį reikalavimą. Priešų ir naudingų žaidėjui daiktų generavimas yra tiesiogiai priklausomas nuo lygio dydžio, nes kuo lygis yra didesnis, tuo daugiau daiktų bei priešų reikia sugeneruoti. Greitaveikos tyrimo metu realizuotas patobulinimas, kuris generuoja priešus bei daiktus tik žaidėjui esant pakankamai arti jų atsiradimo vietos, todėl pašalinta problema dėl jų generavimo laiko krovimo metu. Tačiau ar lygio generavimo laikas daro didelę įtaką žaidimo krovimo laikui ir ar reikalingas papildomas patobulinimas? Taip pat, kiek laiko užtrunka daiktų ir priešų generavimas prieš atliekant greitaveikos tyrimo metu atliktą patobulinimą?

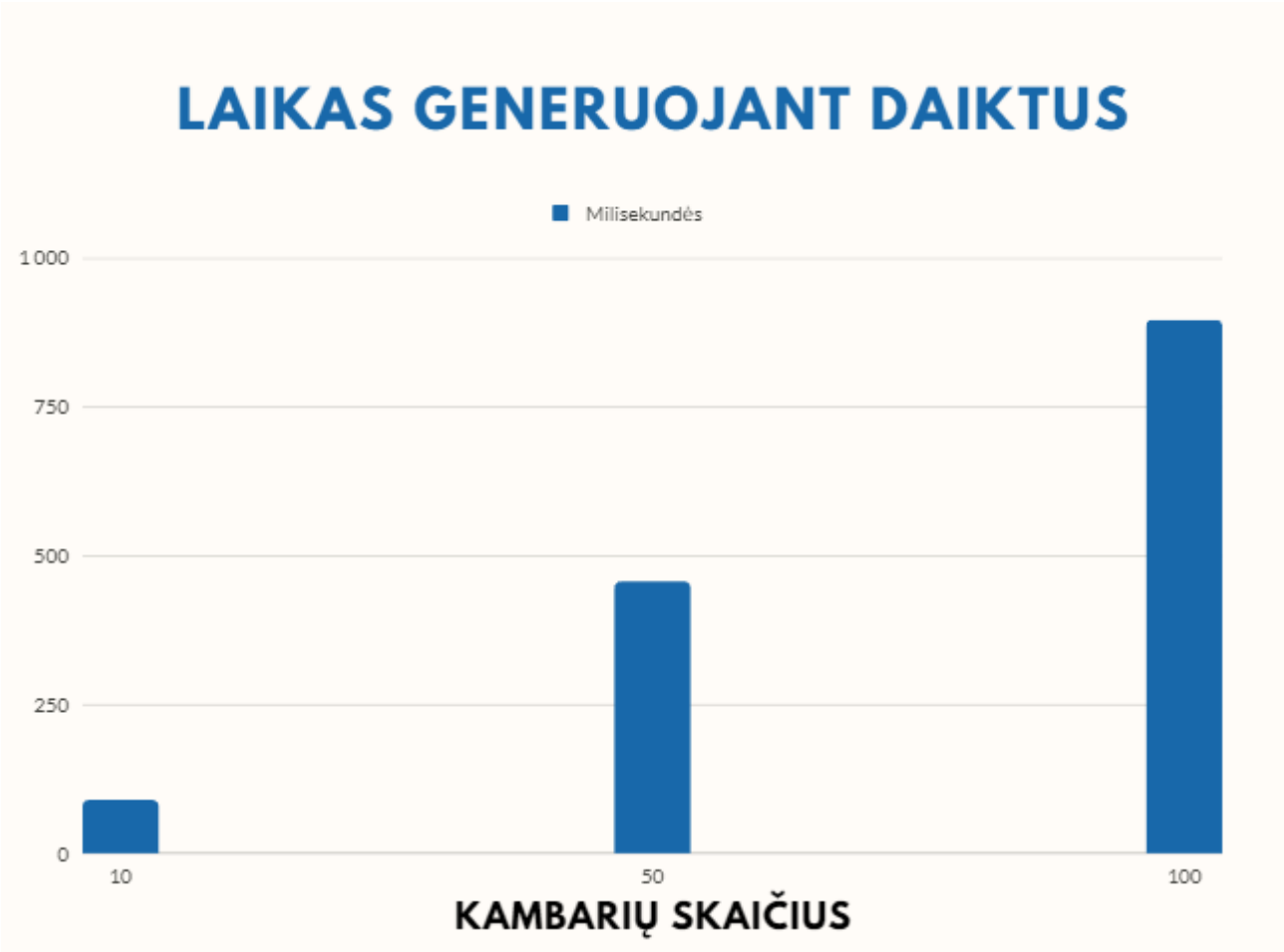
Lygio generavimo laikui apskaičiuoti buvo sukurtas metodas, kuris pradeda skaičiuoti laiką nuo lygio generavimo pradžios iki visų kambarių sudėjimo į lygį pabaigos. Ištestavus visų lygių dydžių generavimo laiką, buvo gauti ir įvertinti rezultatai.



55 pav. Laikas reikalingas sugeneruoti skirtingų dydžių lygius

Lygių generavimo ilgis yra pakankamai trumpas, todėl žaidėjui nesukelia trukdžių generavimo laiku. Laikas generuojant lygį svyruoja nuo 150 ms mažam lygiui iki 500 ms dideliam lygiui. Taip pat, 100 kambarių lygiai yra pakankamai dideli, todėl papildomų patobulinimų lygių generavimo sistemai nereikia.

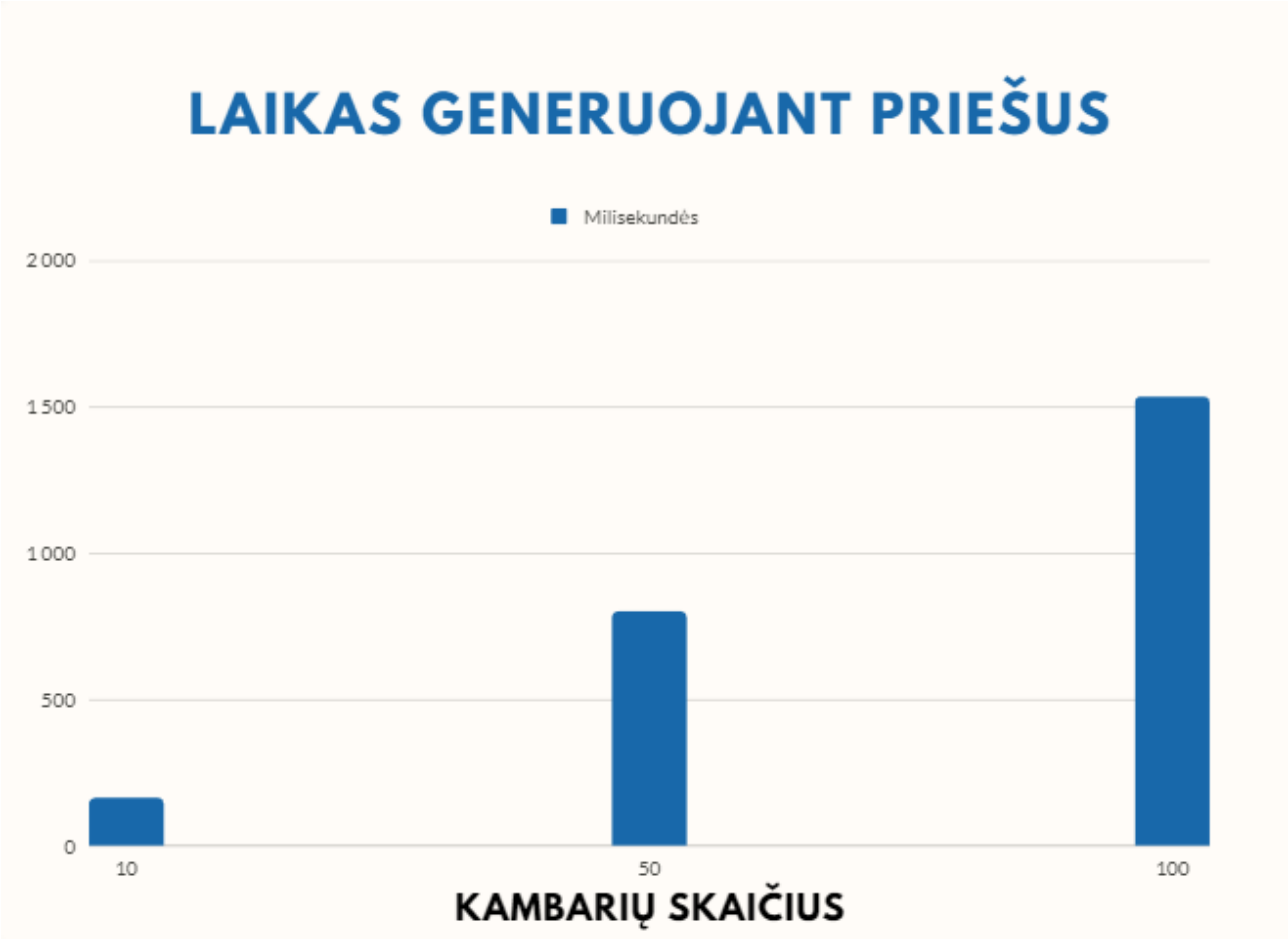
Naudingų daiktų žaidėjui generavimo laiko apskaičiavimui buvo sukurtas metodas, kuris skaičiuoja laiką nuo generavimo pradžios iki paskutinio daikto sugeneravimo pabaigos. Kadangi kiekvienas kambarys turi būti užpildytas daiktais, generavimo laikas tiesiogiai priklauso nuo lygio dydžio, arba, tiksliau, nuo kambarių skaičiaus. Sugeneruoti vieną daiktą užtrunka nuo 1 iki 6 ms, tuo tarpu lygis gali turėti nuo 100 iki 1000 daiktų.



56 pav. Daiktų generavimo laikas

Iš rezultatų matome, kad laikas generuojant daiktus yra truputį ilgesnis už laiką reikalingą lygiui sugeneruoti, todėl tai padidina viso lygio krovimo laiką. Tam padėti turėtų greitaveikos tyrimo metu įgyvendintas patobulinimas, išdalinant daiktų generavimą į atskirus kambarius, o generavimą aktyvuojant tik žaidėjui priėjus pakankamai arti.

Priešų generavimo laiko apskaičiavimui buvo sukurtas metodas, kuris skaičiuoja laiką nuo priešų generavimo pradžios iki paskutinio priešo generavimo pabaigos. Kadangi kiekvienas kambarys turi būti užpildytas atsitiktiniu priešų skaičiumi, generavimo laikas tiesiogiai priklauso nuo lygio dydžio, arba, tiksliau, kambarių skaičiaus. Sugeneruoti vienam priešui vidutiniškai reikia apie 1 ms.



57 pav. Priešų generavimo laikas

Laikas generuojant priešus yra gerokai ilgesnis už lygių generavimą ar naudingų žaidėjui daiktų generavimą, todėl patobulinimas priešų generavimo algoritmui yra reikalingiausias norint išvengti ilgo viso lygio krovimo laiko. Tam padėti turėtų greitaveikos tyrimo metu įgyvendintas patobulinimas išskaidant priešų generavimą į atskiras kambarių dalis ir generuojant priešus tik žaidėjui esant pakankamai arti priešų atsiradimo vietos.

3.7.2. Išplečiamumo patobulinimas

Patobulinimas, kad algoritmas galėtų generuoti skirtingų dydžių lygius buvo įgyvendintas prisitaikymo tyrimo metu. Patobulinimui buvo modifikuota lygių generavimo sistema, kad ši atsižvelgtų į žaidėjo pasirinktą lygio dydžio nustatymą, ir generuotų lygius iš 10, 50 ar 100 kambarių, taip sukuriant mažesnį ar didesnį lygį.

Patobulinimas skirtas naudingų žaidėjui daiktų generavimo laikui sutrumpinti buvo įgyvendintas greitaveikos tyrimo metu. Patobulinimui buvo pridėtas funkcionalumas, kad daiktai kambaryje būtų sugeneruojami tik žaidėjui priartėjus pakankamai arti prie daiktų atsiradimo vietos, taip išskaidant ilgą visų daiktų generavimo laiką į kelių milisekundžių darbą per visą žaidimo eigą.

Patobulinimas skirtas priešų generavimo laikui sutrumpinti buvo įgyvendintas greitaveikos tyrimo metu. Patobulinimui buvo pridėtas funkcionalumas, kad priešai būtų sugeneruojami tik žaidėjui priartėjus pakankamai arti prie priešų atsiradimo vietos, taip išskaidant ilgą priešų generavimo laiką iki kelių milisekundžių darbo per visą žaidimo eigą.

3.8. Tyrimo išvados

Žaidimo įvairovės tyrimo metu pastebėta, kad žaidimo lygių generatorius neturi daug būdų atsitiktinai sukurti lygius su daug įvairovės. Įvairovei pagerinti buvo modifikuota lygių generavimo sistema, pridedant du papildomus atsitiktinių elementų generatorius į kiekvieną žaidimo kambarį. Patobulinus algoritmą, kiekvienas lygis yra sukuriamas pasinaudojant ne tik 13 kambarių, bet ir 9 kambarių tekstūromis ir apie 15 atsitiktinai sugeneruotų daiktų kiekviename kambaryje.

Tiriant žaidimo atsitiktinio turinio generavimo algoritmų poveikį greitaveikai pastebėta, kad užpildant lygius daiktais ir priešais yra stipriai paveikiama žaidimo greitaveika. Problemai išspręsti buvo modifikuoti priešų ir daiktų generavimo algoritmai, pakeičiant jų generavimo logiką - daiktai ir priešai pradėti generuoti žaidėjui pasiekus tam tikrą atstumą nuo jų atsiradimo vietos, o ne užpildant visą lygį iškart.

Tiriant atsitiktinio turinio generavimo algoritmų prisitaikymo metriką pastebėta, kad žaidėjas neturi galimybių keisti turinio generavimo algoritmo veikimo. Problemai išspręsti buvo atlikti atsitiktinio turinio generavimo algoritmų patobulinimai, pridedant funkcionalumą, kad algoritmas generuotų skirtingus žaidimo lygius priklausomai nuo žaidėjo pasirinktų lygio dydžio, sunkumo bei žaidimo režimo nustatymų.

Atsitiktinio turinio generavimo algoritmų sugeneruotų lygių balanso tyrimas parodė, kad sugeneruoti lygiai nesugeneruoja pakankamai resursų susidoroti su sugeneruotais lygio priešais. Problemai išspręsti buvo sukurtas generavimo algoritmo patobulinimas, algoritmui atkreipiant dėmesį į žaidėjo turimus ginklus prieš generuojant atsitiktinius žaidėjui naudingus daiktus lygyje, o daiktų balansas yra generuojamas pagal pasirinkta žaidimo sunkumą.

Žaidimo išplečiamumo tyrimas atskleidė, kad daiktų ir priešų generavimas užtrunka ilgiau už lygio generavimą. Problema išspręsta greitaveikos metu sukurtu daiktų bei priešų generavimo algoritmų patobulinimu, pakeičiant funkcionalumą, kad daiktai ir priešai būtų sugeneruojami žaidėjui priėjus pakankamai arti prie jų atsiradimo vietų.

4. Eksperimentinė dalis

Eksperimentinėje dalyje atliekamas tyrimo metu sukurtų žaidimo atsitiktinio turinio generavimo algoritmų patobulinimų įvertinimas panaudojant tyrimo metu nustatytas algoritmų kokybės metrikas. Eksperimentų rezultatai palyginami su prieš modifikavimą gautais metrikų rezultatais ir padaromos išvados ar patobulinimai išties padėjo pagerinti pasirinktas kokybes metrikas. Šios metrikos yra:

- Įvairovė: ar įvairus yra sukurtas turinys?
- Greitaveika: kaip sugeneruotas turinys paveikia žaidimo greitaveiką?
- Prisitaikymas: ar generuojamas turinys gali prisitaikyti prie žaidimo konteksto, režimų, sudėtingumų?
- Balansas: ar generuojamas turinys turi galimybę prisitaikyti prie žaidimo eigos ir ją balansuoti?
- Išplečiamumas: ar generuojamas turinys turi galimybę keisti savo dydį ir ar didesniu mastu generuojamas turinys yra sukuriamas pakankamai greitai?

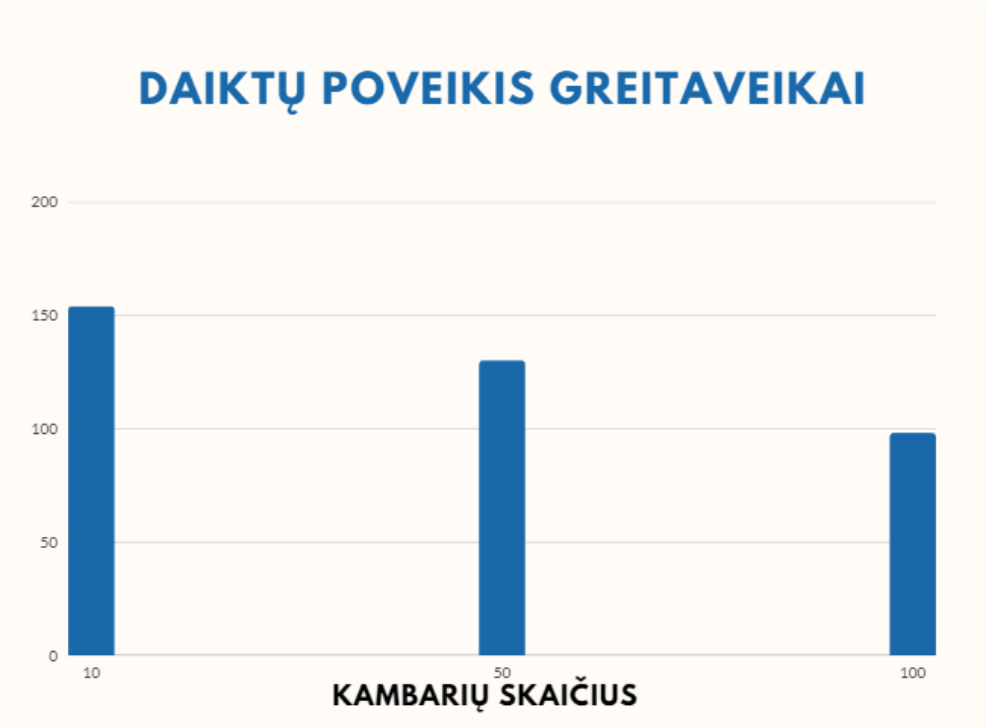
4.1. Įvairovės eksperimentas

Įvairovės tyrimo metu buvo apsirašyta formulė, pagal kurią skaičiuojama atsitiktinio turinio generavimo įvairovės metrika. Įvairovei gerinti atliktų patobulinimų poveikį lygio generavimo algoritmui naudojama ta pati formulė: H(X) = -∑(p(x) log p(x)), kur p(x) yra tam tikro turinio, šiuo atveju, kambario, tikimybė atsirasti lygyje. Atlikus patobulinimą, p(x) tampa ne tik vienu iš visų galimų kambariu, bet vienu iš visų galimų kambarių, padauginus iš vienos iš visų galimų kambario tekstūrų, bei padauginus iš vienos iš visų galimų kambario objektų kombinacijų. Algoritmas gali pasinaudoti 13 skirtingų kambarių, 9 skirtingų tekstūrų ir vidutiniškai 15 skirtingų kambario objektų, kurie gali sudaryti kombinacijas. Apskaičiuojant mažo dydžio lygį, kambarių skaičius yra 10, taigi gaunamas $p(x) = \left(\frac{1}{13} * \frac{1}{9} * \frac{1}{2}^{15}\right)^{10}$. Reikšmę įstačius į formulę gaunama: $H(X) = -(3833856)\left(\frac{1}{13} * \frac{1}{9} * \frac{1}{2}^{15}\right)^{10} \log 2\left(\left(\frac{1}{13} * \frac{1}{9} * \frac{1}{2}^{15}\right)^{10}\right) = -(3833856)^{10} * \left(\frac{1}{3833856}\right)^{10} * -10log2\left(\frac{1}{3833856)}\right) = 10log2(3833856) = \sim 218.70$. Vadinasi 10 kambarių lygiui su visais įgyvendintais atsitiktiniais patobulinimais reikia virš 218 bitų norint išsaugoti jo informaciją.

4.2. Greitaveikos eksperimentas

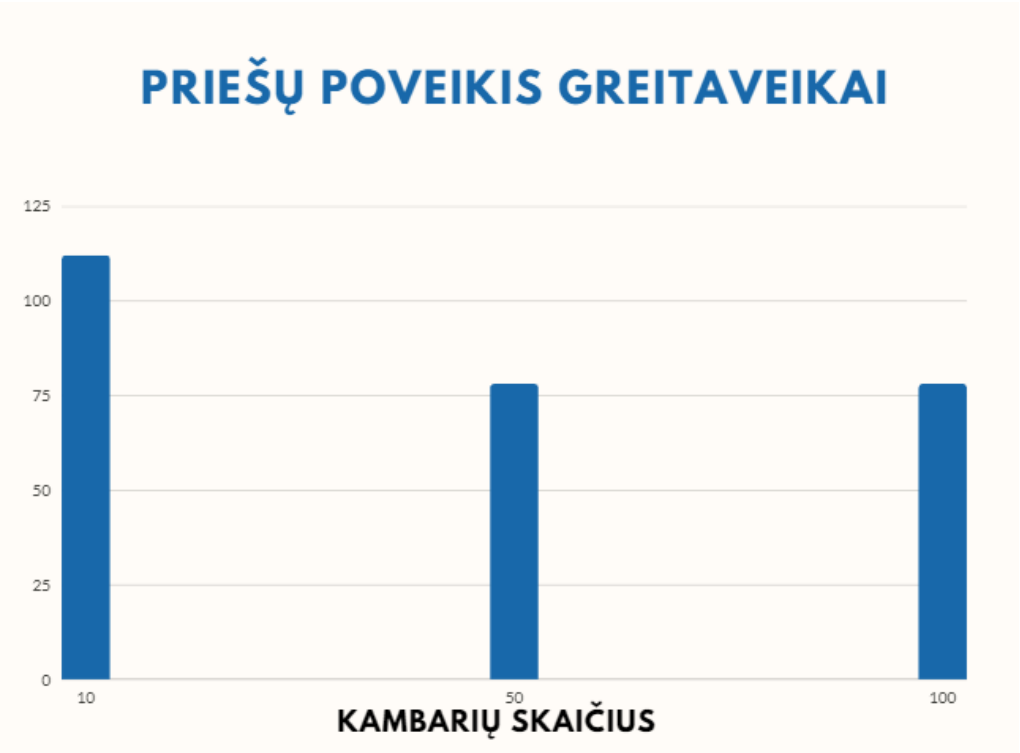
Greitaveikos eksperimentui naudojamas tyrimo metu naudotas kompiuteris: AMD Ryzen 5 3600x 4.25GHz procesorius, GTX 1080 vaizdo plokštė, 16GB RAM. Greitaveikai matuoti naudojamas kadrų per sekundę skaičius. Kuo daugiau kadrų per sekundę gali sugeneruoti kompiuteris, tuo yra geriau. Sklandžiam žaidimui, kompiuteris turėtų sugeneruoti bent 60 kadrų per sekundę.

Pirmiausia atliekamas eksperimentas daiktų generavimo algoritmui su patobulinimais. Daiktų skaičius lygyje skiriasi priklausomai nuo pasirinkto lygio dydžio, tiksliau, kambarių skaičiaus, todėl kadrų per sekundę skaičius matuojamas pagal 10, 50 ir 100 kambarių lygius. Kadrų per sekundę skaičius nepriklauso nuo sugeneruotų kambarių skaičiaus dėl įgyvendintų kambarių optimizavimo algoritmų projekte, tačiau tai nepaveikia sugeneruotų daiktų. Norint, kad eksperimentas daiktams būtų kuo tikslesnis, eksperimento metu išjungiamas priešų generavimo algoritmas.



58 pav. Daiktų poveikis greitaveikai po patobulinimų

Toliau atliekamas eksperimentas priešų generavimo algoritmui su patobulinimais. Priešų skaičius lygyje skiriasi nuo pasirinkto lygio dydžio, todėl kadrai per sekundę matuojami ties 10, 50 ir 100 sugeneruotų kambarių. Norint, kad eksperimentas sugeneruotiems priešams būtų kuo tikslesnis, eksperimento metu išjungiamas daiktų generavimas.



59 pav. Priešų poveikis greitaveikai po patobulinimų

4.3. Prisitaikymo eksperimentas

Prisitaikymo eksperimentui naudojama tyrimo metu naudota formulė: $PS = PM/TP$, kur PS yra personalizavimo taškų kiekis, PM yra kiek skirtingų nustatymų suteikia atsitiktinio turinio generavimo algoritmas, o TP yra visų galimų žaidėjo norų skaičius. Kuo sistemos PS yra didesnis, tuo yra geriau, nes sistema gali labiau prisitaikyti prie žaidėjų norų. Galime paimti TP kaip statinį skaičių, sakydami, kad žaidėjas gali norėti keisti 3 nustatymus, kurių kiekvienas turėtų po 3 pasirinkimus. Tokiu atveju $TP = 3 * 3 * 3 = 27$ skirtingi pasirinkimai, kurių gali norėti žaidėjas. Tyrimo metu buvo atliktas patobulinimas, kuriame lygiai yra generuojami dviem skirtingais būdais priklausomai nuo pasirinkto žaidimo režimo, lygių dydžiai yra generuojami trimis skirtingais būdais priklausomai nuo pasirinkto lygio dydžio, o priešai, žaidėjui naudingi daiktai ir skirtingi kambariai yra generuojami trimis skirtingais būdais priklausomai nuo pasirinkto žaidimo sudėtingumo, todėl $PM = 2 * 3 * 3 = 18$, rodantis, kad atsitiktinio generavimo sistemos 18 skirtingų pasirinkimų kombinacijų. Taigi mūsų modifikuotos sistemos $PS = \frac{18}{27}$.

4.4. Balanso eksperimentas

Balanso eksperimentui naudojama balanso tyrimo metu apsibrėžtas balanso koeficientas: Tam panaudojama formulę, kuri apskaičiuoja koeficientą tarp reikalingų resursų kiekio ir vidutiniškai sugeneruojamų resursų kiekio. Jeigu koeficientas mažesnis už 1, reiškia, kad lygis nesugeneruoja pakankamai resursų lygiui įveikti, 1 reiškia, kad generavimas yra subalansuotas ir generuojamų priešų skaičius ir sugeneruojamų resursų skaičius norint jiems įveikti yra lygus. Koeficientui būnant daugiau už vieną, reiškia, kad sugeneruojama daugiau resursų negu reikia, todėl žaidėjas gali padaryti daugiau klaidų ir žaidimas tampa lengvesnis. Formulė: $balanso\ koeficientas = \frac{(galima\ padaryti\ žala)}{(reikalinga\ padaryti\ žala)}$. Kadangi žaidimas turi tris sudėtingumo lygius, koeficientas turėtų būti virš 1 lengvam lygiui, kuo arčiau 1 vidutiniam lygiui ir mažiau už 1 sudėtingam lygiui. Atlikus patobulinimus, tiek daiktų generavimo algoritmas, tiek priešų generavimo algoritmas geba generuoti turinį priklausomai nuo pasirinkto žaidimo sudėtingumo, todėl tampa įmanoma subalansuoti žaidimą trims skirtingiems sudėtingumams.

Balansas lengvam sudėtingumui: pirmiausia apskaičiuojamas vidutinis amunicijos atsiradimo taškų skaičius: $\frac{77}{13} * 50 = \sim 296$. Toliau apskaičiuojama vidutiniškai sugeneruota amunicija lygyje: $296 * 0.7 * 0.80 * 30 = 4972.8$. Galima padaryti žala: $(300 + 4972.8) * 35 = 184548$. Kadangi žaidime galima turėti du skirtingus ginklus, šį skaičių galima padvigubinti: $184548 * 2 = 369096$ Vidutinės gyvybės taške: $100 * 2 * 0.5 + 300 * 1.5 * 0.14 + 200 * 1 * 0.1 + 2000 * 1 * 0.01 + 250 * 1 * 0.1 + 100 * 2 * 0.15 = 258$. Vidutinis taškų skaičius: $\frac{234}{13} * 50 = 900$. Vidutinės gyvybės lygyje: $900 * 258 = 232200$. Galiausiai balanso koeficientas: $\frac{369096}{232200} = \sim 1.59$.

Balansas vidutiniam sudėtingumui: pirmiausia apskaičiuojamas vidutinis amunicijos atsiradimo taškų skaičius: $\frac{77}{13} * 50 = \sim 296$. Toliau apskaičiuojama vidutiniškai sugeneruota amunicija lygyje: $296 * 0.6 * 0.7 * 30 = 3729.6$. Galima padaryti žala: $(300 + 3729.6) * 35 = 141036$. Kadangi žaidime galima turėti du skirtingus ginklus, šį skaičių galima padvigubinti: $141036 * 2 = 282072$ Vidutinės gyvybės taške: $100 * 2.5 * 0.1 + 300 * 2 * 0.15 + 200 * 1.5 * 0.15 + 2000 * 1 *$

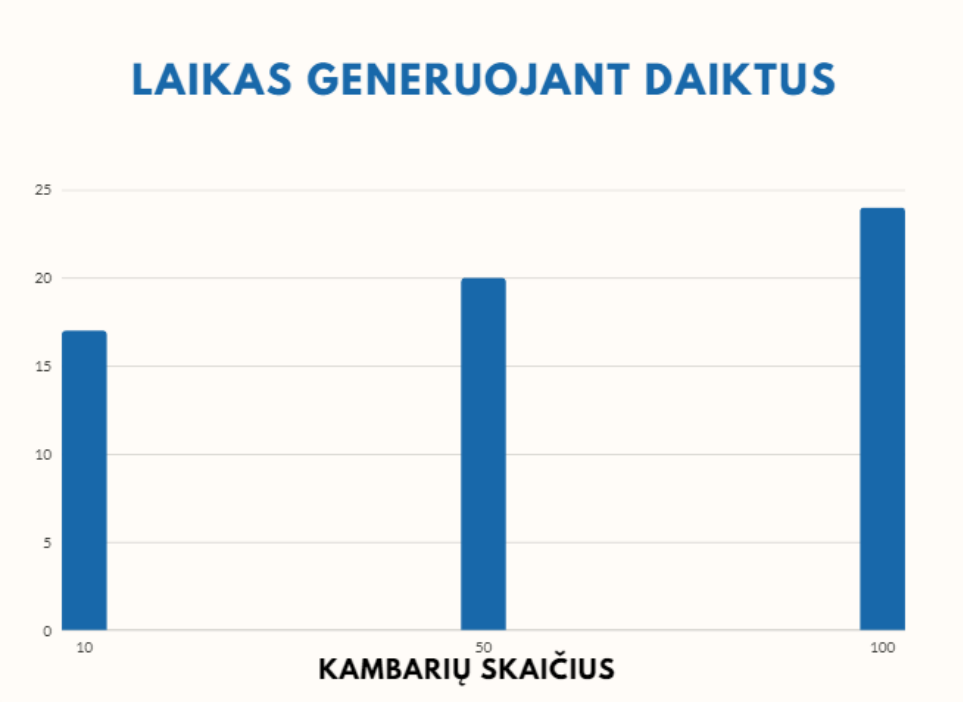
$0.03 + 250 * 1 * 0.15 + 100 * 2.5 * 0.16 = 297.5$. Vidutinis taškų skaičius: $\frac{234}{13} * 50 = 900$. Vidutinės gyvybės lygyje: $900 * 297.5 = 267750$. Galiausiai balanso koeficientas: $\frac{282072}{267750} = \sim 1.05$.

Balansas sunkiausiam sudėtingumui: pirmiausia apskaičiuojamas vidutinis amunicijos atsiradimo taškų skaičius: $\frac{77}{13} * 50 = \sim 296$. Toliau apskaičiuojama vidutiniškai sugeneruota amunicija lygyje: $296 * 0.5 * 0.6 * 30 = 2664$. Galima padaryti žala: $(300 + 2664) * 35 = 103740$. Kadangi žaidime galima turėti du skirtingus ginklus, šį skaičių galima padvigubinti: $103740 * 2 = 207480$ Vidutinės gyvybės taške: $100 * 3 * 0.05 + 300 * 2.5 * 0.2 + 200 * 2 * 0.2 + 2000 * 1.5 * 0.04 + 250 * 2 * 0.18 + 100 * 3 * 0.17 = 506$. Vidutinis taškų skaičius: $\frac{234}{13} * 50 = 900$. Vidutinės gyvybės lygyje: $900 * 506 = 455400$. Galiausiai balanso koeficientas: $\frac{207480}{455400} = \sim 0.46$.

4.5. Išplečiamumo eksperimentas

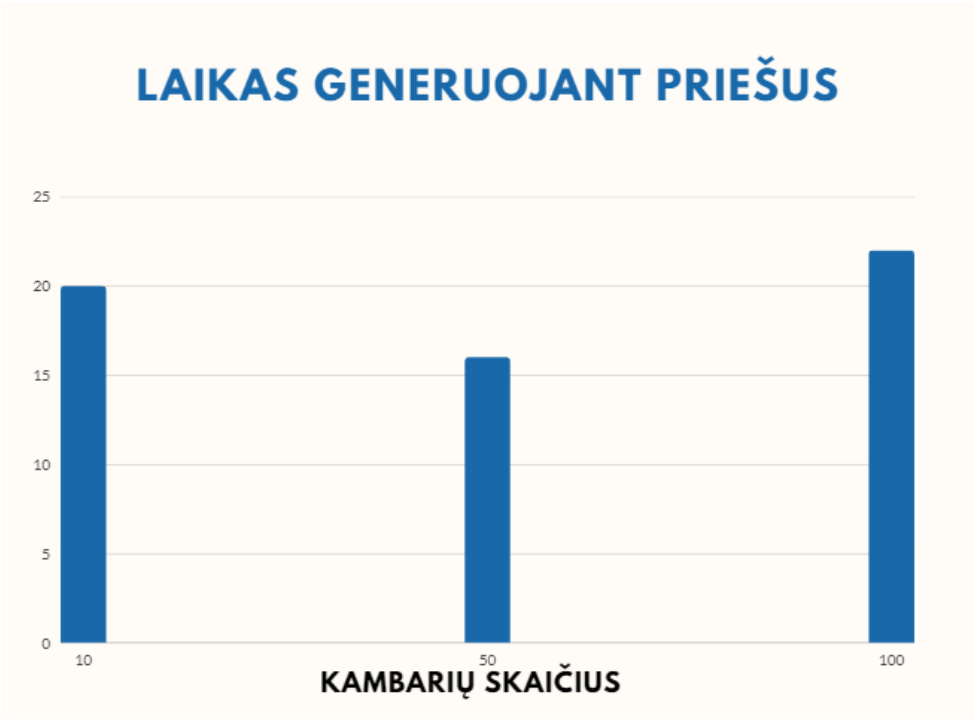
Išplečiamumo eksperimentui apskaičiuojami išplečiamumo tyrimo metu sukurtų daiktų ir priešų algoritmų patobulinimų poveikis lygio generavimo laikui. Sukurti algoritmų patobulinimai padaro, kad sugeneruotas daiktų ir priešų skaičius lygio pradžioje būtų užtektinas žaidimo pradžiai, užpildant vietą aplink žaidėjo atsiradimo vietą. Likęs daiktų ir priešų generavimas yra išskaidomas per visą žaidimo eigą, žaidėjui keliaujant per lygį. Vieno daikto ar priešo sugeneravimas žaidimo eigoje vidutiniškai užtrunka apie 1 ms, todėl generavimas žaidimo eigoje nesukelia nepatogumų ar delsos žaidėjui.

Daiktų generavimo laikui lygio pradžioje rasti panaudojamas išplečiamumo tyrimo metu sukurtas metodas, kuris skaičiuoja laiką nuo daiktų generavimo pradžios iki daiktų generavimo pabaigos žaidimo lygio krovimo metu.



60 pav. Laikas reikalingas sugeneruoti daiktams lygio pradžioje su patobulinimais

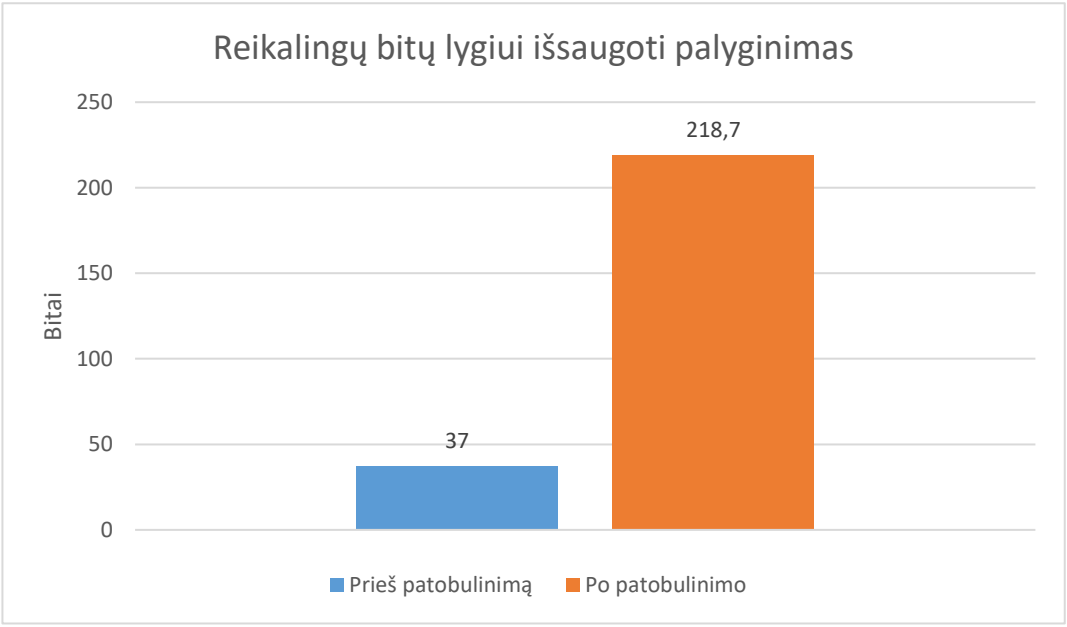
Priešų generavimo laikui lygio pradžioje rasti panaudojamas išplečiamumo tyrimo metu sukurtas metodas, kuris skaičiuoja laiką nuo priešų generavimo pradžios iki priešų generavimo pabaigos žaidimo lygio krovimo metu.



61 pav. Laikas reikalingas sugeneruoti priešams lygio pradžioje su patobulinimais

4.6. Rezultatų palyginimas

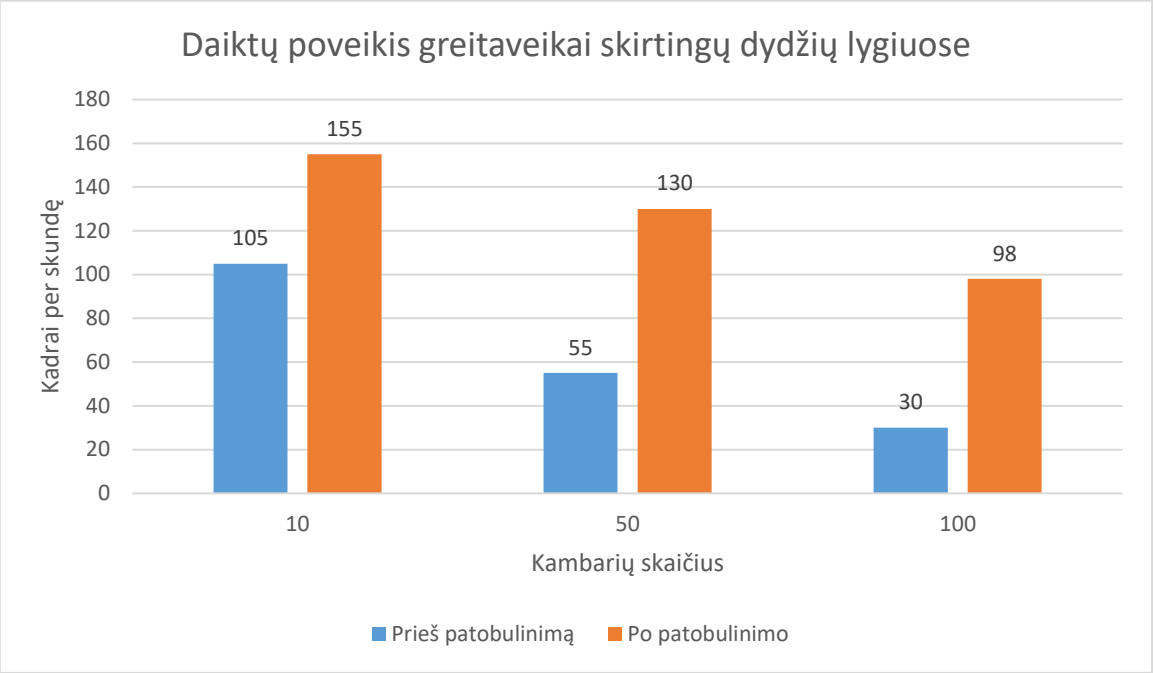
Įvairovei įvertinti apskaičiuotų bitų kiekis, reikalingas 10 kambarių dydžio lygio informacijai išsaugoti, tyrimo metu prieš patobulinimus buvo apie 37 bitus. Atlikus patobulinimus, reikalingų bitų skaičius išaugo iki apie 218.7 bitų.



62 pav. Įvairovės rezultatų palyginimas prieš ir po patobulinimo

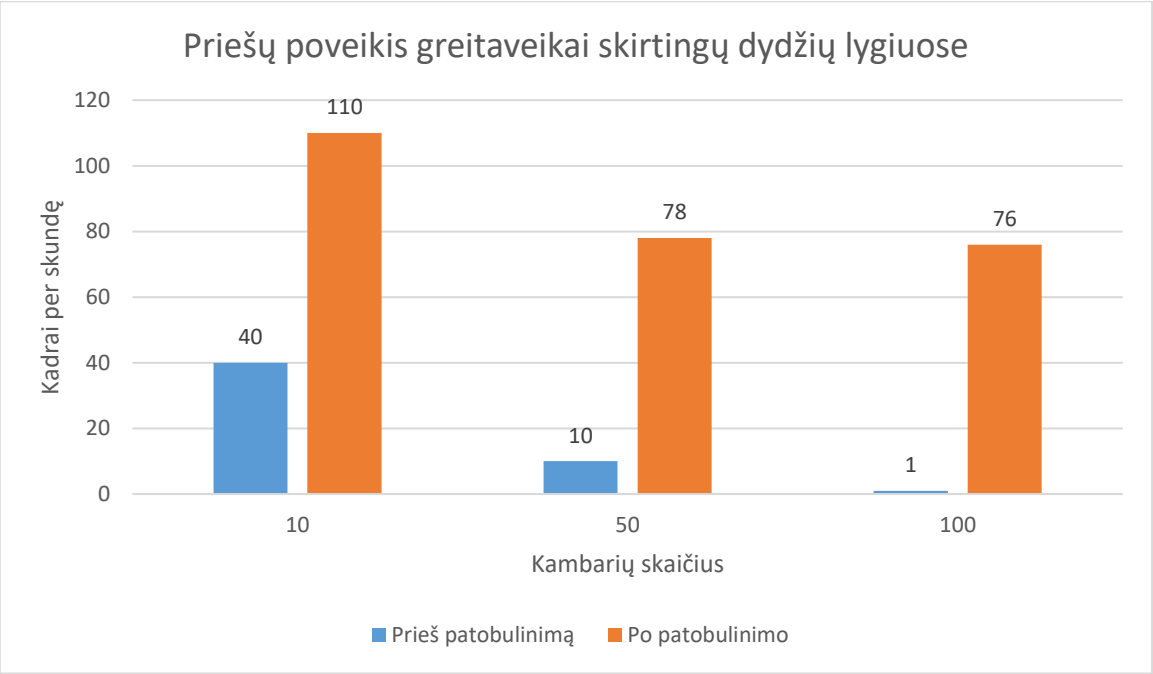
Reikalingų lygiui išsaugoti bitų skaičius išaugo beveik 6 kartus. Tai rodo, kad žaidimo lygių generavimo algoritmas tapo daug įvairesnis, turintis daugiau galimybių sugeneruoti lygius, kurie skirsis. Žaidžiant žaidimą iš naujo, tikimybė, kad žaidėjas gaus žaisti lygiai tokį patį lygį tampa žymiai mažesnė.

Greitaveikai įvertinti apskaičiuotų kadrų per sekundę skaičius 10, 50 ir 100 kambarių dydžio lygiuose tyrimo metu prieš patobulinimą daiktų generavimo algoritmui jau ties 50 kambarių krito žemiau 60 kadrų per sekundę, kurių tikimasi iš žaidimo, kad šis būtų sklandus. Po patobulinimų daiktų generavimo algoritmui, kadrų per sekundę skaičius ties visų dydžių lygiais sugebėjo išlaikyti daugiau nei 60 kadrų per sekundę, todėl patobulinimas padėjo žaidimo greitaveikai pagerinti kadrų per sekundę skaičių.



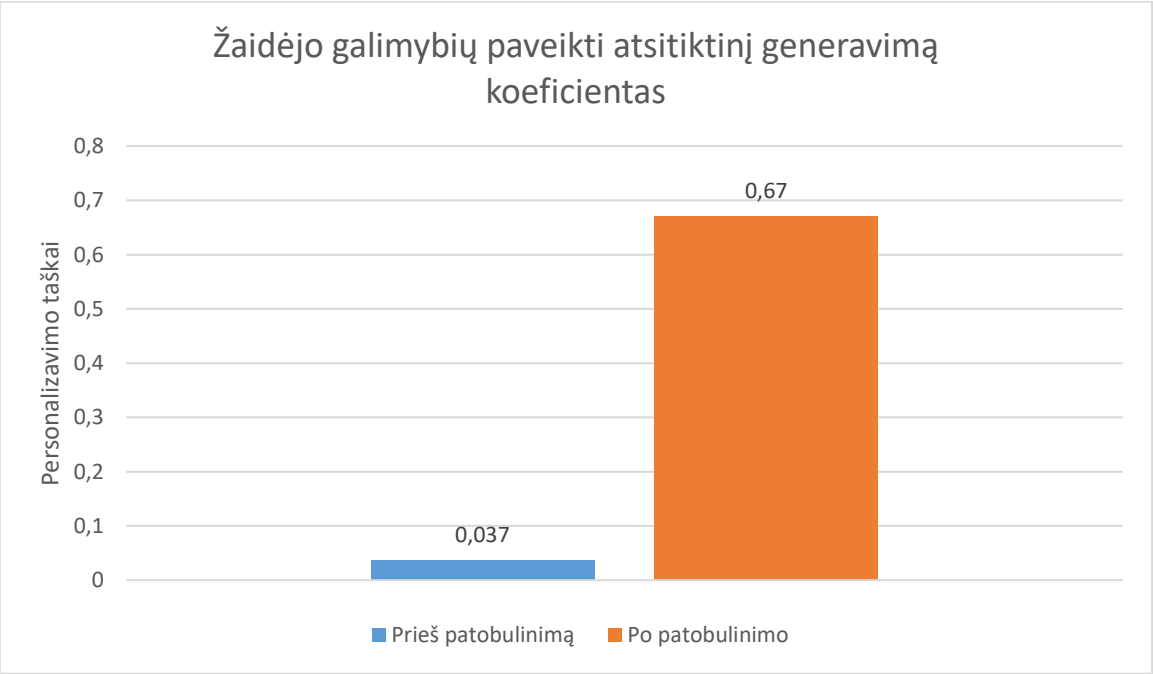
63 pav. Daiktų poveikio rezultatų palyginimas prieš ir po patobulinimo

Greitaveikai įvertinti apskaičiuotų kadrų per sekundę skaičius 10, 50 ir 100 kambarių dydžio lygiuose tyrimo metu prieš patobulinimą priešų generavimo algoritmui jau ties 10 kambarių krito žemiau 60 kadrų per sekundę, kurių tikimasi iš žaidimo, kad šis būtų sklandus. Po patobulinimų priešų generavimo algoritmui, kadrų per sekundę skaičius ties visų dydžių lygiais sugebėjo išlaikyti daugiau nei 60 kadrų per sekundę, todėl patobulinimas padėjo žaidimo greitaveikai pagerinti kadrų per sekundę skaičių.



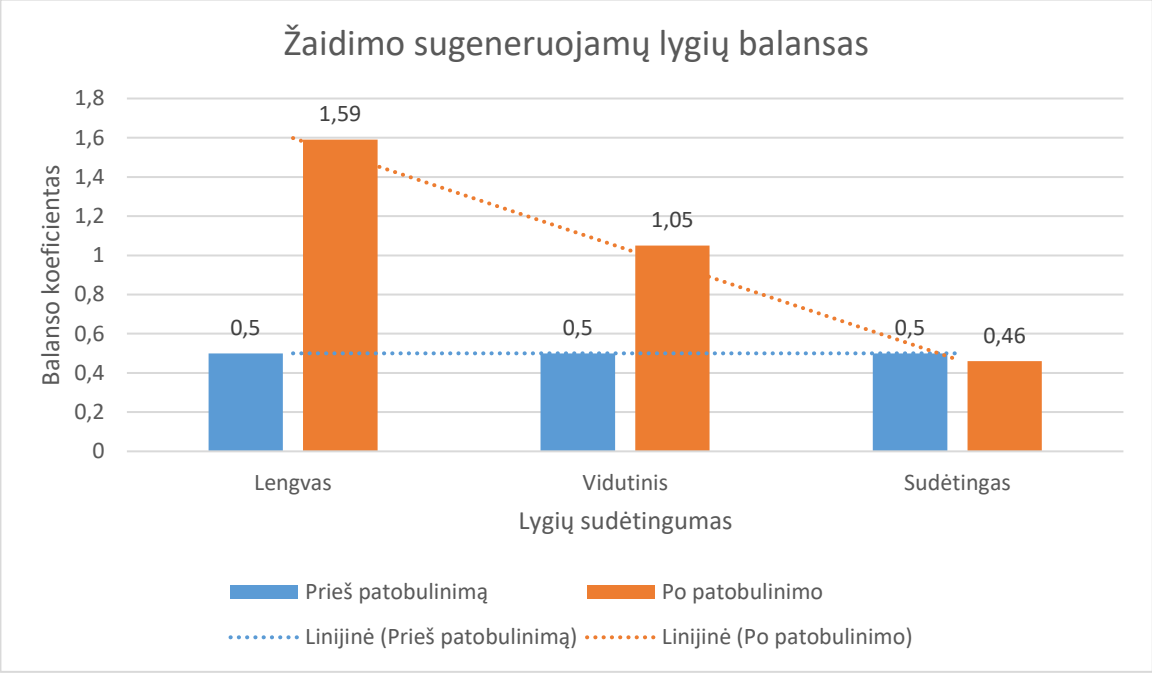
64 pav. Priešų poveikio greitaveikai palyginimas prieš ir po patobulinimų

Prisitaikymui įvertinti buvo skaičiuojami personalizavimo taškai. 1 reiškia, kad atsitiktinio turinio generavimo algoritmai atsižvelgia į visų žaidėjų norus, tuo tarpu skaičius mažesnis už vieną rodo, kad algoritmai neturi galimybių išpildyti visų žaidėjo norų. Žaidėjo norams paimtas statinis 3 pasirinkimų po 3 galimybes skaičius, kuris yra 27. Prieš patobulinimus, tyrimo metu nustatyta, kad algoritmas gali sukurti tik vieno tipo lygius, neatsižvelgdamas į žaidėjo norus, todėl personalizavimo taškų skaičius buvo apie 0.037. Atlikus patobulinimus, įgyvendinti 3 pasirinkimai, kurių vienas iš jų turi 2 galimybes, o likusieji po 3 galimybes, todėl personalizavimo taškų skaičius išaugo iki apie 0.67. Išaugęs skaičius rodo, kad atlikti patobulinimai buvo sėkmingi ir padidino žaidėjo galimybes keisti algoritmo generavimą.



65 pav. Personalizavimo taškai prieš ir po patobulinimo

Balansui matuoti buvo skaičiuojama, ar žaidimo lygiai sugeneruoja pakankamai resursų norint įveikti lygį. Tam naudojamas balanso koeficientas. Jeigu koeficientas mažesnis už 1, reiškia, kad lygis nesugeneruoja pakankamai resursų lygiui įveikti, 1 reiškia, kad generavimas yra subalansuotas ir generuojamų priešų skaičius ir sugeneruojamų resursų skaičius norint jiems įveikti yra lygus. Koeficientui būnant daugiau už vieną, reiškia, kad sugeneruojama daugiau resursų negu reikia, todėl žaidėjas gali padaryti daugiau klaidų ir žaidimas tampa lengvesnis. Kadangi žaidimas turi tris sudėtingumo lygius, koeficientas turėtų būti virš 1 lengvam lygiui, kuo arčiau 1 vidutiniam lygiui ir mažiau už 1 sudėtingam lygiui. Atlikus balanso tyrimą, gautas koeficientas buvo ~0.5, kas reiškė, kad žaidimas nesugeneruoja pakankamai resursų lygiui įveikti. Taip pat, žaidimas neturėjo galimybės sukurti skirtingus balansus skirtingiems žaidimo sunkumo režimams. Atlikus patobulinimus, generavimo algoritmai generavo ne tik labiau subalansuotą turinį, tačiau ir įgavo galimybę skirtingiems žaidimo sudėtingumams generuoti skirtingai subalansuotą turinį.

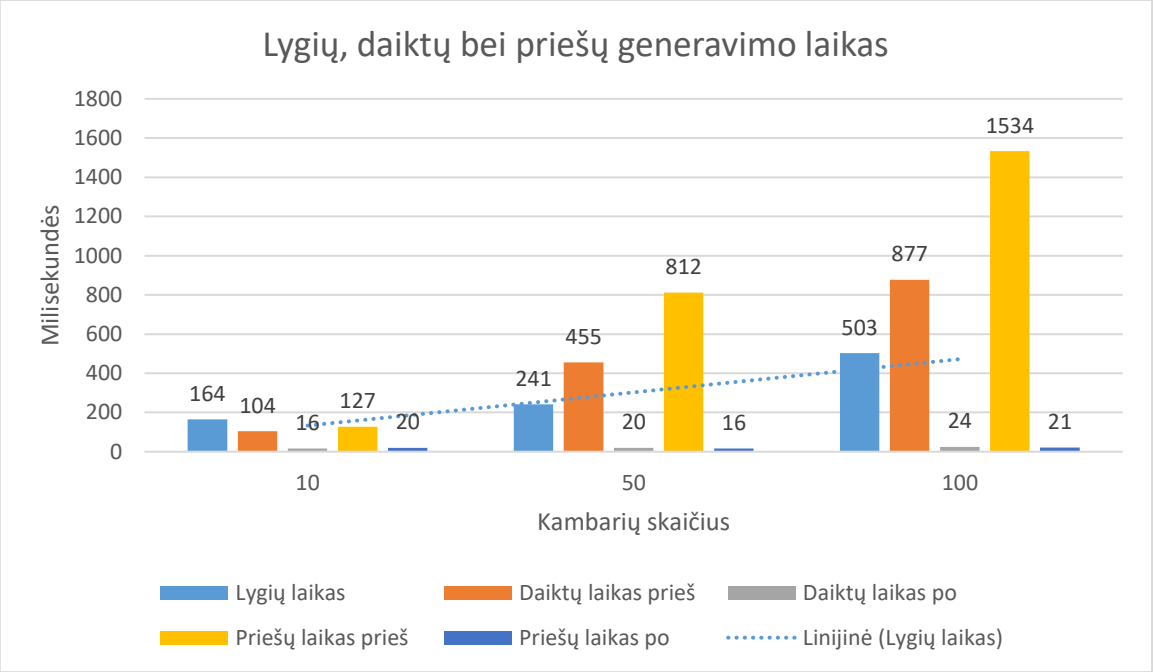


66 pav. Lygių sugeneruotas balansas prieš ir po patobulinimų

Rezultatai rodo, kad generuojamas turinys yra beveik subalansuotas ties vidutiniu sunkumo režimu, ko ir buvo siekiama. Tuo tarpu, lengvas sunkumas yra subalansuotas žaidėjo naudai, suteikiant jam daugiau resursų priešams nukauti, o žaidėjui leidžiant atlikti daugiau klaidų, tokių kaip nepataikyti į priešą. Sudėtingas sunkumas yra žemiau vieneto, kas reiškia, kad žaidimo lygis yra sugeneruotas žaidėjo nenaudai, suteikiant žaidėjui mažiau resursų nukauti visiems priešams, todėl žaidėjas turi naudoti šaltuosius ginklus norint taupyti amuniciją, šaudyti priešus į jų silpnas vietas, taip padarant daugiau žalos, taip pat ir išnaršyti visą lygį norint surasti kuo daugiau resursų. Taškinė linija taip pat rodo, kad kuo sudėtingesnis žaidimo režimas, tuo mažiau subalansuotas yra žaidimo lygis, ko ir buvo siekiama patobulinimais. Taigi patobulinimai ne tik labiau subalansavo žaidimo lygius, tačiau ir sukūrė žaidimo sunkėjimą pagal pasirinktą sunkumo režimą.

Išplečiamumui matuoti buvo skaičiuojama, kaip greitai lygių, daiktų bei priešų generavimo algoritmai sugeba sugeneruoti turinį žaidimo pradžioje. Tyrimo metu nustatyta, kad žaidimo lygio generavimas užtrunka nuo 150 ms iki 500 ms, priklausomai nuo generuojamo lygio dydžio. Tai nėra ilgas laikas, todėl lygių generavimui patobulinimų nebuvo pasiūlyta. Tuo tarpu daiktų bei priešų

generavimas prieš patobulinimus užtruko ilgiau nei pačio lygio generavimas, todėl pailgėdavo viso lygio krovimo laikas. Atlikus patobulinimus buvo tikimasi, kad daiktų bei priešų generavimo laiką galima būtų sumažinti, kad jie būtų trumpesni arba lygūs lygio generavimui, kad nebūtų prailgintas lygio krovimo laikas.



67 pav. Daiktų ir priešų generavimo laikas prieš ir po patobulinimų

Rezultatai rodo, kad patobulinimai buvo sėkmingi. Lygių generavimas, taip pat pažymėtas taškine linija, buvo mažesnis už daiktų bei priešų generavimo laiką generuojant 50 bei 100 kambarių dydžio lygius, kas prailgindavo krovimo laiką. Po patobulinimų, daiktų bei priešų generavimo laikas žymiai nukrito ir yra mažesnis už lygio generavimo laiką, todėl lygio krovimas užtrunka tik tiek, kiek yra generuojamas lygis.

4.7. Eksperimentinės dalies išvados

Įvairovės eksperimento metu buvo palyginti atsitiktinių lygių algoritmo sugeneruoto turinio įvairovės vertinimo metrikų rezultatai prieš ir po atliktų patobulinimų. Po eksperimento paaiškėjo, kad atlikti pakitimai sugeneruoja didesnę įvairovę, nei buvo prieš tai: įvairovės metrika išaugo beveik šešis kartus. Tai leidžia daryti prielaidą, jog algoritmo patobulinimas buvo efektyvus.

Atliekant greitaveikos eksperimentą buvo palyginti atsitiktinio turinio algoritmų poveikio greitaveikai rezultatai prieš ir po atliktų patobulinimų. Eksperimentas parodė, kad atlikti pakitimai pagerina žaidimo greitaveiką: kadrų per sekundę skaičius išlieka virš 60 net ir didžiausiuose lygiuose, ko nebūdavo iki pakitimų realizavimo. Taigi, galima daryti išvada, kad algoritmų poveikio žaidimo greitaveikai patobulinimas buvo efektyvus.

Prisitaikymo eksperimente buvo palyginti atsitiktinio turinio generavimo algoritmų prisitaikymo prie žaidėjo norų ir nustatymų koeficientai prieš ir po patobulinimų. Eksperimentas parodė, kad algoritmų veikimo prisitaikymas prie žaidėjo norų išaugo nuo 1 iki 18 variantų. Taigi, eksperimento rezultatai rodo, kad algoritmo patobulinimas buvo veiksmingas.

Atliekant balanso eksperimentą buvo palyginti algoritmų sugeneruojamų resursų ir priešų koeficientai prieš ir po patobulinimo. Atliktas eksperimentas parodė, kad algoritmai po patobulinimo sugeba sugeneruoti trijų skirtingų balansų turinį, lyginant su vienu nesuderinto balanso lygių prieš patobulinimą. Lengvo sudėtingumo turinys yra sugeneruotas žaidėjo naudai (koeficientas buvo apytiksliai 1.5), vidutinis yra subalansuotas ir apskaičiuotas koeficientas siekė beveik 1, o balansas sunkiausiam sudėtingumui yra subalansuotas žaidėjo nenaudai (koeficiento reikšmė buvo 0.5). Taigi, eksperimento rezultatai rodo, kad patobulinimai yra sėkmingi ir suteikia ne tik geresnį žaidimo balansą, tačiau ir pasirinkimą žaidėjui subalansuoti žaidimą pagal savo sugebėjimus.

Išplečiamumo eksperimento metu buvo palygintas laikas reikalingas sugeneruoti atsitiktinį žaidimo turinį prieš ir po patobulinimų. Atlikus eksperimentą pastebėta, kad po atliktų patobulinimų žymiai sutrumpėjo daiktų bei priešų generavimo laikas (jis sumažėjo nuo apytiksliai 1500 ms iki 20 ms), kas pagerina visą žaidimo krovimo laiką. Taigi galima teigti, kad atlikti patobulinimai buvo efektyvus, o žaidimo turinio generavimo laikas pagerintas.

Išvados

1. Analizės metu nustatyta, kad yra daug skirtingų būdų generuoti žaidimų žemėlapius. Atlikta analizė parodė, kad vienas optimaliausių būdų generuoti žemėlapius šaudyklės žanro žaidimams yra procedūrinis labirinto metodas. Šio metodo generuojamų lygių testavimas reikalauja mažiausiai resursų. Žaidimus, naudojančius šį metodą, yra lengviau atnaujinti ir išplėsti papildant generavimo masyvą papildomais kambariais. Tačiau, šis būdas nesugeba generuoti didelės įvairovės žemėlapių, todėl tai yra kompromisas tarp sugeneruotų lygių kokybės ir įvairovės.
2. Tyrimo metu buvo pasirinktos penkios atsitiktinio turinio generavimo algoritmų kokybės metrikos: įvairovė, greitaveika, prisitaikymas, balansas ir išplečiamumas. Atliekant kiekvienos metrikos tyrimą buvo nustatyta, kad algoritmai veikia ne optimaliai ir jų veikimas galėtų būti pagerintas. Minėtoms metrikoms pagerinti buvo įgyvendinti algoritmo patobulinimo sprendimai. Įvairovei padidinti buvo sukurti papildomi elementai, iš kurių gali būti generuojami lygiai. Greitaveikai ir išplečiamumui pagerinti buvo patobulinti daiktų bei priešų generavimo algoritmai išskaidant jų generavimą per visą žaidimo eigą. Prisitaikymo metrikai gerinti algoritmų veikimas buvo modifikuotas taip, kad jų sugeneruotas turinys priklausytų nuo naudotojo pasirinktų nustatymų. Balansui tobulinti, algoritmui buvo suteiktas papildomas funkcionalumas, keičiantis atsitiktinių turinio generavimo algoritmų sugeneruojamą turinį, priklausomai nuo žaidėjo pasirinkto sunkumo lygio.
3. Eksperimentinėje tyrimo dalyje buvo tirti tyrimo dalyje priimti sprendimai ir jų įtaka tiriamoms metrikoms. Atlikus eksperimentinį tyrimą paaiškėjo, jog algoritmus patobulinti pavyko sėkmingai: tai įrodo eksperimentų metu gauti metrikų rezultatai ir jų pokyčiai. Įvairovės metrika išaugo nuo 37 iki 218, o tai rodo, kad žaidimo lygiai yra įvairesni. Didžiausios apkrovos metu greitaveikos metrika (t.y. kadrai per sekundę) išaugo nuo 1 iki 75, o tai užtikriną sklandų žaidimo veikimą. Prisitaikymo rodiklis išaugo nuo 1 iki 18; tai rodo, kad algoritmas geba prisitaikyti prie didesnės žaidėjo norų įvairovės. Atlikus balanso eksperimentą buvo nustatyta, kad po patobulinimo, algoritmas geba sugeneruoti trijų skirtingų balansų lygį, jį vertinant balanso koeficientu (lengvo sudėtingumo koeficientas siekė 1.5, vidutinio — 1, o sunkaus — 0.5). Išplečiamumo rodiklis eksperimentinio tyrimo metu parodė, jog algoritmo pokyčiai buvo efektyvūs: daiktų bei priešų generavimo laikas sumažėjo nuo 1500 ms iki 20 ms, o tai paspartino žaidimo krovimo laiką.

Informacijos šaltinių sąrašas

1. Richard Rouse, „Game Design: Theory & Practice 2nd Edition“, p. 450, 2004.
2. William L. Raffe B.CompSc, „Personalized Procedural Map Generation in Games via Evolutionary Algorithms“, p. 5, 2014.
3. Taylor & Francis Group, „Procedural Generation in Game Design“, p. 6, 2017.
4. Noor Shaker, Julian Togelius Mark, J. Nelson, „Procedural Content Generation in Games“, p. 1, 2016.
5. Benefits of procedural generation [interaktyvus]. 2023 [žiūrėta 2023-04-08]. Prieiga per: https://subscription.packtpub.com/book/game-development/9781785886713/1/ch01lvl1sec13/benefits-of-procedural-generation
6. Joel Lee, How Procedural Generation Took Over The Gaming Industry [interaktyvus]. 2014. [žiūrėta 2023-04-13]. Prieiga per: https://www.makeuseof.com/tag/procedural-generation-took-gaming-industry/
7. Rogue Steam page [interaktyvus] [žiūrėta 2023-04-13]. Prieiga per: https://store.steampowered.com/app/1443430/Rogue/
8. Steam store search [interaktyvus]. 2023. [žiūrėta 2023-04-13]. Prieiga per: https://store.steampowered.com/search/?tags=1716%2C3959&category1=998&supportedlang=english&ndl=1
9. Eric Van Allen. Roguelike vs Roguelite: What‘s the difference between the two? [interaktyvus]. 2022. [žiūrėta 2023-04-13]. Prieiga per: https://www.destructoid.com/the-difference-between-roguelike-and-roguelite-games/#:~:text=Often%2C%20the%20difference%20in%20roguelite,making%20that%20the%20easiest%20signifier.
10. Ethan Hawkes, What Separates a Roguelike from a Roguelite? [interaktyvus]. 2013. [žiūrėta 2023-04-16]. Prieiga per: https://hardcoregamer.com/features/articles/what-separates-a-roguelike-from-a-roguelite/47151/
11. Rogue Legacy Steam page [interaktyvus] [žiūrėta 2023-04-13]. Prieiga per: https://store.steampowered.com/app/241600/Rogue_Legacy/
12. Roguelike Games – Steam Marketing Tool [interaktyvus] [žiūrėta 2023-04-13]. Prieiga per: https://games-stats.com/steam/?tag=roguelike&page=1
13. Hades Steam page [interaktyvus] [žiūrėta 2023-04-13]. Prieiga per: https://store.steampowered.com/app/1145360/Hades/
14. Simon Read, Gaming is booming and is expected to keep growing [interaktyvus]. 2022 [žiūrėta 2023-04-14]. Prieiga per: https://www.weforum.org/agenda/2022/07/gaming-pandemic-lockdowns-pwc-growth/
15. Kate Irwin, Valve‘s Steam Store breaks all-time highs with 28 million users [interaktyvus]. 2022 [žiūrėta 2023-04-14]. Prieiga per: https://www.inverse.com/input/gaming/valves-steam-store-breaks-all-time-highs-with-28-million-users
16. Brian Dean, Steam Usage and Catalog Stats for 2022 [interaktyvus]. 2021. [žiūrėta 2023-04-14]. Prieiga per: https://backlinko.com/steam-users
17. Tomasz Hachaj, „Creating dynamically changing world map for computer games with advanced image processing - a use case“, IEEE, p. 155-160, spal. 2016.

18. K. S. Emmanuel, C. Mathuram, A. R. Priyadarshi, R. A. George ir J. Anitha, „A Beginners Guide to Procedural Terrain Modelling Techniques“, IEEE, p. 212-217, 2019.
19. A. Ajith, S. R. Babu, S. K, S. A. K and M. V. Thomas, „A Novel Method in Procedural Maze Generation“, IEEE, p. 1-5, 2022.
20. Juan Rodriguez, „Procedural Level Generator“ [interaktyvus]. 2019. [žiūrėta 2023-04-26]. Prieiga per: https://assetstore.unity.com/packages/tools/ai/procedural-level-generator-136626
21. Michael Beukman, Steven James, Christopher Cleghorn, „Towards Objective Metrics for Procedurally Generated Video Game Levels“, Cornell University, p. 1-6, 2022.
22. „Shannon Diversity Index – The Diversity Function in R“ [interaktyvus]. [žiūrėta 2023-04-26]. Prieiga per: https://www.programmingr.com/shannon-diversity-index-the-diversity-function-in-r/
23. Steve Hoffman, „Improving Player Choices“ [interaktyvus]. 2004. [žiūrėta 2023-04-26]. Prieiga per: https://www.gamedeveloper.com/design/improving-player-choices
24. Ra´ul Lara-Cabrera, V´ıctor Rodr´ıguez-Fern´andez, Javier Paz-Sedano, David Camacho, „Procedural Generation of Balanced Levels for a 3D Paintball Game“., Department of Computer Engineering Universidad Aut´onoma de Madrid, p. 1-5.