Kauno technologijos universitetas

Informatikos fakultetas

Šaudyklės žanro žaidimų optimizacijos metodų tyrimas

Baigiamasis magistro studijų projektas

Domas Raulinaitis

Projekto autorius

assoc. prof. dr. Tomas Blažauskas

Vadovas

Kaunas, 2023

Kauno technologijos universitetas

Informatikos fakultetas

Šaudyklės žanro žaidimų optimizacijos metodų tyrimas

Baigiamasis magistro studijų projektas

Programų sistemų inžinerija (6211BX011)

Domas Raulinaitis

Projekto autorius

assoc. prof. dr. Tomas Blažauskas

Vadovas

dr. Šarūnas Packevičius

Recenzentas

Kaunas, 2023

Kauno technologijos universitetas

Informatikos fakultetas

Domas Raulinaitis

Šaudyklės žanro žaidimų optimizacijos metodų 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.

Domas Raulinaitis

Patvirtinta elektroniniu būdu

Kauno technologijos universitetas

Informatikos fakultetas

Baigiamojo magistro projekto užduotis

Projekto tema: Šaudyklės žanro žaidimų optimizacijos metodų tyrimas

Reikalavimai ir sąlygos (tikslinti pavadinimą pagal poreikį):

Vadovas / Vadovė _____

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

Raulinaitis Domas. Šaudyklės žanro žaidimų optimizacijos metodų tyrimas. Magistro studijų 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: kompiuterinis žaidimas, žaidimų kūrimas, žaidimų optimizacija.

Kaunas, 2023. 52 p.

Santrauka

Magistro studijų baigiamojo projekto metu sukurtas pirmo asmens šaudyklės žanro žaidimas. Žaidimą galima žaisti kartu su draugais – yra palaikomas kelių žaidėjų režimas (angl. *Multiplayer*). Visi kartu žaidžiantys žaidėjai turi bendrą tikslą – įvykdyti misiją, kuri yra nustatoma pagal pasirinktą žaidimo tipą. Žaidimas sukurtas naudojant „*Unity*“ žaidimų variklį. Analitinėje dalyje apžvelgiami žaidimuose naudojami našumo optimizacijos metodai ir technikos (objektų telkimas, nematomų objektų paslėpimas, detalumo lygis) bei gerosios žaidimų kūrimo praktikos kūrimui naudojant „*Unity*“ žaidimų variklį. Projektinėje dalyje aprašoma sukurto žaidimo architektūra ir įgyvendintos funkcijos. Tyrimo dalyje atliekamas žaidimo kokybės įvertinimas pagal šias metrikas: žaidimo generuojamo vidutinio kadrų kiekio per sekundę, operatyviosios atminties naudojimą ir sukompiliuoto žaidimo užimamą dydį. Tyrimo dalyje įvertinus kokybę, pateikiami kokybės patobulinimo sprendimai - objektų paslėpimo metodo patobulinimas pakeičiant naudojamą komponentą, tekstūrų maksimalaus leidžiamo dydžio sumažinimas, tekstūrų grupavimas į atlasus ir tekstūrų bei garsų suspaudimo metodų pritaikymas. Eksperimentinėje dalyje atliekamas sistemos patobulinimų vertinimas. Darbo pabaigoje pateikiamos išvados, kad tiriamam žaidimui nematomų objektų paslėpimo metodo naudojimas padidino vidutinį kadrų per sekundę kiekį, detalumo lygio metodo naudojimas įtakos vidutiniam kadrų per sekundę kiekiui nepadarė, objektų telkimo metodas sumažino generuojamų atminties šiukšlių kiekį, tekstūrų maksimalaus leidžiamo dydžio ribojimas sumažino operatyviosios atminties naudojimą, o sukompiliuoto žaidimo užimamą dydį sumažinti padėjo naudojami tekstūrų ir garso failų suspaudimo metodai.

Raulinaitis Domas. A Study of Optimization Methods 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: video game, game development, video game optimization.

Kaunas, 2023. 52 p.

Summary

A game of the first-person shooter genre was created during the final project of master's studies. The game works in a multiplayer mode where players have to cooperate and coordinate their actions with each other in order to successfully complete the mission assigned to them. The game is developed using the Unity game engine. The analytical part reviews performance optimization methods and techniques used in games (object pooling, occlusion culling, level of detail) and good game development practices for creating games using the Unity game engine. The design part describes the architecture of the developed game and the implemented functions. The research part evaluates the quality of the game based on the following metrics: the average number of frames per second generated by the game, the use of RAM and the size occupied by the compiled game build. After evaluating the quality in the research part, quality improvement solutions are presented - improvement of the method of hiding objects by changing the used component, reduction of the maximum allowed size of textures, grouping of textures into atlases and selecting proper texture and sound compression settings. In the experimental part, the evaluation of system improvements is carried out. At the end of the paper, the conclusions are presented that the use of occlusion culling to hide invisible objects increased the average number of frames per second, the use of the level of detail method did not affect the average number of frames per second, usage of object pooling reduced the amount of allocated memory garbage, limiting the maximum allowed size of textures reduced the use of RAM , and the compression methods used for textures and sound files helped to reduce the size of the compiled game build.

Turinys

Lentelių sąrašas

Paveikslų sąrašas

Santrumpų ir terminų sąrašas

Santrumpos:

GB – gigabaitas.

MB – megabaitas.

CPU – centrinis procesorius.

GPU – grafinis procesorius, grafikos plokštė.

RAM – kompiuterio operatyvioji atmintis.

Terminai:

3D arba **trimatė erdvė** – erdvė, kurioje taško pozicijos nustatymui naudojamos trys reikšmės.

Atminties suskaldymas (angl. *Memory fragmentation*) - reiškinys, kuomet atmintis nėra valdoma efektyviai.

„Steam“ – žaidimų talpinimo platforma, skirta platinti ir parduoti žaidimus.

Šiukšlių rinkėjas (angl. *Garbage collector*) - automatinis atminties valdiklis.

Žaidimų variklis – programinė įranga su kuria yra kuriamas žaidimas.

Įvadas

Darbo naujumas ir aktualumas

„*System Requirements Lab*“ savo naudotojams pateikia formą, kurioje galima nurodyti savo kompiuterio parametrus ir norimą žaidimą, o sistema pateikia atsakymą ar atitinkamas kompiuteris bus pajėgus nurodytą žaidimą paleisti. Puslapyje taip pat pateikiamas 30-ies populiariausių žaidimų sąrašas, kuriame, prie kiekvieno žaidimo parodomas procentas, kiek žmonių gavo teigiamą atsakymą į šią užklausą. „*System Requirements Lab*“ duomenimis, vos nuo 77% iki 25% (vid. 48%) apklaustų žaidėjų turi kompiuterius, kurie atitinka minimalius populiariausių kompiuterinių žaidimų keliamus sistemos reikalavimus [1]. Todėl žaidimų optimizavimas yra aktualus žaidimų kūrėjams, kurie siekia pritraukti kuo didesnes auditorijas – kuo mažesni bus sistemai keliami reikalavimai, tuo didesnė tikimybė, kad žaidimu galės mėgautis daugiau žmonių. Galingėjant kompiuteriams žaidimų kūrėjai vis mažiau dėmesio skiria žaidimų optimizavimui, todėl žaidėjai su nepakankamai galingais kompiuteriais dažnai net negali žaisti naujai išleidžiamų žaidimų ir dėl to kūrėjai praranda nemenką dalį savo potencialios auditorijos. Tuo tarpu patys žaidėjai būna priversti įsigyti geresnę ir brangesnę įrangą norėdami galėti šiuos žaidimus žaisti. Buvo pasirinkta kurti 3D pirmo asmens šaudyklės žaidimą, nes šis žanras yra dominuojantis tarp populiariausių žaidimų ir tarp tų, kurių minimalūs sistemai keliami reikalavimai dažnai neatitinka žaidėjų lūkesčių. Žaidimo realizacijai pasirinktas „*Unity*“ žaidimų variklis.

Tikslas ir uždaviniai

Tikslas: Nustatyti naudojamų optimizacijos metodų ir technikų daromą įtaką sukurtam šaudyklės žanro žaidimo našumui.

Uždaviniai:

1. atlikti žaidimų našumo optimizacijos metodų, technikų ir geriausių praktikų analizę;
2. sukurti 3D pirmo asmens šaudyklės žanro kompiuterinį žaidimą, veikiantį bent 60-ies kadrų per sekundę ekrano atnaujinimo dažniu žaidžiant kompiuteriu, kuris turi bent 8 GB operatyviosios atminties, *NVIDIA GeForce GTX 1050* (arba geresnę) vaizdo plokštę ir turi *Intel Core i5-4430* (arba geresnį) procesorių;
3. atlikti sukurto žaidimo greitaveikos tyrimus ir įvertinti naudojamų optimizacijos metodų daromą įtaką žaidimo našumui.

Dokumento struktūra

Šis darbas sudarytas iš kelių skyrių: analitinė dalis, projektinė dalis, tyrimo dalis, eksperimentinė dalis ir išvados. Analitinėje dalyje apžvelgiami žaidimuose naudojami našumo optimizacijos metodai ir technikos bei gerosios žaidimų kūrimo praktikos kūrimui naudojant „*Unity*“ žaidimų variklį. Skyriaus pabaigoje apžvelgiami realizacijos įgyvendinimui pasirinkti sprendimai. Projektinėje dalyje apžvelgiama sukurto žaidimo architektūra, pateikiama panaudojimo atvejų diagrama, sistemos statinį ir dinaminį vaizdą apibūdinančios diagramos. Tyrimo dalyje apžvelgiami realizuoti sprendimai, nustatoma jų kokybė bei pateikiami sprendimų patobulinimai. Eksperimentinėje dalyje įvertinami sprendimų patobulinimai ir atliekamas eksperimentinis tyrimas. Išvadų skyriuje pateikiamos gautos išvados ir rezultatai apie optimizacijos metodų šaudyklės žanro žaidimo našumui daromą įtaką.

1. Analitinė dalis

Šiame skyriuje apžvelgiamos žaidimuose naudojami našumo optimizacijos metodai ir technikos, bei gerosios optimizacijos praktikos žaidimo kūrimui naudojant „*Unity*“ variklį. Kuriant žaidimą, bus bandoma pritaikyti įgytas žinias apie įvairius metodus ir optimizuoti žaidimą, kad šis veiktų bent 60-ies kadrų per sekundę ekrano atsinaujinimo dažniu ant kompiuterių su nustatytais parametrais.

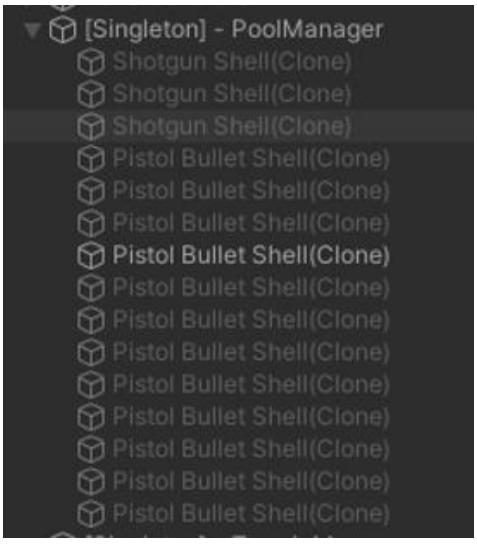
1.1. Žaidimuose naudojami našumo optimizacijos metodai ir technikos

Šiame poskyryje apžvelgiami žaidimuose ir jų kūrime naudojami našumo optimizacijos metodai ir technikos. Analizei pasirinkti objektų telkimo, nematomų objektų paslėpimo ir detalumo lygio optimizacijos metodai.

1.1.1. Objektų telkimas (angl. *Object Pooling*)

Objektų telkimas yra technika, kuri gali pagerinti žaidimo našumą sumažinant procesoriaus apdorojimo galią, naudojamą vykdant pasikartojančias objektų kūrimo ir naikinimo užklausas [2]. Šis našumo optimizacijos metodas žaidimo pradžioje inicijuoja pasirinktų objektų rinkinį, kuris būna paruoštas naudojimui, vietoje to, kad objektai būtų sukuriami ir sunaikinami pagal poreikį. Norint panaudoti pasirinktą objektą, šis paimamas iš sukurto rinkinio, jam priskiriamos norimos savybės (pvz, koordinatės arba pasisukimo kampas). Kai objektas tampa nereikalingu, jis yra paslėpiamas ir grąžinamas į rinkinį, o ne sunaikinamas [3 p. 1]. Taip tą patį objektą galima pakartotinai panaudoti kelis kartus skirtinguose scenarijuose neapkraunant variklio šiukšlių rinkimo valdiklio, kadangi kiekvieno objekto kūrimo metu procesorius turi išskirti reikalingą kiekį atminties, o objekto naikinimo metu tą išskirtą atmintį atlaisvinti [4]. Taigi, taip išvengiama ne tik galimo procesoriaus veikimo sutrikimų dėl šiukšlių rinkimo šuolių, bet ir galimo atminties suskaidymo [5].

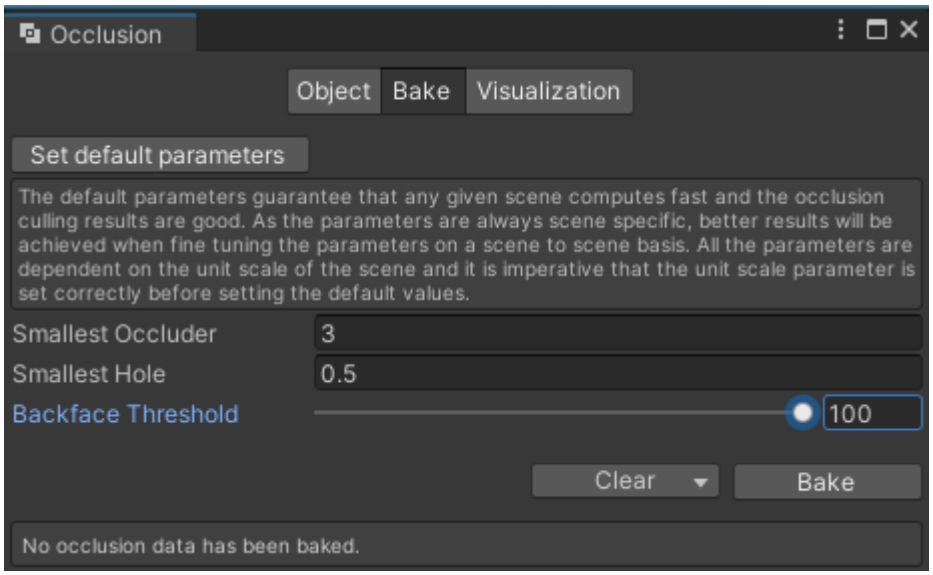
Šią techniką ypatingai naudinga pritaikyti tokiems objektams kaip kulkos, kulkų žymės, garso efektai, sprogimų efektai ir t.t., dėl to, kad juos dažnai reikia pakartotinai panaudoti žaidimo eigoje. Naudojant objektų telkimą, taip pat svarbu pasirinkti ir tinkamą pradinio rinkinio dydį – jis turi būti ne per didelis tam, kad nebūtų sukuriami objektai, kurie niekada nebus panaudojami. 1-ajame paveiksle pavaizduotas objektų telkimo sistemos pavyzdys „*Unity*“ redaktoriuje. Į objektų rinkinį įdėti kulkų tūtelių objektai.



1 pav. Objektų telkimo pavyzdys „*Unity*“ scenoje

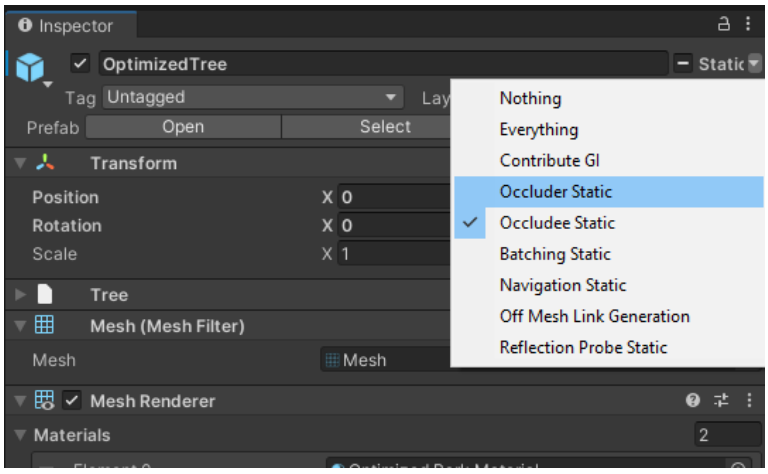
1.1.2. Nematomų objektų paslėpimas (angl. *Occlusion Culling*)

Nematomų objektų paslėpimo sistema paslepia ir išjungia (tačiau nesunaikina) žaidėjo kamerai nematomus daiktus, tam, kad jų nereikėtų grafiškai apdoroti ir grafikos plokštei bereikalingai naudoti papildomų resursų [6 p. 205]. Objektai žaidėjo kamerai tampa nematomi kai jie nepatenka į kameros akiratį arba jei daiktą pilnai užstoja kita struktūra. „*Unity*“ variklio nematomų objektų paslėpimo sistema sugeneruoja duomenis apie pasirinktą sceną, o vėliau tuos duomenis naudoja žaidimo veikimo metu, tam, kad nustatytų kokius objektus žaidėjo kamera turi atvaizduoti. Šis duomenų generavimo procesas vadinamas „kepimu“ [7]. 2-ajame paveiksle parodytas „*Unity Occlusion*“ sistemos duomenų generavimo langas. „Kepimo“ proceso trukmė priklauso nuo scenos sudėtingumo ir kompleksiškumo bei nuo pasirinktų nustatymų.



2 pav. „*Unity*“ variklio paslėpimo duomenų „kepimas“

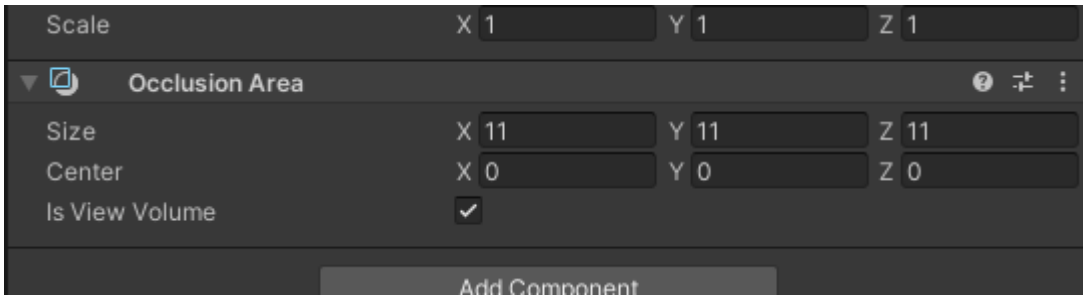
Tam, kad objektas būtų įtrauktas į generuojamus paslėpimo duomenis, scenoje jis turi būti pažymėtas atitinkama žyma. Dideli, nepermatomi ir nejudantys objektai, tokie kaip sienos, pastatai, reljefo detalės, būna žymimi kaip „*Occluder Static*“, o mažesni ir permatomi – žymimi kaip „*Occludee Static*“ [8]. 3-ajame paveiksle pavaizduotas objekto paruošimas paslėpimo sistemos duomenų generavimui.



3 pav. Objekto paruošimas paslėpimo sistemai

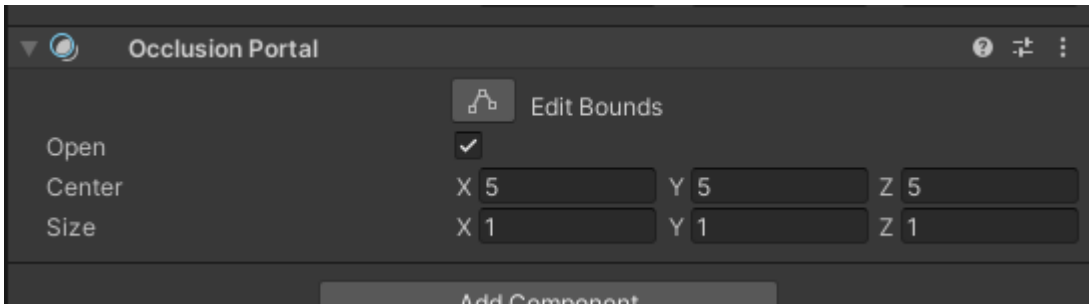
„*Unity*“ variklio objektų paslėpimo sistemą galima naudoti tik statiškiems objektams – t.y. objektams, kurie viso žaidimo metu nekeičia savo pozicijos, mastelio ar pasisukimo kampo bei yra pažymėti kaip *static* objektai. Objektai ne tik turi būti statiški, bet ir įdėti į žaidimo sceną tam, kad jiems galėtų būti sugeneruoti paslėpimo sistemos duomenys. Tai reiškia, kad variklyje įdiegto šio sprendimo įrankio negalima naudoti procedūriškai ir atsitiktinai generuojamiems lygiams, nes jų kūrimui reikia keisti objektų pozicijas arba objektus inicijuoti ir kurti programiškai. Dinamiški objektai negali būti paslėpiami, bet gali būti naudojami kaip užstojantys vaizdą kamerai [9].

Be statiškų objektų žymėjimų, paslėpimo duomenims generuoti, taip pat galima naudoti „*Occlusion Area*“ komponentą. Šiuo komponentu galima nustatyti erdves žaidimo scenoje, kuriose, žaidimo eigos metu, turėtų būti žaidėjo kamera. Šio komponento naudojimas gali pagerinti scenos geometrijos skaičiavimo tikslumą, sumažinti paslėpimo duomenų generavimo laiką ir sumažinti sunaudojamų duomenų kiekį [10]. 4-ajame paveiksle pavaizduotas „*Occlusion Area*“ komponento pavyzdys.



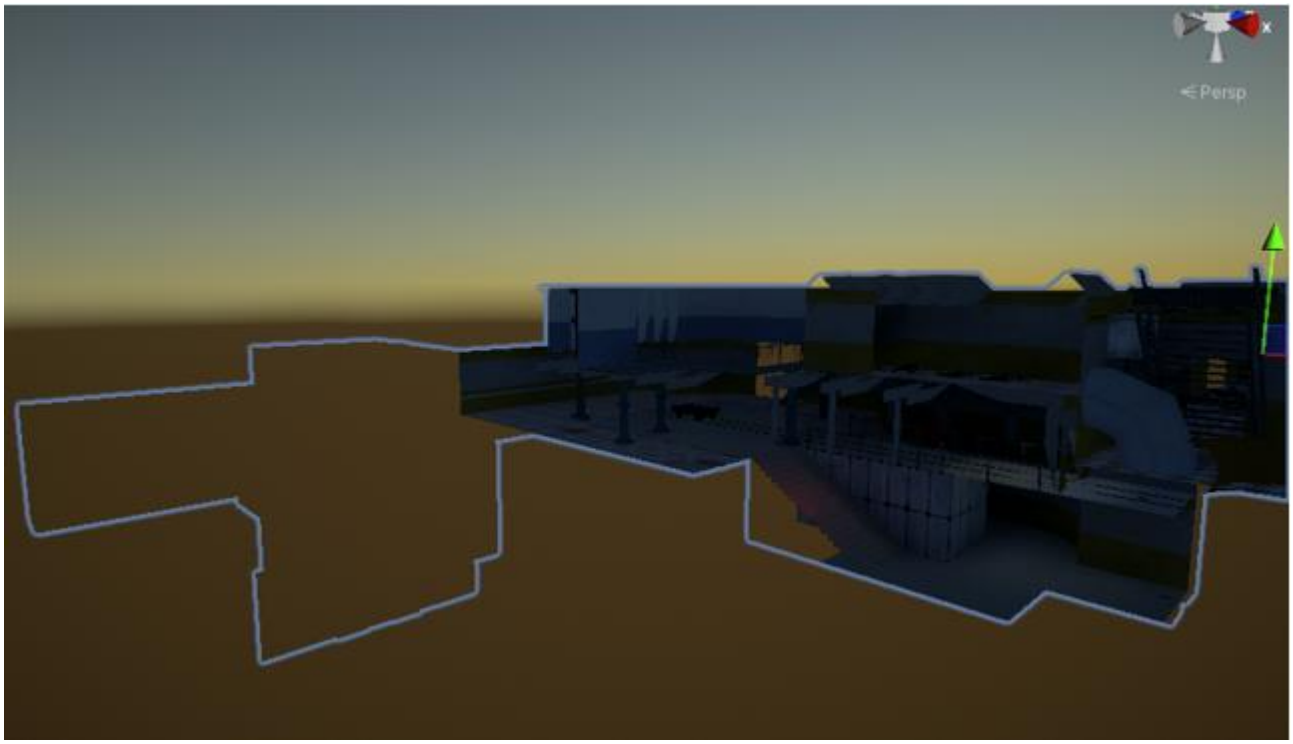
4 pav. „*Occlusion Area*“ komponento pavyzdys

„Unity“ variklyje yra dar vienas komponentas susijęs su nematomų objektų paslėpimo sistema – paslėpimo vartai (angl. „*Occlusion Portals*“). Šie vartai gali būti atidaromi arba uždaromi, priklausant nuo situacijos – kai vartai uždaryti, už jų esantys objektai yra paslėpiami, o kai vartai atidaryti, pastarieji objektų nepaslepia. Šis komponentas tinka objektams, kurie turi atsidarymo ir užsidarymo būsenas (pavyzdžiui, durys) [11]. 5-ajame paveiksle pavaizduotas „*Occlusion Portal*“ komponento pavyzdys.



5 pav. „*Occlusion Portal*“ komponento pavyzdys

6-ajame paveiksle pavaizduotas objektų paslėpimo sistemos naudojimo pavyzdys, kuomet žaidimo lygio kambario sienos, kurias užstoja kitos sienos esančios arčiau žaidėjo, nėra grafiškai apdorojamos, taip sutaupant GPU veikimo laiką.



6 pav. Objektų paslėpimo sistemos pavyzdys

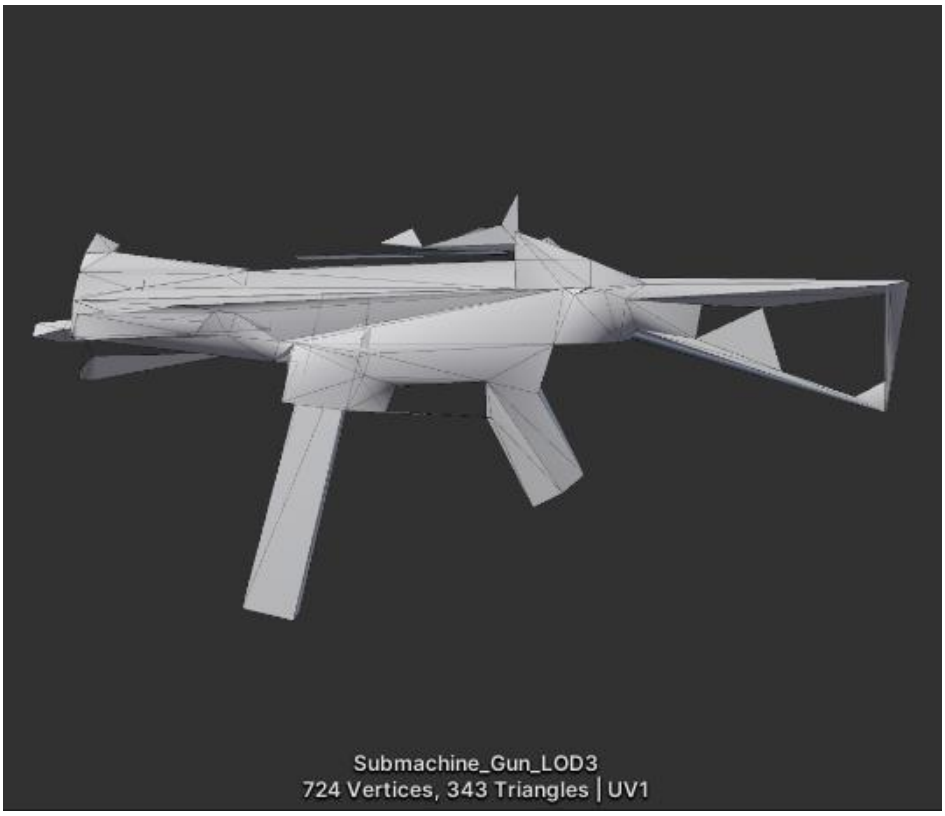
1.1.3. Detalumo lygis (angl. Level of Detail)

Detalumo lygis (toliau LOD) yra metodas, sumažinantis GPU operacijų skaičių, kurių reikia „*Unity*“ varikliui, kad būtų galima grafiškai apdoroti nutolusius objektus. LOD technika leidžia sumažinti 3D modelių trikampių skaičių taip supaprastinant modelius, priklausomai nuo atstumo iki žaidėjo kameros [12]. Pagal nutylėjimą, „*Unity*“ variklyje objektai naudoja tą patį detalumo lygį nepriklausomai nuo atstumo iki žaidėjo. Objektai, kuriems gali būti taikomas šis metodas, turi turėti kelias skirtingas versijas, kurios skiriasi savo detalumo lygiu. Kuo toliau objektas yra nuo žaidėjo kameros, tuo žemesnė detalumo lygio versija yra rodoma [12].

Ši technika gali sumažinti apdorojimo krūvį, nes mažesnio detalumo modelių versijos reikalauja mažiau apdorojimo galios, nei aukšto detalumo objektai [13]. 7-ajame paveiksle pateikiamas aukšto detalumo lygio ginklo modelio pavyzdys. Šis modelis yra sudarytas iš daugiau nei 7000 trikampių, yra apie 450 KB dydžio ir yra tinkamas naudoti kuomet yra arti žaidėjo, pavyzdžiui, kai žaidėjas ginklą laiko rankose.

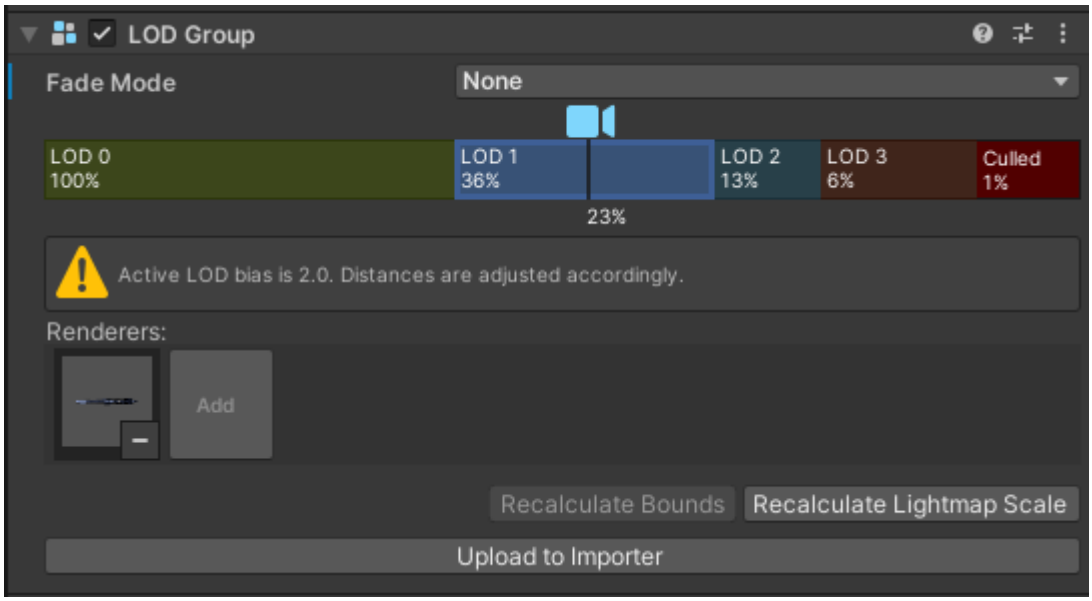
7 pav. Aukšto detalumo lygio modelis

Tuo tarpu 8-ajame paveiksle pateikiamas to paties ginklo modelio žemo detalumo lygio modelis. Šis modelis yra sudarytas iš 343 trikampių, yra maždaug 35 KB dydžio ir naudoja kelis kartus mažiau kompiuterio resursų norint jį grafiškai apdoroti lyginant su anksčiau minėtu modeliu. Šio modelio versiją tinkama naudoti kai ginklas yra toli nuo žaidėjo ir dėl esamo atstumo modelio detales būtų sunku įžiūrėti plika akimi.



8 pav. Žemo detalumo lygio modelis

Šios detalumo versijos valdomos naudojant „*Unity*“ „*LOD Group*“ komponentą, kuriame galima nustatyti kameros atstumus, kuriuos viršijus keičiasi detalumo lygis, nustatyti detalumo lygių kiekį bei nustatyti perėjimo tarp lygių nustatymus [14]. Šiuos nustatymus ir atstumus reikia kruopščiai kalibruoti, kad žaidėjas žaidimo eigoje lengvai nepastebėtų itin mažo detalumo modelių, bei nepastebėtų nesklandžių perėjimų iš vieno detalumo lygio į kitą, nes tai gali sutrikdyti žaidėjo įsitraukimą į žaidimo eigą. 9-ajame paveiksle pavaizduotas šio komponento naudojimo pavyzdys.

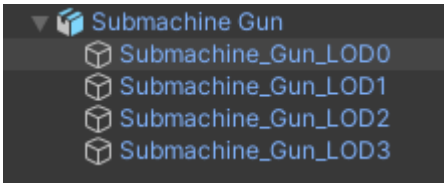


9 pav. „*Unity*“ „*LOD Group*“ komponentas

Verta paminėti, kad norint šią techniką naudoti, reikia turėti 3D modelius, kurie būtų sudaryti iš kelių detalumo lygių. Norint modeliams pritaikyti LOD sistemą, juos importuojant į žaidimo variklį, reikia detalumo lygius pavadinti specifiška tvarka [15]:

- *ModelioPavadinimas_LOD0* pirmam, aukščiausio detalumo, lygiui.
- *ModelioPavadinimas_LOD1*
- *ModelioPavadinimas_LOD2* ir t.t. priklausomai nuo lygių kiekio.

10-ajame paveiksle pavaizduota 3D modelio, turinčio detalumo lygio sistemos komponentą, objektų struktūros pavyzdys.



10 pav. Detalumo lygių objektų hierarchija

1.2. Gerosios optimizacijos praktikos žaidimo kūrimui naudojant „*Unity*“ variklį

Šiame poskyryje pateikiamos kelios gerosios optimizacijos praktikos, kurios bus naudojamos kuriant žaidimą. Šios praktikos ir veiksmai yra išsamiai aprašytos „*Unity*“ variklio kūrėjų elektroninėje knygoje „*Optimize your console and PC game performance*“ [16].

1.2.1. Programavimas variklio architektūros kontekste

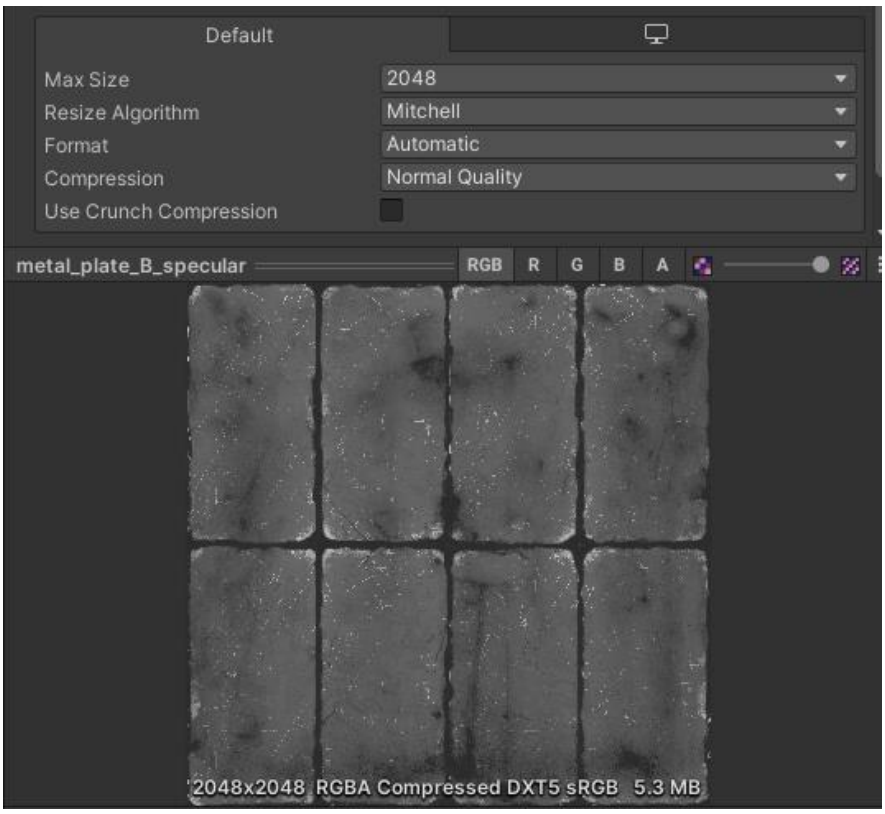
Programuojant žaidimą, svarbu suprasti kaip veikia variklio architektūra. „*Unity*“ žaidimas yra sudarytas iš daugybės žaidimų objektų (angl. *GameObjects*), iš kurių kiekvienas turi sau priskirtus komponentus ar programinio kodo failus. Tie kodo failai, kuriuos galima priskirti objektams, paveldi klasę *MonoBehaviour*, kuri ir suteikia objektams pagrindines variklio veikimo funkcijas. Pavyzdžiui, yra du pagrindiniai šios klasės metodai, kurie pradeda komponento logikos veikimą – prabusti (angl. *Awake*) ir pradėti (angl. *Start*). Kiti du svarbūs metodai – atnaujinimas (angl. *Update*) ir vėlus atnaujinimas (angl. *LateUpdate*). Šie metodai yra vykdomi kiekvieną žaidimo kadrą [17]. Turint tai omenyje, reikia stengtis dėti kuo mažiau kodu aprašytos logikos į šiuos metodus, jeigu to nereikalauja žaidimo eiga. Pavyzdžiui, nebūtina kiekvieno kadro metu atnaujinti teksto, kuris atvaizduoja kiek gyvybių turi žaidėjas. Tekstą galima atnaujinti tik tada, kai pasikeičia žaidėjo gyvybių kiekis, taip išvengiant dažnų teksto atvaizdavimo užklausų. Net ir tuščios *Update* funkcijos reikalauja papildomų resursų, todėl nenaudojamus *MonoBehaviour* klasės metodus geriau yra ištrinti [18].

1.2.2. Apšvietimas

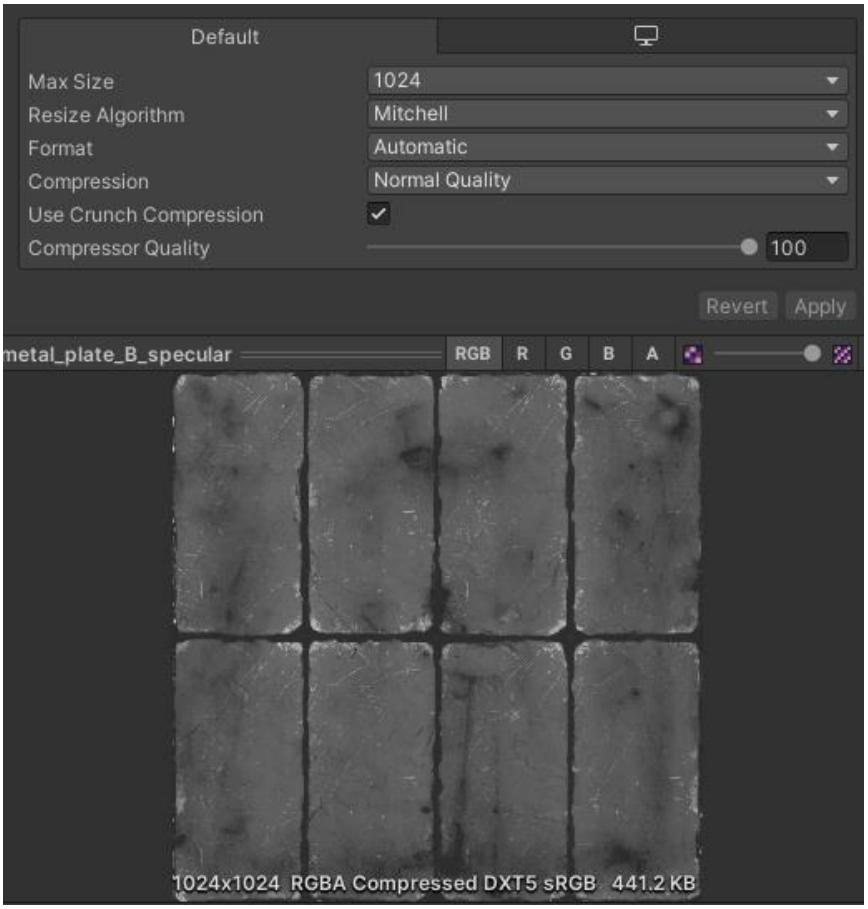
Kuriant žaidimo lygius, reikėtų stengtis naudoti kuo mažiau dinamiškai kintančių šviesų ir šešėlių, nes realaus laiko apšvietimas gali būti brangus procesas. Scenos apšvietimo duomenis rekomenduotina sugeneruoti iš anksto siekiant išvengti našumo kritimo žaidimo eigos metu. Taip pat, rekomenduojama išjungti objektų šėšėlius siekiant sumažinti ekrano piešimo užklausų kiekį [19].

1.2.3. Kūrimui naudojamų failų (angl. *Assets*) optimizavimas ir importavimas

Bene svarbiausia failų importavimo dalis – žaidimo tekstūrų importavimas. Didžioji dalis žaidimo naudojamos atminties sunaudoja būtent tekstūros, todėl yra labai svarbu joms parinkti tinkamus nustatymus. Rekomenduotina naudoti kiek įmanoma mažesnės raiškos tekstūras, kurių kraštinių ilgis būtų lygus skaičiaus 2 pakėlimo tam tikru laipsniu rezultatui (pvz., 128, 256, 512 ir t.t.). Taip pat, tekstūroms reikėtų išjungti „*Read/Write Enabled*“ nustatymą, nes priešingu atveju tekstūros kopija bus sukurta tiek CPU, tiek GPU atmintyje [16 p. 31]. Importavus tekstūras, joms taip pat reikėtų dar uždėti suspaudimo (angl. *compression*) nustatymus – tai dar labiau sumažina tekstūros užimamą dydį [16 p. 30]. Be to, dažnai kartu rodomas tekstūras ir piešiamus paveiksliukus, galima supakuoti į bendrus tekstūrų atlasus naudojant „*Unity Sprite Atlas“* įrankį. Tekstūrų atlasų naudojimas gali sumažinti piešimo užklausų kiekį [19]. 11-ajame ir 12-ajame paveiksluose galima pamatyti prastai optimizuotos ir geriau optimizuotos tekstūrų palyginimą. Nors vizualiai abi tekstūros atrodo beveik vienodai, 11-ajame paveiksle pavaizduota tekstūra yra 2048x2048 pikselių dydžio ir užima 5,3 MB atminties, o 12-ajame paveiksle pavaizduota suspausta, 1024x1024 pikselių dydžio tekstūra užima apie 10 kartų mažiau atminties – 441,2 KB. Taigi, tekstūrų suspaudimas ir paruošimas naudojimui yra labai svarbus žaidimo optimizavimo aspektas, nes tai gali sutaupyti nemažai brangios naudojimui skirtos atminties.

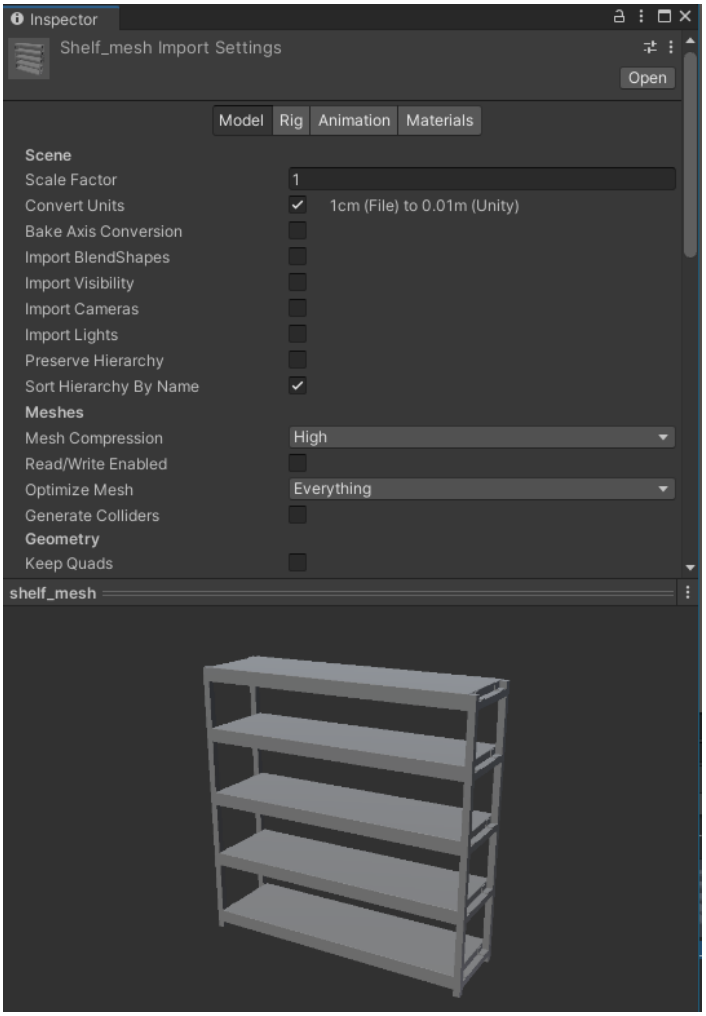


11 pav. Prastai optimizuota (2048x2048 dydžio, nesukompresuota) tekstūra



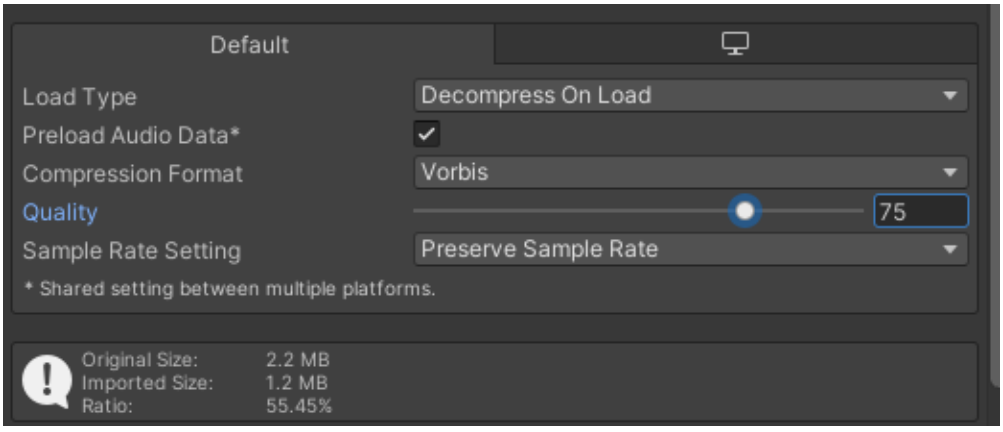
12 pav. Geriau suoptimizuota (1024x1024 dydžio, sukompresuota) tekstūra

Panašiai kaip ir su tekstūromis, svarbu parinkti gerus importavimo parametrus įkėlinėjant ir žaidimo 3D modelius. Importuojant modelius, svarbiausia yra pažymėti modelių suspaudimo (angl. *Compress the mesh*) ir skaitymo / rašymo išjungimo (angl. *Disable read/write*) nustatymus [16 p. 34]. 13-ajame paveiksle rodomas modelio importavimo nustatymų pavyzdys.



13 pav. Modelio importavimo nustatymų pavyzdys

Garsus ir audio failus taip pat galima optimizuoti, norint išsaugoti šiek tiek naudojamos atminties [20]. Garso failams galima sumažinti jų kokybės nustatymą, kaip parodyta 14-ajame paveiksle.



14 pav. Garso efekto kokybės mažinimas

1.2.4. Grafinės sąsajos struktūra

Kuriant grafinę sąsają, svarbu, kad jos hierarchija būtų sudaryta sluoksniais, aktyvuojant tik reikalingus objektus atvaizdavimui taip mažinant persidengiančių objektų kiekį. Nereikalingoms sąsajos „drobėms“ kartu turėtų būti išjungiamas ir pats „*Canvas*“ komponentas siekiant sumažinti GPU piešimo kreipinių (angl. *Draw calls*) kiekį [21].

1.3. Apibendrinimas

Kuriant šaudyklės žanro žaidimą, patartina įgyvendinti objektų telkimo, nematomų objektų paslėpimo ir detalumo lygio metodus, nes pastebėta, kad šių metodų naudojimas gali padėti sumažinti operatyviosios atminties naudojimą bei pagerinti procesoriaus ir grafinės plokštės resursų našumą. Kuo procesoriaus ir grafinės plokštės našumas geresnis, tuo žaidimas yra geriau optimizuotas ir gali sugeneruoti daugiau kadrų per sekundę. Kita vertus, jeigu šie metodai netinkamai implementuoti, žaidimo našumas gali net pablogėti.

Žaidimas turi veikti bent 60-ies kadrų per sekundę vaizdo atnaujinimo dažniu kompiuteryje turinčiu tokią aparatinę įrangą: bent 8 GB operatyviosios atminties (RAM), *NVIDIA GeForce GTX 1050* (arba geresne) vaizdo plokštė ir *Intel Core i5-4430* (arba geresnis) procesorius. Šie parametrai parinkti atlikus „*Steam*“ platformos naudotojų įrangos apklausos analizę ir nustačius, kad tokie komponentai atspindi vidutinio „Steam“ naudotojo turimą įrangą [22].

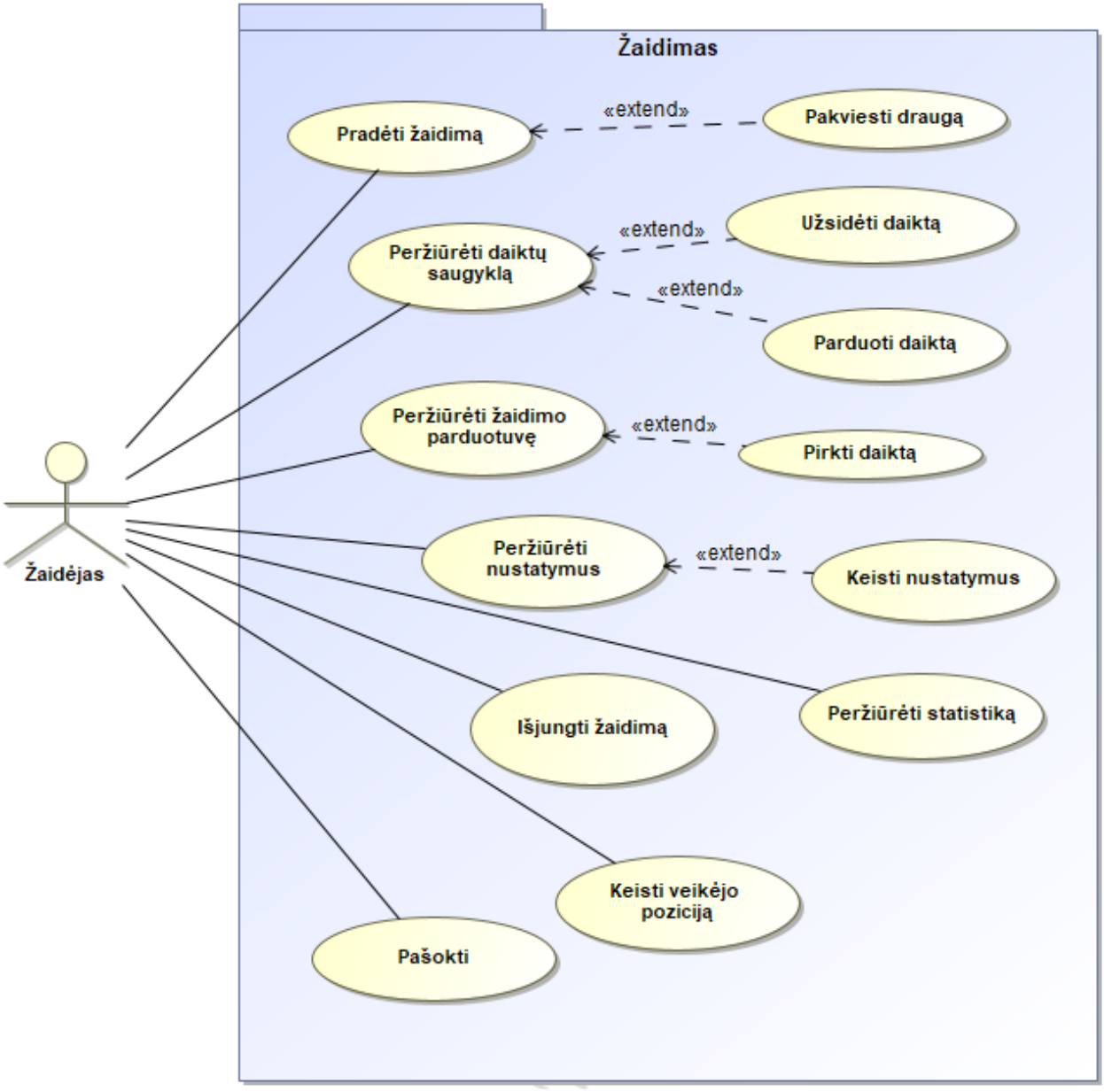
Pasirinkta kurti 3D pirmo asmens šaudyklės žaidimą, nes šis žanras yra dominuojantis tarp populiariausių žaidimų, kurių minimalūs sistemai keliami reikalavimai dažnai neatitinka žaidėjų lūkesčių. Taip pat, tie žaidimai dažniausiai yra žaidžiami tinkle (angl. *Online*), todėl pasirinkta kurti kooperacinio tipo žaidimą, kuris galėtų veiktų kelių žaidėjų režime. Tai taip pat prideda ir papildomą sluoksnį optimizacijai dėl galimos delsos ir objektų sinchronizacijos. Žaidimo grafinė sąsaja taip pat gali būti prastai optimizuota ir kelti našumo problemų. Norint, kad grafinė sąsaja būtų labiau apkrauta ir turėtų sudėtingesnę hierarchiją, buvo nuspręsta implementuoti daiktų parduotuvę ir inventoriaus sistemą meniu scenoje, tam, kad būtų galima atlikti su grafinės sąsajos optimizacija susijusius tyrimus.

2. Projektinė dalis

Šio darbo metu buvo realizuotas „*Steam*“ platformai skirtas 3D pirmo asmens šaudyklės žanro kooperatyvus tinklo kompiuterinis žaidimas pavadinimu „*Toxic Wasteland*“. Žaidimą galima žaisti kartu su draugais – yra palaikomas kelių žaidėjų režimas (angl. *Multiplayer*). Visi kartu žaidžiantys žaidėjai turi bendrą tikslą – įvykdyti misiją, kuri yra nustatoma pagal pasirinktą žaidimo tipą. Žaidimas buvo suprojektuotas ir įgyvendintas dirbant grupėje kartu su kolega Luku Kalade, užduotys ir realizacijos darbai buvo paskirstyti ir atlikti pagal žaidimo posistemes. Šiame skyriuje aptariami mano paties projektuoti sistemos aspektai.

2.1. Panaudojimo atvejai

15-ajame paveiksle pavaizduoti sistemos mano realizuotų panaudos atvejų diagrama.



15 pav. Panaudos atvejų diagrama

Žaidimas yra pradedamas nuo meniu scenos, kurioje žaidėjas gali pasiruošti savo įrangą, keisti nustatymus ir pradėti misiją. Pradėjus žaidimą iš pradinės scenos, žaidėjas yra perkeliamas į žaidimo sceną, kurioje vykdoma misija. Realizuoti panaudojimo atvejai detaliau aprašyti nuo 1-os iki 13-os lentelėse imtinai:

1 lentelė. Panaudojimo atvejo „Pradėti žaidimą“ aprašymas

PA 1 Pradėti žaidimą	
Tikslas/uždavinys. Pereiti iš meniu scenos į žaidimo sceną ir pradėti žaisti žaidimą.	
Aprašymas. Vykdant šį PA, pereinama iš pradinio meniu scenos į žaidimo sceną kur galima kviesti draugus ir keisti lygio konfigūraciją.	
Prieš-sąlyga	Naudotojas yra pagrindiniame meniu.
Aktorius	Žaidėjas.
Sužadinimo sąlyga	Naudotojas paspaudžia mygtuką „Host Game“.
Pagrindinis scenarijus	
Naudotojo veiksmai	**Sistemos reakcija**
1. Inicijuojamas žaidimo pradėjimas.	1.1 Sistema pakeičia sceną iš pradinio meniu scenos į žaidimo sceną ir žaidėjui leidžiama pasirinkti lygio konfigūraciją ir / arba pakviesti draugą.
2. Baigiamas PA.	
Po-sąlyga	Naudotojas yra nukeliamas į žaidimo sceną.

2 lentelė. Panaudojimo atvejo „Pakviesti draugą“ aprašymas

PA 2 Pakviesti draugą	
Tikslas/uždavinys. Žaidėjui pakviesti draugą su kuriuo galėtų žaisti žaidimą.	
Aprašymas. Vykdant šį PA, atidaromas „Steam“ langas kuriame jis gali pakviesti savo „Steam“ platformos draugus į serverio kambarį.	
Prieš-sąlyga	Naudotojas yra paspaudęs „Host Game“ mygtuką pradiniame meniu.
Aktorius	Žaidėjas.
Sužadinimo sąlyga	Naudotojas paspaudžia mygtuką „Invite“.
Pagrindinis scenarijus	
Naudotojo veiksmai	**Sistemos veiksmai**
1. Inicijuojamas draugo pakvietimas.	1.1 Žaidimas atveria „Steam Overlay“ su visų žaidėjo draugų sąrašu.
2. Baigiamas PA.	
Po-sąlyga	Naudotojui pateiktas draugų sąrašas iš jo „Steam“ draugų.

3 lentelė. Panaudojimo atvejo „Peržiūrėti žaidimo parduotuvę“ aprašymas

PA 3 Peržiūrėti žaidimo parduotuvę	
Tikslas/uždavinys. Atidaryti žaidimo parduotuvės meniu.	
Aprašymas. Vykdant šį PA, galima atsidaryti žaidimo parduotuvės meniu ir peržiūrėti galimus pirkti daiktus.	
Prieš-sąlyga	Naudotojas yra pradiniame meniu.
Aktorius	Žaidėjas
Sužadinimo sąlyga	Naudotojas paspaudžia „Shop“ mygtuką.
Pagrindinis scenarijus	

Naudotojo veiksmai	**Sistemos veiksmai**
1. Inicijuojamas parduotuvės atidarymas.	1.1 Sistema atidaro žaidimo parduotuvės meniu ir pateikia galimus pirkti daiktus.
2. Baigiamas PA.	
Po-sąlyga	Naudotojas yra žaidimo parduotuvės meniu.

4 lentelė. Panaudojimo atvejo „Pirkti daiktą“ aprašymas

PA 4 Pirkti daiktą	
Tikslas/uždavinys. Žaidėjui nusipirkti daiktą iš žaidimo parduotuvės.	
Aprašymas. Šis panaudojimo atvejis skirtas nupirkti daiktui iš žaidimo parduotuvės ir įdėti jį į žaidėjo saugyklą.	
Prieš-sąlyga	Naudotojas yra paspaudęs mygtuką „Shop“ ir yra žaidimo parduotuvės meniu.
Aktorius	Žaidėjas.
Sužadinimo sąlyga	Naudotojas pasirenka daiktą ir paspaudžia ant jo kainos.
Pagrindinis scenarijus	
Naudotojo veiksmai	**Sistemos veiksmai**
1. Inicijuojamas daikto pirkimas.	1.1. Tikrinama, ar žaidėjas turi pakankamai pinigų ir parodomas patvirtinimo pranešimas.
2. Priimamas patvirtinimas.	2.1. Daiktas yra nuperkamas, atimami pinigai, daiktas įdedamas į saugyklą, uždaromas patvirtinimo pranešimas.
3. Baigiamas PA.	
Po-sąlyga	Atidaromas parduotuvės meniu.
Alternatyvūs scenarijai	
1a. Jei žaidėjas neturi pakankamai pinigų.	1.1a. Daiktas nenuperkamas ir patvirtinimo pranešimas nėra parodomas.
2a. Jei žaidėjas patvirtinimą atmeta.	2.1a. Daiktas nenuperkamas ir patvirtinimo pranešimas yra uždaromas.

5 lentelė. Panaudojimo atvejo „Peržiūrėti daiktų saugyklą“ aprašymas

PA 5 Peržiūrėti daiktų saugyklą	
Tikslas/uždavinys. Atidaryti daiktų saugyklos meniu	
Aprašymas. Vykdant šį PA, žaidėjui yra atidaromas daiktų saugyklos meniu ir galima peržiūrėti turimus daiktus.	
Prieš-sąlyga	Naudotojas yra pagrindiniame meniu.
Aktorius	Žaidėjas.
Sužadinimo sąlyga	Naudotojas paspaudžia mygtuką „Inventory“.
Pagrindinis scenarijus	
Naudotojo veiksmai	**Sistemos veiksmai**
1. Inicijuojamas daiktų saugyklos atidarymas.	1.1. Atidaromas daiktų saugyklos meniu ir išdėstomas visų žaidėjo turimų daiktų sąrašas.
2. Baigiamas PA.	
Po-sąlyga	Atidarytas daiktų saugyklos meniu.

6 lentelė. Panaudojimo atvejo „Užsidėti daiktą“ aprašymas

PA 6 Užsidėti daiktą	
Tikslas/uždavinys. Leisti žaidėjui užsidėti turimą ginklą ar įrankį.	
Aprašymas. Vykdant šį PA, žaidėjas gali užsidėti pasirinktą daiktą iš savo daiktų saugyklos.	
Prieš-sąlyga	Naudotojas yra daiktų saugyklos meniu lange.
Aktorius	Žaidėjas.
Sužadinimo sąlyga	Naudotojas paspaudžia ant vieno iš galimų žaidėjo daiktų laukelių ir iš turimų daiktų sąrašo pasirenka norimą daiktą spausdamas šalia jo esantį mygtuką „Equip“.
Pagrindinis scenarijus	
Naudotojo veiksmai	**Sistemos veiksmai**
1. Inicijuojamas daikto užsidėjimo procesas.	1.1. Sistema apginkluoja žaidėją pasirinktu daiktu.
2. Baigiamas PA.	
Po-sąlyga	Daiktas yra uždedamas, parodomas saugyklos meniu.

7 lentelė. Panaudojimo atvejo „Parduoti daiktą“ aprašymas

PA 7 Parduoti daiktą	
Tikslas/uždavinys. Parduoti turimą daiktą	
Aprašymas. Vykdant šį PA, žaidėjo pasirinktas daiktas yra parduodamas už 10% jo įsigyjimo kainos.	
Prieš-sąlyga	Naudotojas yra daiktų saugyklos meniu lange.
Aktorius	Žaidėjas.
Sužadinimo sąlyga	Naudotojas iš turimų daiktų sąrašo pasirenka daiktą, kurį nori parduoti spausdamas šalia jo esantį mygtuką „Sell“.
Pagrindinis scenarijus	
Naudotojo veiksmai	**Sistemos veiksmai**
1. Inicijuojamas daikto pardavimas.	1.1. Parodomas patvirtinimo pranešimas.
2. Priimamas patvirtinimas.	2.1. Daiktas yra parduodamas, pridedami pinigai, daiktas pašalinamas iš saugyklą, uždaromas patvirtinimo pranešimas.
3. Baigiamas PA.	
Po-sąlyga	Parodomas saugyklos meniu, patvirtinimo pranešimas yra uždaromas.
Alternatyvūs scenarijai	
2a. Jei žaidėjas patvirtinimą atmeta.	2.1a. Daiktas neparduodamas ir patvirtinimo pranešimas yra uždaromas.

8 lentelė. Panaudojimo atvejo „Peržiūrėti statistiką“ aprašymas

PA 8 Peržiūrėti statistiką	
Tikslas/uždavinys. Peržiūrėti žaidėjo statistiką (pvz, žaidėjo lygį, nužudytų priešų kiekį, pereitų lygių kiekį ir t.t.)	
Aprašymas. Vykdant šį PA, žaidėjui parodoma jo statistika.	
Prieš-sąlyga	Naudotojas yra pagrindiniame meniu.
Aktorius	Žaidėjas.

Sužadinimo sąlyga	Naudotojas paspaudžia ant mygtuko „Profile & Stats”.
Pagrindinis scenarijus	
Naudotojo veiksmai	**Sistemos veiksmai**
1. Inicijuojamas statistikos lango atidarymas.	1.1. Sistema atidaro statistikos meniu ir parodo žaidėjo statistiką.
2. Baigiamas PA.	
Po-sąlyga	Atidaromas statistikos meniu.

9 lentelė. Panaudojimo atvejo „Peržiūrėti nustatymus“ aprašymas

PA 9 Peržiūrėti nustatymus	
Tikslas/uždavinys. Atidaryti nustatymų meniu.	
Aprašymas. Vykdant šį PA, žaidėjui yra atidaromas nustatymų meniu ir galima peržiūrėti nustatymus.	
Prieš-sąlyga	Naudotojas yra pagrindiniame meniu.
Aktorius	Žaidėjas.
Sužadinimo sąlyga	Naudotojas paspaudžia mygtuką „Settings“.
Pagrindinis scenarijus	
Naudotojo veiksmai	**Sistemos veiksmai**
1. Inicijuojamas nustatymų atidarymas.	1.1. Atidaromas nustatymų meniu.
2. Baigiamas PA.	
Po-sąlyga	Atidarytas nustatymų meniu.

10 lentelė. Panaudojimo atvejo „Keisti nustatymus“ aprašymas

PA 10 Keisti nustatymus	
Tikslas/uždavinys. Keisti įvairius video ar audio nustatymus žaidime.	
Aprašymas. Vykdant šį PA, žaidėjas gali keisti video ar audio nustatymus žaidime.	
Prieš-sąlyga	Naudotojas yra nustatymų meniu.
Aktorius	Žaidėjas
Sužadinimo sąlyga	Naudotojas paspaudžia vieną iš nustatymų mygtukų.
Pagrindinis scenarijus	
Naudotojo veiksmai	**Sistemos veiksmai**
1. Inicijuojamas nustatymų pakeitimas.	1.1. Sistema pakeičia nustatymą, išsaugo nustatymo pakeitimą.
2. Baigiamas PA.	
Po-sąlyga	Pakeičiamas pasirinktas nustatymas.

11 lentelė. Panaudojimo atvejo „Išjungti žaidimą“ aprašymas

PA 11 Išjungti žaidimą
Tikslas/uždavinys. Išjungti žaidimo langą.
Aprašymas. Vykdant šį PA, žaidėjas gali išjungti žaidimą.

Prieš-sąlyga	Naudotojas yra pagrindiniame meniu.
Aktorius	Žaidėjas.
Sužadinimo sąlyga	Naudotojas paspaudžia mygtuką „Quit“.
Pagrindinis scenarijus	
Naudotojo veiksmai	**Sistemos veiksmai**
1. Inicijuojamas žaidimo išjungimas.	1.1. Sistema išsaugo progresą ir išjungia žaidimo programą.
2. Baigiamas PA.	
Po-sąlyga	Žaidimas yra išjungtas.

12 lentelė. Panaudojimo atvejo „Keisti veikėjo poziciją“ aprašymas

PA 12 Keisti veikėjo poziciją	
Tikslas/uždavinys. Pakeisti žaidimo veikėjo poziciją žaidime.	
Aprašymas. Vykdant šį PA, žaidėjas gali keisti veikėjo poziciją žaidime naudodamas klavišus arba sukiodamas pelę.	
Prieš-sąlyga	Naudotojas yra žaidimo scenoje.
Aktorius	Žaidėjas.
Sužadinimo sąlyga	Naudotojas paspaudžia vieną iš kontrolės klavišų arba pasuka pelę.
Pagrindinis scenarijus	
Naudotojo veiksmai	**Sistemos veiksmai**
1. Paspaudžiamas vienas iš kontrolės klavišų arba pasukama pelė.	1.1. Veikėjo pozicija žaidime pasikeičia priklausomai nuo įvesties, veikėjas įgauna judėjimo animaciją.
2. Baigiamas PA.	
Po-sąlyga	Nauja veikėjo pozicija.

13 lentelė. Panaudojimo atvejo „Keisti veikėjo poziciją“ aprašymas

PA 13 Pašokti	
Tikslas/uždavinys. Padaryti, kad veikėjas pašoktų.	
Aprašymas. Vykdant šį PA, žaidėjas gali padaryti, kad valdomas veikėjas pašoktų pasinaudojant tam skirtu klavišu.	
Prieš-sąlyga	Naudotojas yra žaidimo scenoje, veikėjas yra ant žemės.
Aktorius	Žaidėjas.
Sužadinimo sąlyga	Naudotojas paspaudžia pašokimo mygtuką.
Pagrindinis scenarijus	
Naudotojo veiksmai	**Sistemos veiksmai**
1. Paspaudžiamas pašokimo mygtukas.	1.1. Veikėjas pašoka į viršų, suteikiama pašokimo animacija.
2. Baigiamas PA.	
Po-sąlyga	Veikėjas pašoka į viršų.

2.2. Nefunkciniai reikalavimai

Sistemai buvo iškelti šie nefunkciniai reikalavimai:

2.2.1. Stiliaus reikalavimai

Sistemos grafinės sąsajos stilius turi būti tamsių atspalvių todėl, kad šis stilius atitinka žaidimo tematiką bei nevargina akių. Visi žaidime matomi grafinės sąsajos elementai yra tamsiame arba juodo atspalvio fone.

2.2.2. Personalizavimo ir kalbos konfigūravimo reikalavimai

Sistema turi būti pateikiama anglų kalba todėl, kad sistema skirta angliškai kalbančiai auditorijai, norint pasiekti kuo didesnį žaidėjų skaičių. Visi žaidimo tekstai, kuriuos mato žaidėjas yra parašyti anglų kalba.

2.2.3. Reikalavimai užduočių vykdymo greičiui

Sistema turi veikti bent 60-ies kadrų per sekundę ekrano atnaujinimo dažniu kompiuteriuose, turinčiuose bent 8 GB operatyviosios atminties (RAM), *NVIDIA GeForce GTX 1050* (arba geresnę) vaizdo plokštę ir *Intel Core i5-4430* (arba geresnį) procesorių, todėl, kad 60 yra minimalus kadrų kiekis reikalingas sklandžiai žaidimo eigai vykdyti, o įvardinti parametrai atitinka vidutinio „*Steam*“ platformos naudotojo turimo kompiuterio galingumą.

2.2.4. Reikalavimai tikslumui

Norint pirkti daiktus žaidimo parduotuvėje, jų kainos turi būti nurodytos sveiku skaičiumi todėl, kad žaidime numatyta valiuta neatspindi jokios realios valiutos ir skaičiavimo paprastumui užtenka naudoti sveikąsias skaičiaus dalis.

2.2.5. Reikalavimai išplečiamumui

Norint, kad sistema būtų paruošta jos išplečiamumui ir papildymui, esminiai turinio elementai turi būti realizuoti naudojant *Unity* „*Scriptable Object*“ klases. Norint žaidimą tobulinti jau po jo išleidimo, turėtų būti sąlyginai nesunku pridėti naują ginklą ar priešo tipą. Taip pat, atsiradus poreikiui subalansuoti žaidimo eigą, minėtoji klasė suteikia galimybę nesunkiai pakeisti ginklų ar priešų charakteristikas. Žaidimo lygių, priešų ir ginklų konfigūracijos implementuotos panaudojant *Unity* „*Scriptable Object*“ klases.

2.2.6. Reikalavimai darbui su gretimomis sistemomis

Sistema turi veikti kompiuteriuose su „*Windows 10*“ operacine sistema todėl, kad tai yra populiariausia operacinė sistema tarp „*Steam*“ platformos naudotojų.

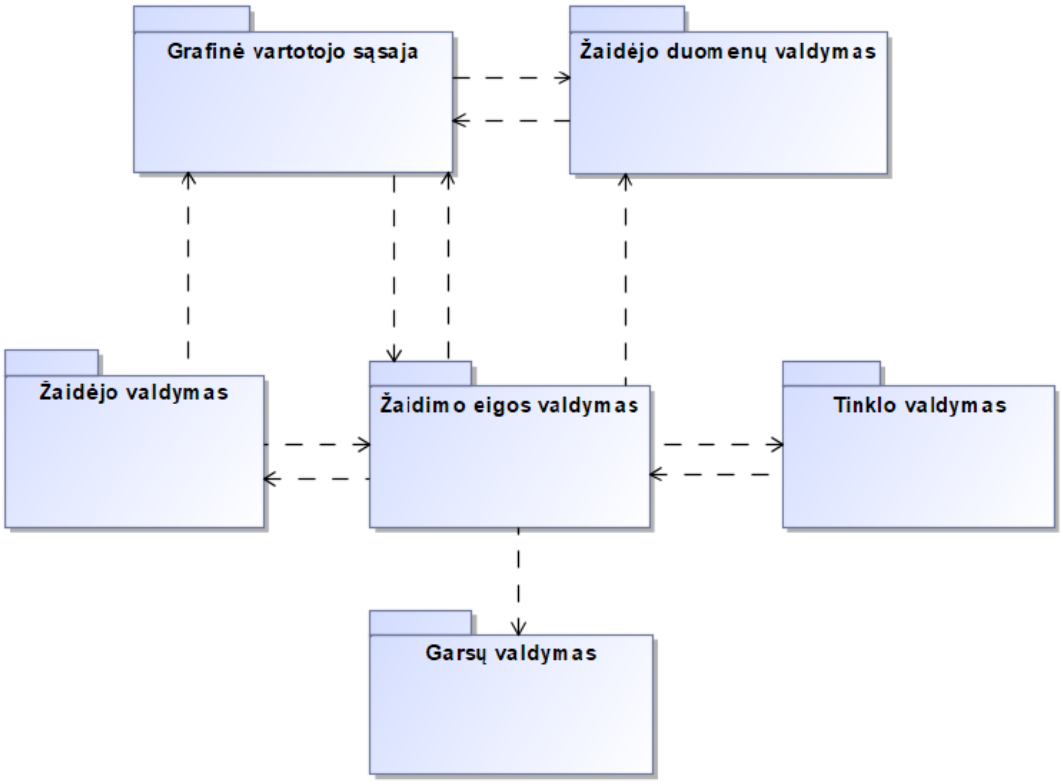
2.2.7. Kultūriniai rinkos reikalavimai

Sistema privalo neturėti jokių antisemitinių, rasistinių, homofobiškų ar kitaip diskriminuojančių ar įžeidžiančių simbolių ar tekstų.

2.3. Sistemos statinis vaizdas

2.3.1. Apžvalga

Visas žaidimas sudarytas iš šešių paketų, iš kurių aš įgyvendinau šiuos tris: grafinė vartotojo sąsaja, žaidėjo duomenų valdymas ir tinklo valdymas. 16-ajame paveiksle pavaizduota sistemos paketų diagrama.

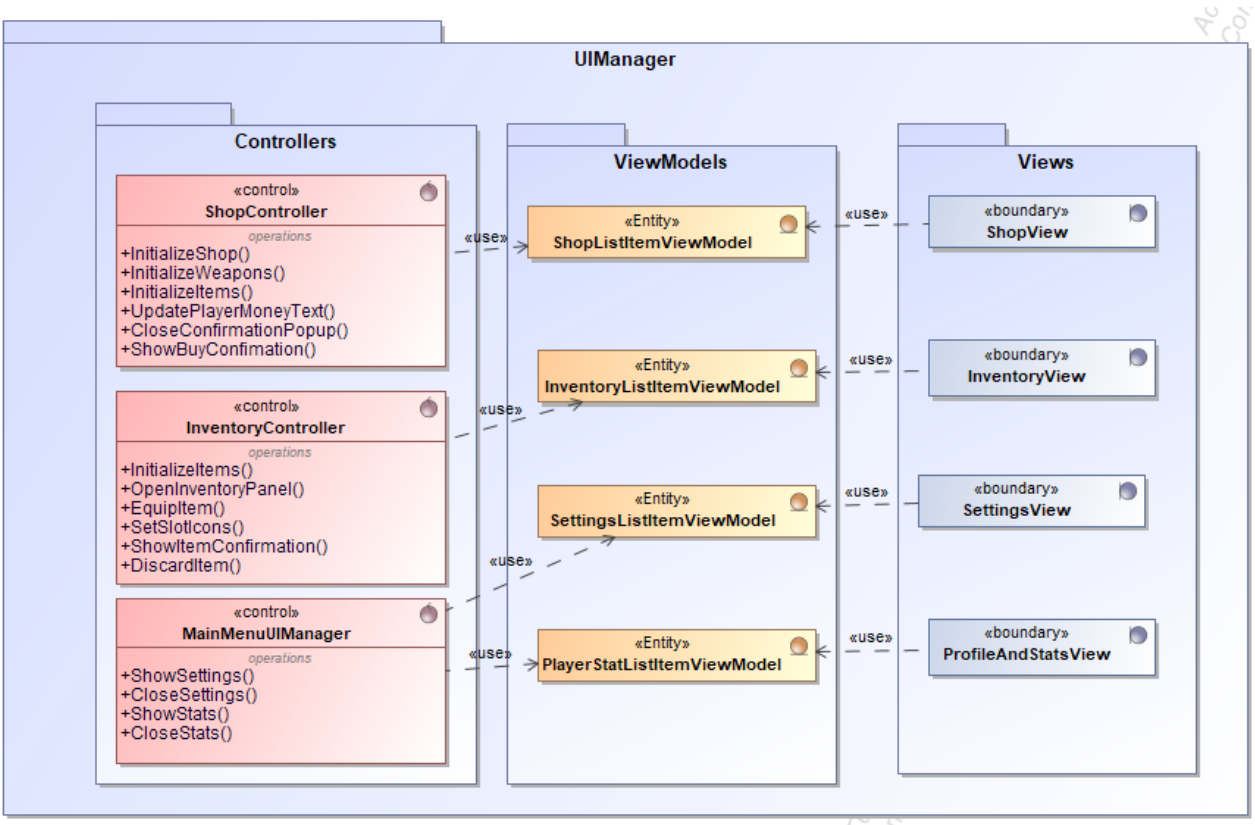


16 pav. Sistemos paketų diagrama

2.3.2. Paketų detalizavimas

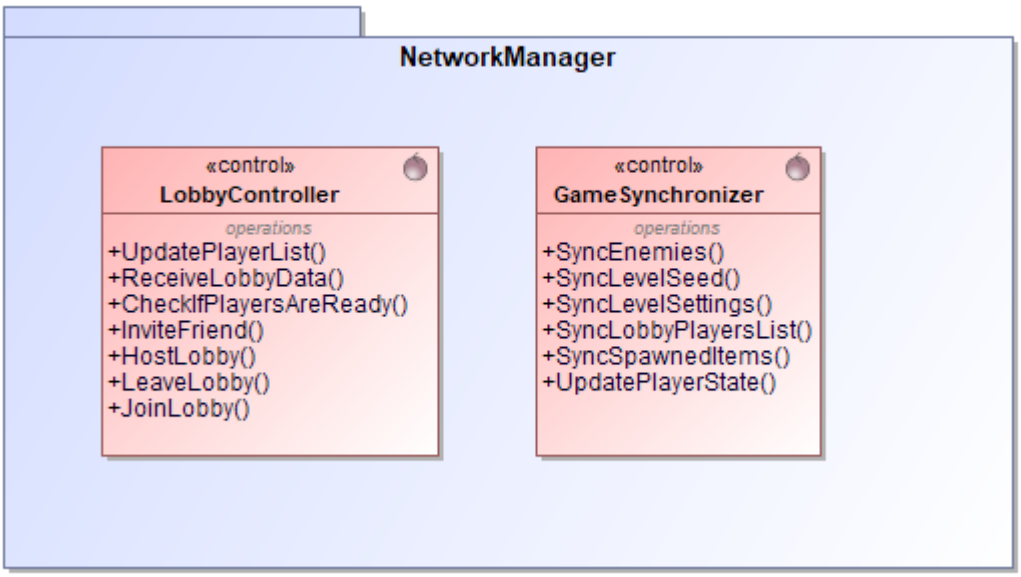
Kiekvienam realizuotam paketui pateikiama klasių diagrama ir trumpas paketo aprašymas.

17-ajame paveikslėlyje pavaizduota grafinės vartotojo sąsajos paketo klasių diagrama. Čia išskirti pagrindiniai valdikliai susiję su grafine sąsaja ir per ją atliekamais žaidimo veiksmais: „*ShopController*“ – valdiklis skirtas atlikti veiksmus žaidimo parduotuvėje, „*InventoryController*“ – valdiklis atlieka operacijas susijusias su žaidėjo daiktų saugykla ir jo turimais ginklais ar daiktais, „*MainMenuUIManager*“ valdiklis skirtas likusiems pradinio meniu scenos veiksmams, tokiems kaip nustatymų lango ar žaidėjo statistikos rodymui, atlikti.



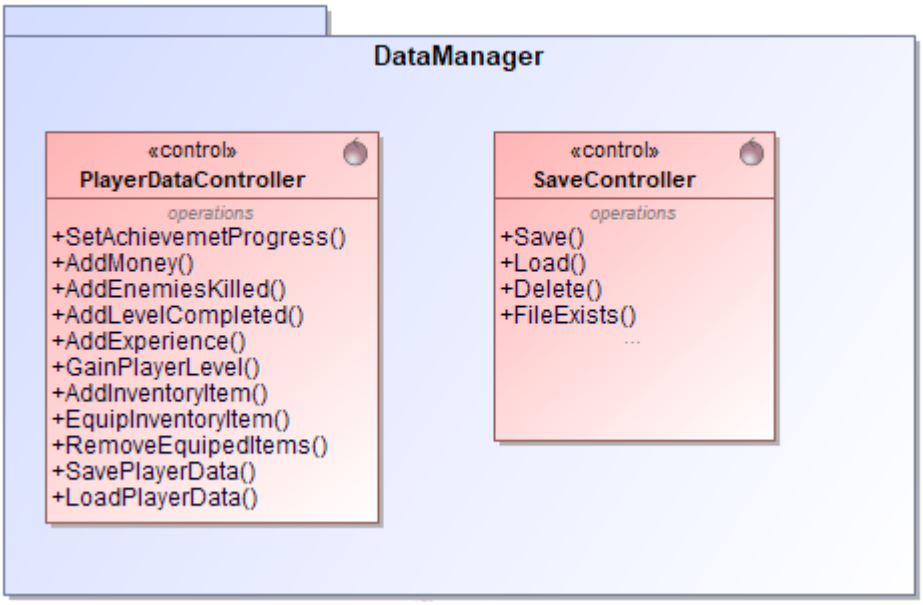
17 pav. Grafinės vartotojo sąsajos paketo klasių diagrama

18-ajame paveikslėlyje pavaizduota tinklo valdymo paketo klasių diagrama. Tinklo valdymo paketas atsakingas už serverio ir jo kambario (angl. *lobby*) sukūrimą, prisijungimą prie jo, draugų pakvietimus ir bendrą žaidimo eigos sinchronizaciją tarp žaidėjų. Šioje diagramoje išskirtos dvi pagrindinės klasės: „*LobbyController*“ ir „*GameSynchronizer*“.



18 pav. Tinklo valdymo paketo klasių diagrama

19-ajame paveikslėlyje pavaizduota žaidėjo duomenų valdymo paketo klasių diagrama. Žaidėjo duomenų valdymo paketas atsakingas už žaidėjo statistikos, pasiekimų, nustatymų, turimų daiktų, ginklų ir pinigų išsaugojimą diske tam, kad žaidėjo progresas nedingtų kiekvieną kartą naujai įsijungiant žaidimą.



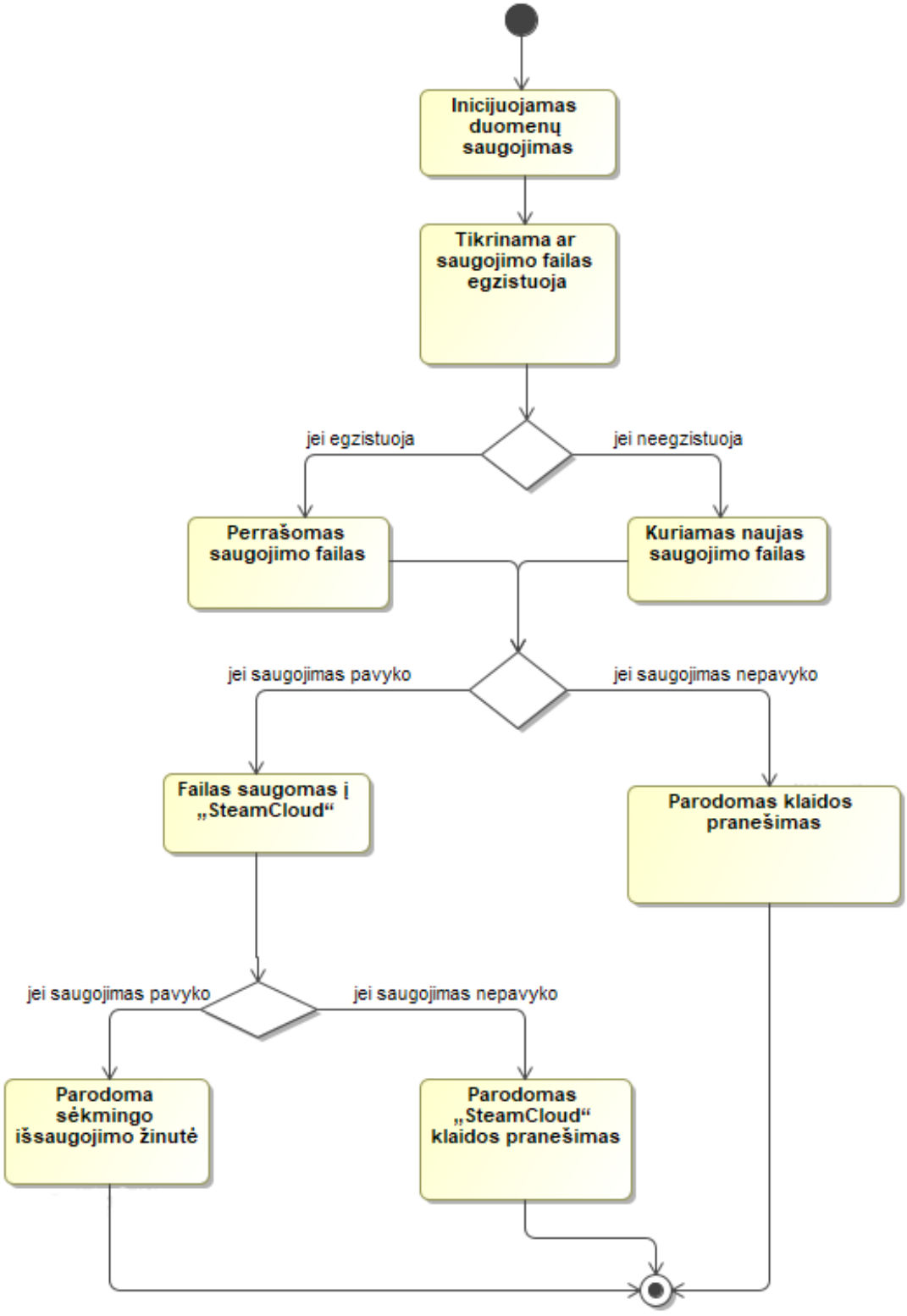
19 pav. Žaidėjo duomenų valdymo paketo klasių diagrama

2.4. Sistemos dinaminis vaizdas

Sistemos dinaminio vaizdo poskyryje pateikiamos būsenos ir sekų diagramos.

2.4.1. Būsenos diagramos

20-ajame paveikslėlyje pavaizduota duomenų saugojimo būsenos diagrama. Pradedant duomenų saugojimą, tikrinama ar jau egzistuoja saugojimo failas. Jeigu ne, sukuriamas naujas failas, o priešingu atveju, failas būna perrašomas vietoje jau egzistuojančio . Nepasisekus duomenis išsaugoti faile, rodoma klaida, o priešingu atveju, bandoma įkelti failą į „*SteamCloud*“ saugyklą. Parodoma sėkmės arba nesėkmės žinutė.

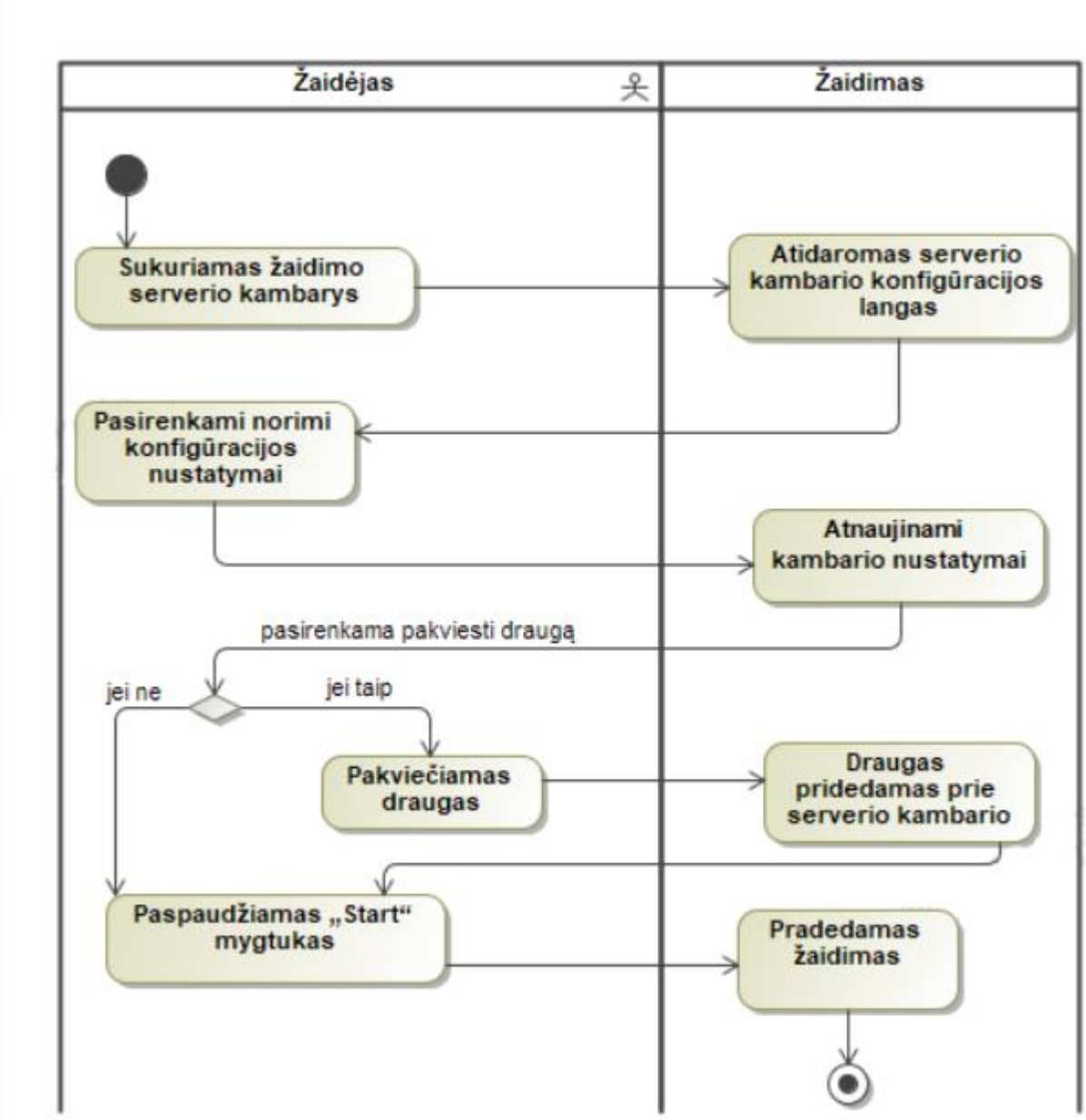


20 pav. Duomenų saugojimo būsenos diagrama

2.4.2. Veiklos diagramos

21-ajame paveiksle pavaizduota žaidimo pradėjimo veiklos diagrama. Prieš pradedant žaidimą, žaidėjui leidžiama pasirinkti lygio konfigūraciją ir pakviesti draugą. Paspaudus „*Start*“ mygtuką,

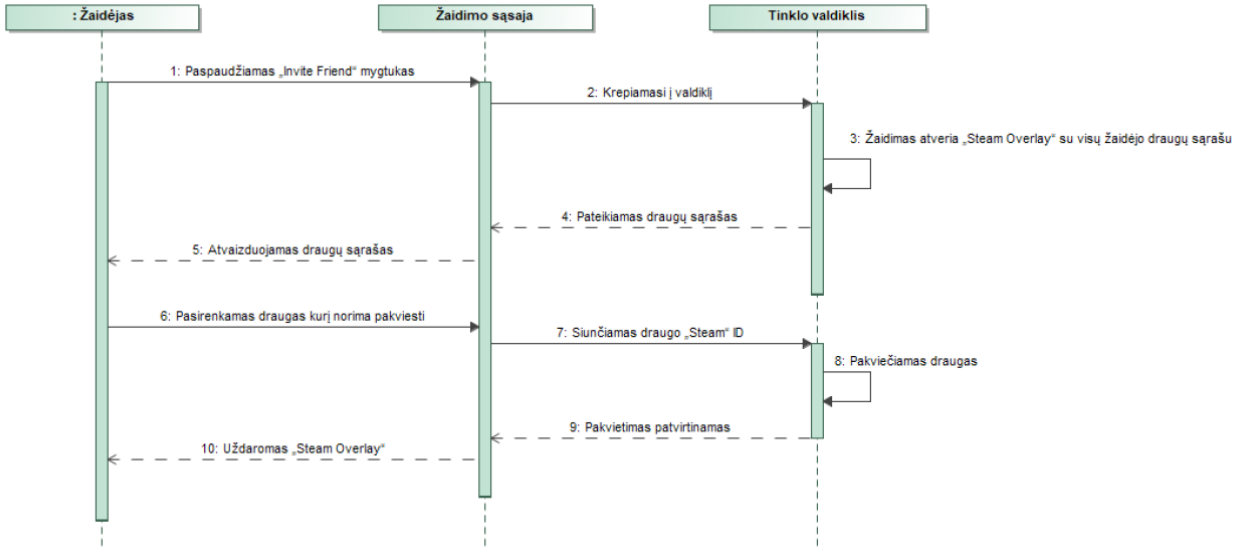
žaidėjas perkeliamas į misijos sceną ir ten pradedamas žaidimas pagal pasirinktus konfigūracijos nustatymus.



21 pav. Žaidimo pradėjimo veiklos diagrama

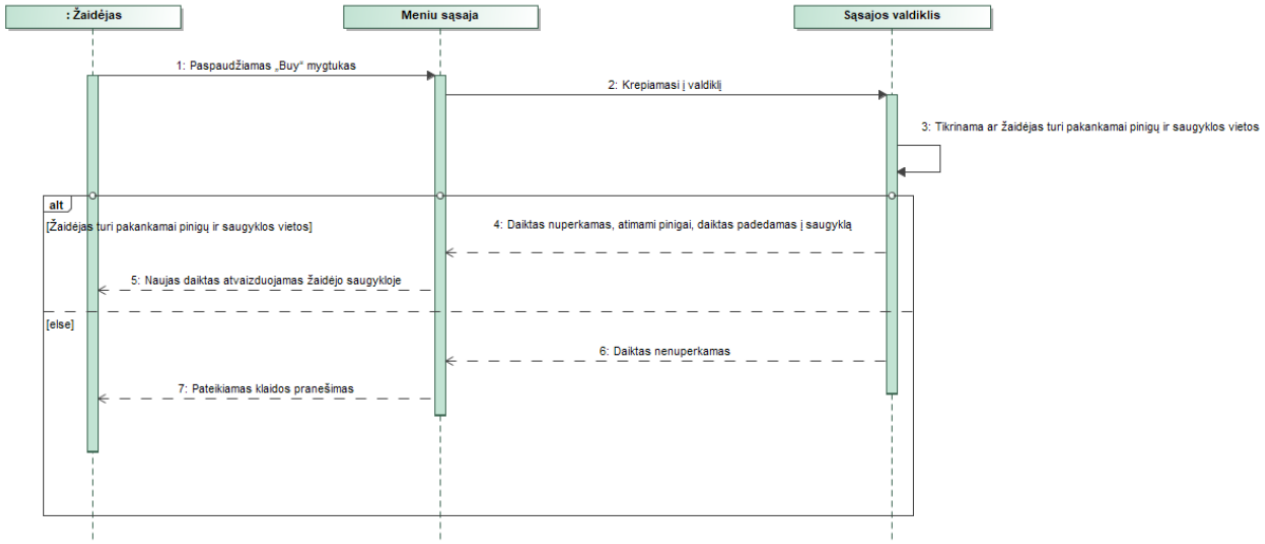
2.4.3. Sekų diagramos

22-ajame paveiksle pavaizduota komandos draugo pakvietimo į žaidimo serverio kambarį sekų diagrama.



22 pav. Draugo pakvietimo į žaidimo serverio kambarį sekų diagrama

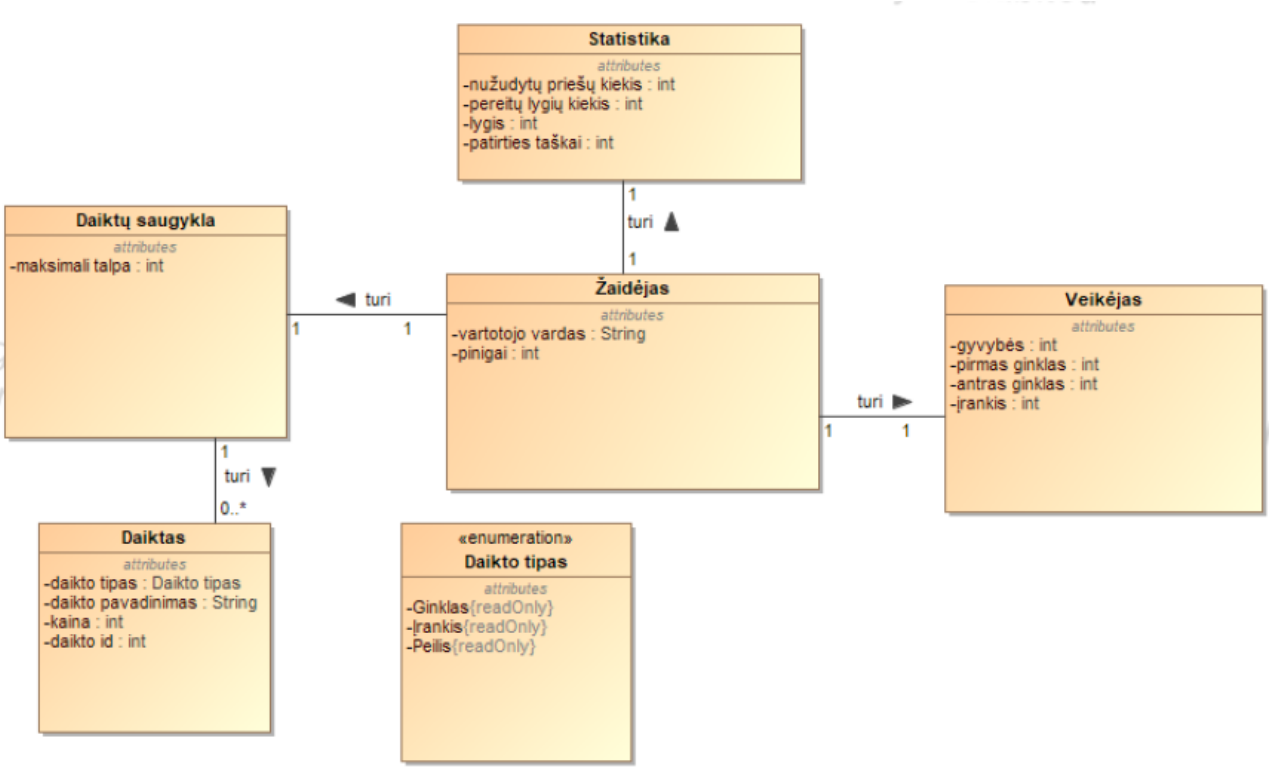
23-ajame paveikslėlyje pavaizduota daikto pirkimo parduotuvėje sekų diagrama.



23 pav. Daikto pirkimo parduotuvėje sekų diagrama

2.5. Duomenų vaizdas

24-ajame paveiksle vaizduojamas žaidimo naudojamų duomenų modelis. Šiame modelyje matomos esybės ir jų atributai naudojami žaidėjo duomenų saugojimo failuose.



24 pav. Sistemos duomenų modelis

2.6. Naudoti papildiniai

Žaidimo realizacijai įgyvendinti buvo panaudoti šie papildiniai:

- „*Aurora Engine – Mirror Network*“. Šaudymo ir veikėjo kontrolių papildinys su modeliais ir animacijomis.
- „*Mirror*“. Tinklo sąsajos biblioteka.
- „*FizzySteamworks*“. „*Mirror*“ sujungimo su „*SteamWorks*“ biblioteka.
- „*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.
- Zombių modeliai ir įvairios animacijos iš „*Mixamo*“.
- Žaidimo garsai panaudoti iš „*FreeSound*“.
- Žaidimo muzika panaudota iš „*Pixabay*“.

3. Tyrimo dalis

Šiame skyriuje apžvelgiama sukurta sistema ir jos kokybės įvertinimas pagal nustatytas metrikas. Tyrimas atliekamas su kompiuteriu, turinčiu tokią aparatinę įrangą: 8 GB operatyviosios atminties (RAM), *NVIDIA GeForce GTX 1050* vaizdo plokštę ir *Intel Core i5-4430* procesorių.

3.1. Tiriamos sistemos aprašymas

Tyrimas atliekamas žaidime pavadinimu „*Toxic Wasteland*“, kuris buvo sukurtas dirbant kartu su kolega Luku Kalade. Sistema realizuota naudojant „*Unity*“ žaidimų variklį. Žaidimą galima žaisti kartu su draugais – yra palaikomas kelių žaidėjų režimas (angl. *Multiplayer*). Visi kartu žaidžiantys žaidėjai turi bendrą tikslą – įvykdyti misiją, kuri yra nustatoma pagal pasirinktą žaidimo tipą.

Misijos gali būti dviejų tipų: „Išgyvenimo“ ir „Išvalymo“. „Išgyvenimo“ tipo misijos metu žaidėjai turi nukauti nurodytą priešų kiekį, o „Išvalymo“ tipo misijos metu apžvelgti ir išnaršyti visus sugeneruotus kambarius ir įveikti juose atsirandančius priešus. Priešai būna kelių skirtingų tipų. Kiekvienas tipas turi unikalias savybes, stiprumo charakteristikas ir duodamą atlygį. Lygio kambariai yra išdėstomi atsitiktine tvarka kiekvienos misijos metu

Žaidimas prasideda nuo pradinio meniu scenos, kur žaidėjas gali pasiruošti misijai – daiktų parduotuvėje nusipirkti norimus ginklus ir įrangą, o inventoriaus dalyje pasirinktus ginklus užsidėti. Pralaimėjus misiją, žaidėjo turėti daiktai yra prarandami ir žaidėjas juos turi vėl nusipirkti. Sėkmingai įveikus misiją žaidėjai išsaugo turimus daiktus, gauna pinigų ir patirties taškų, nuo kurių priklauso žaidėjo lygis. Nuo turimo lygio priklauso, kokius ginklus žaidėjui leidžiama nusipirkti. Gaunamų patirties taškų kiekis priklauso nuo lygio nustatymų: lygio dydžio, sunkumo, nukautų priešų kiekio. Taip pat, žaidėjai gali sekti savo statistikos rodiklius: nukautų priešų kiekį, sukauptus patirties taškus, įveiktų lygių kiekį, mirčių kiekį, nukautų priešų ir mirčių santykį bei išleistų pinigų kiekį. 25-ajame paveiksle pateikiama ištrauka iš žaidimo misijos.

25 pav. Ištrauka iš žaidimo

3.2. Sistemos įvertinimo metrikos

Sukurto žaidimo našumo ir optimizacijos lygio įvertinimas bus atliekamas pagal tris metrikas:
1. Žaidimo kadrų per sekundę kiekis.
2. Žaidimo operatyviosios atminties sunaudojimas.
3. Sukompiliuoto žaidimo užimamas dydis.

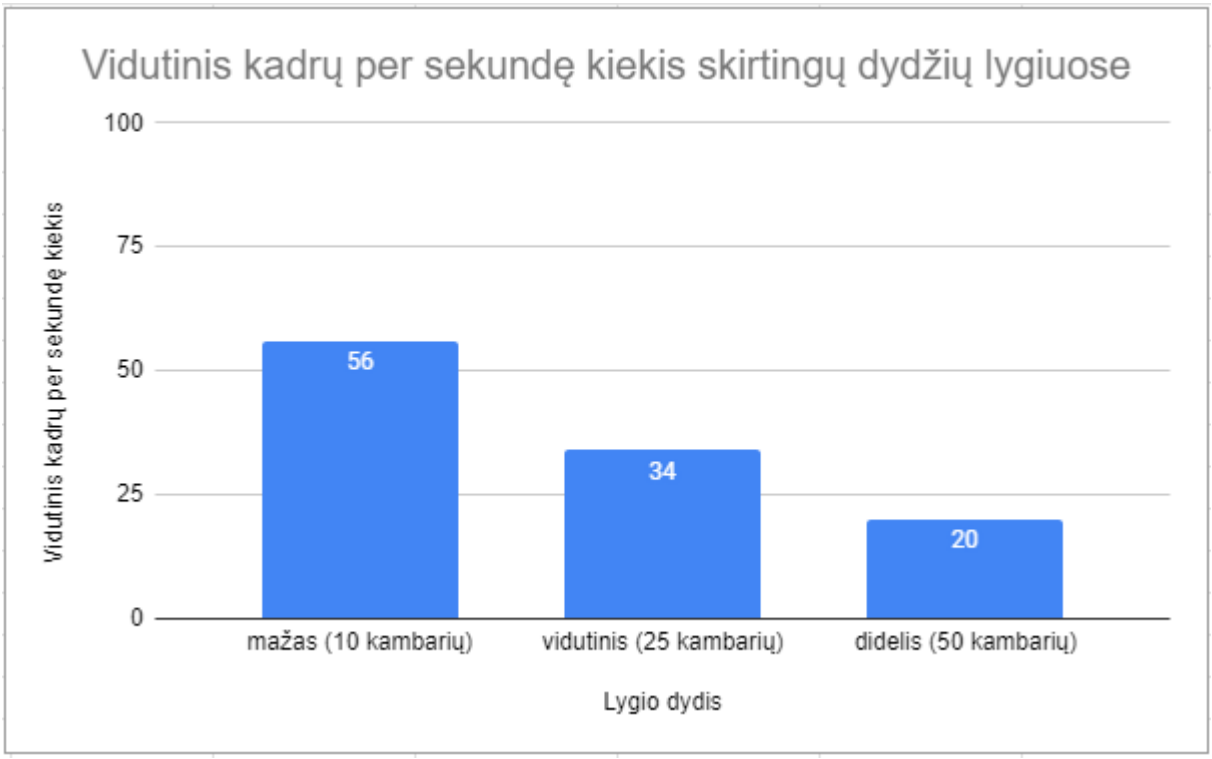
Tiriant žaidimo kadrų per sekundę kiekį ir žaidimo operatyviosios atminties sunaudojimą, bus išjungtos žaidime esančios atsitiktinio turinio generavimo sistemos dalys, atsakingos už priešų ir daiktų generavimą, kambarių daiktų bei sienų tekstūrų parinkimą. Tyrimo metu lygio kambarių išdėstymui bus naudojama ta pati atsitiktinės atrankos sėkla (angl. *seed*), kuri užtikrina, kad kiekvieno bandymo metu kambarių išsidėstymas bus vienodas ir nedarys įtakos greitaveikos matavimams. Taip pat, reikia pažymėti, kad tyrimas buvo atliekamas „*Unity*“ redaktoriuje. Testuojant greitaveiką redaktoriuje dalis resursų yra skiriama būtent paties „*Unity*“ variklio veiklai palaikyti [23]. Testavimo metu buvo įjungiamas padidintas žaidimo režimo vaizdas (angl. *Play mode view*), tam kad žaidimui nereiktų papildomai atlikti piešimo darbų kitiems redaktoriaus langams. Kadrų per sekundę kiekio ir atminties sunaudojimo matavimams buvo panaudotas „*Unity Profiler*“ įrankis. Sukompiliuoto žaidimo užimamam dydžiui analizuoti panaudotas „*Build Report*“ įrankis, kuris gali parodyti kokie konkretūs failai yra įtraukiami į sukompiliuotą versiją ir kiek tie failai užima vietos [24].

3.3. Kadrų per sekundę vidutinis kiekis

Kadrų per sekundę kiekis parodo kaip dažnai žaidėjui yra atnaujinamas vaizdas. Kuo dažniau vaizdas yra atnaujinamas (t.y. kuo didesnis kadrų per sekundę kiekis), tuo žaidimas yra sklandesnis. Laikoma, kad žaidimas veikia pakankamai sklandžiai, jei jo vidutinis kadrų per sekundę kiekis siekia bent 60. Jeigu žaidimas šios ribos nesiektų, jis gali tapti nežaidžiamas – žaidėjams gali būti sunku išlaikyti dėmesį stebint lėtai ir nesklandžiai besikeičiantį vaizdą. Taip pat gali atsirasti nemalonus įvesčių vėlavimas (angl. *Input lag*), kuomet paspaudus mygtuką ar atliekant kitą įvesties veiksmą, atsirastų delsa to veiksmo atvaizdavimo žaidime.

3.3.1. Kadrų per sekundę vidutinio kiekio tyrimas

Kadangi prieš pradedant misiją žaidėjas gali pasirinkti lygio dydžio nustatymą, tyrimo metu bus matuojamas kadrų per sekundę kiekis skirtingo dydžio lygiuose. Atliekant tyrimą, buvo pastebėta, kad vidutinis kadrų per sekundę kiekis priklauso nuo lygio dydžio (tiksliau, nuo lygio kambarių kiekio). Mažas lygis yra sudarytas iš 10-ies kambarių, vidutinio dydžio lygis iš 25-ių kambarių, o didelį lygį sudaro 50 kambarių. 26-ajame paveiksle pateikiami kadrų per sekundę vidutinio kiekio tyrimo rezultatai. Žaidimas net mažo dydžio lygio metu nesugebėjo pasiekti 60-ies kadrų per sekundę ribos, o dideliame lygyje vidutinis kadrų kiekis siekė vos 20.



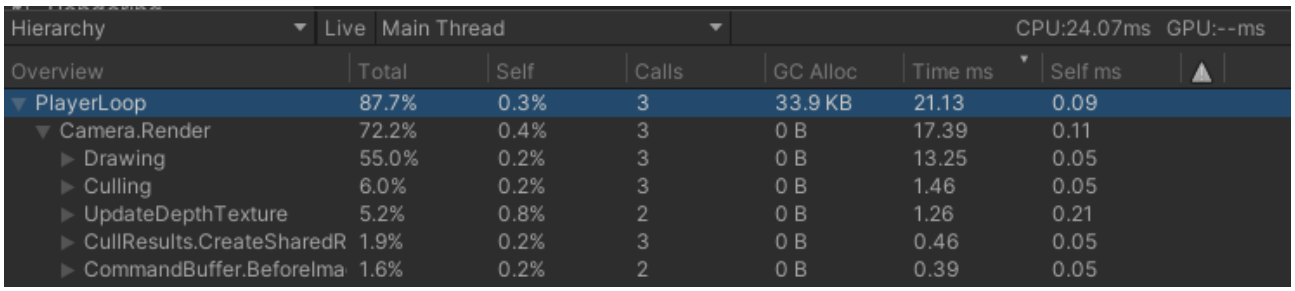
26 pav. Vidutinis kadrų per sekundę kiekis skirtingų dydžių lygiuose

Pradinio meniu scenoje, žaidimas stabiliai siekia bent 200 kadrų per sekundę. Šioje scenoje daugiausiai resursų sunaudojama grafinei sąsajai atvaizduoti. Grafinės sąsajos hierarchija buvo sudaryta sluoksniais, aktyvuojant tik reikalingus objektus atvaizdavimui taip mažinant persidengiančių objektų kiekį. 27-ajame paveiksle matoma sukurtos grafinės sąsajos struktūra. Kairėje paveikslėlio pusėje matoma hierarchija, kai įjungtas parduotuvės langas, o dešinėje pusėje matoma hierarchija, kai įjungtas inventoriaus langas. Abiem atvejais įjungiami tik reikalingi sąsajos langai ir jų objektai esantys žemesniame hierarchijos lygyje.

27 pav. Grafinės sąsajos struktūra

3.3.2. Kadrų per sekundę kiekio patobulinimas

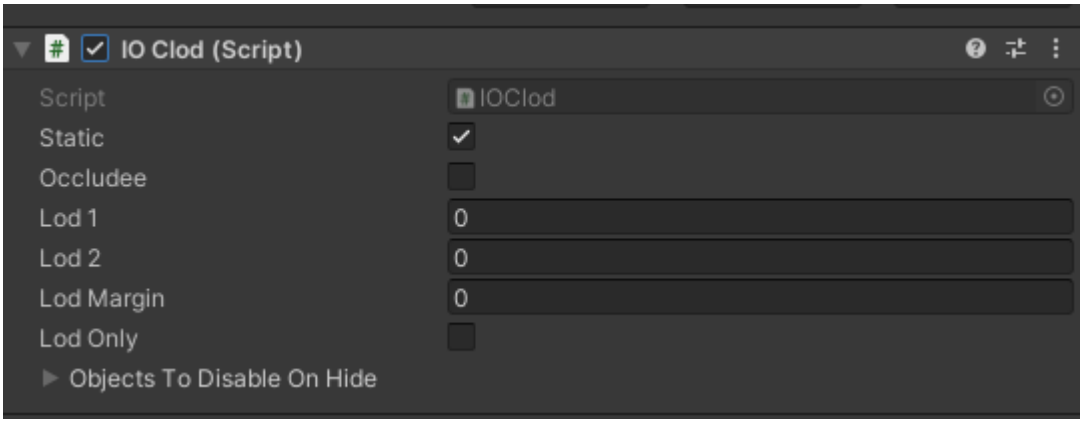
Norint pagerinti žaidimo greitaveiką, pirmiausiai reikia išsiaiškinti, kas sunaudoja daugiausiai laiko vieno kadro apdorojimo metu. 28-ajame paveiksle matomas „*Unity Profiler*“ langas, kuriame matyti, kad 72,2% vieno kadro laiko yra sunaudojama kameros grafinio apdorojimo ir piešimo užduotims.



28 pav. „*Unity Profiler*“ CPU įverčių langas

Siekiant padidinti žaidimo kadrų per sekundę vidutinį kiekį, buvo pritaikyti analizės metu apžvelgti optimizacijos metodai. Pirmiausiai taikytas nematomų objektų paslėpimo metodas, nes jis turėtų paslėpti lygyje esančius daiktus nematomus žaidėjo kamerai ir sumažinti lygyje grafiškai apdorojamų objektų kiekį, taip kartu sumažinant ir žaidėjo kameros darbo apkrovą. Kadangi „*Unity*“ siūlomi paslėpimo sistemos įrankiai veikia tik iš anksto suprojektuotiems lygiams ir netinka atsitiktinai gerneruojamiems lygiams, buvo panaudotas „*InstantOC*“ įrankis su dinaminio paslėpimo galimybe [25].

Kambarių objektams, iš kurių yra kuriami žaidimo lygiai, buvo priskirti *IOClod* komponentai, matomi 29-ajame paveiksle. Šiems komponentams buvo parinktas *Static* parametras, kadangi žaidimo kambariai nekeičia savo koordinačių ar išmatavimų po to kai būna sukūriami ir inicijuojami. Priešams buvo priskirtas „*Ocludees*“ sluoksnis (angl. *Layer*).



29 pav. *IOClod* komponentas

Priešingai, nei „*Unity*“ paslėpimo sistema, šis komponentas negeneruoja išankstinių duomenų ir skaičiavimus atlieka realiu laiku vykstant žaidimui. Priklausomai nuo naudojamų objektų kiekio, gali padidėti CPU resursų naudojimas skaičiavimams. Taip pat, kameros atstumų skaičiavimo algoritme *Physics.Raycast* funkcija buvo pakeista mažiau atminties šiukšlių generuojančia *Physics.RaycastNonAlloc* funkcija.

3.4. Atminties sunaudojimas

Žaidimo operatyviosios atminties sunaudojimas aktualus tuo, kad netinkamas atminties valdymas, gali priversti žaidimą nepageidautinai išsijungti (angl. *crash*). Rekomenduojama, kad žaidimas naudotų iki dviejų trečdalių galimos operatyviosios atminties, nes likusi jos dalis turi būti palikta kitoms kompiuterio funkcijoms. Kadangi siekiama, kad žaidimas veiktų bent 8 GB RAM turinčiuose kompiuteriuose, pageidautinas žaidimo operatyviosios atminties sunaudojimas turėtų siekti iki 5 GB.

Taip pat, svarbu, kad atminties lygis drastiškai nesikeistų ir nebūtų atminties nuotėkių (angl. *memory leaks*). Atminties nuotėkių tikrinimui buvo atlikti vadinamieji mirkymo (angl. *soak*) testai. Jų metu įjungtas žaidimas paliekamas bent 12-ai valandų. Praėjus laikui patikrinama ar žaidimas nebuvo nepageidautinai išjungtas dėl nutekėjusios atminties.

3.4.1. Atminties sunaudojimo tyrimas

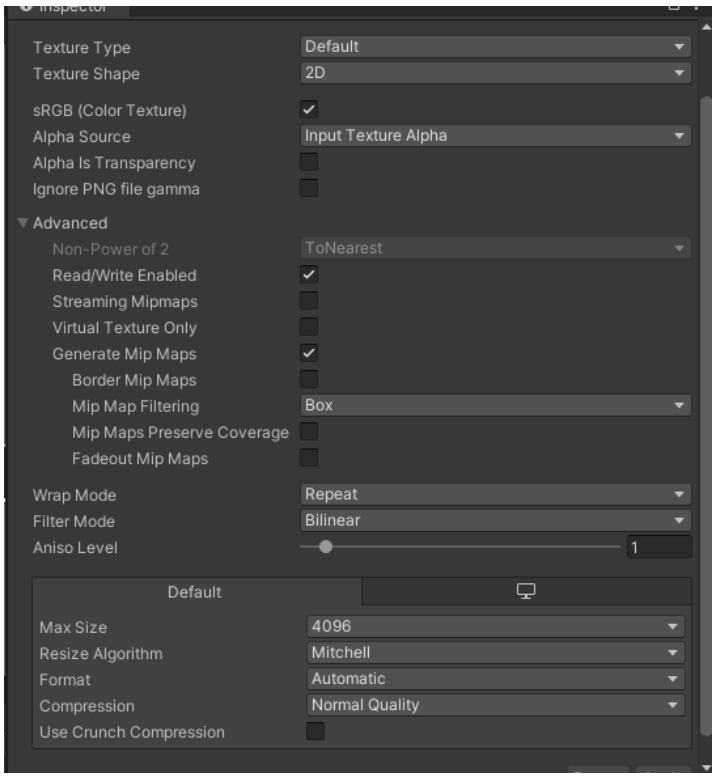
Atliekant atminties sunaudojimo tyrimą, buvo pastebėta, kad lygio dydis neturi įtakos atminties sunaudojimui. 30-ajame paveiksle pateikiami žaidimo operatyviosios atminties matavimo rezultatai.

30 pav. Operatyviosios atminties sunaudojimas

Iš viso žaidimas naudojo šiek tiek daugiau nei 3 GB operatyviosios atminties. Daugiausiai atminties resursų naudojo tekstūros (apie 1,5 GB atminties), garso klipai (apie 100 MB atminties), modeliai (apie 60 MB atminties) ir maždaug 900 MB atminties naudojo pats „*Unity Profiler*“ įrankis ir „*Unity*“ redaktorius. Taip pat, atliekant mirkymo (angl. *soak*) testus buvo tikrinama ar sistema yra atspari atminties nuotėkiams — įjungtas žaidimas buvo paliktas dvylikai valandų. Po dvylikos valandų žaidimas nebuvo užstrigęs ar nepageidaujamai išsijungęs ir vis dar stabiliai veikė. Tai reiškia, kad didelių atminties nuotėkių nėra, o mažesni nuotėkiai neigiamai nepaveiks žaidėjo žaidimo patirties.

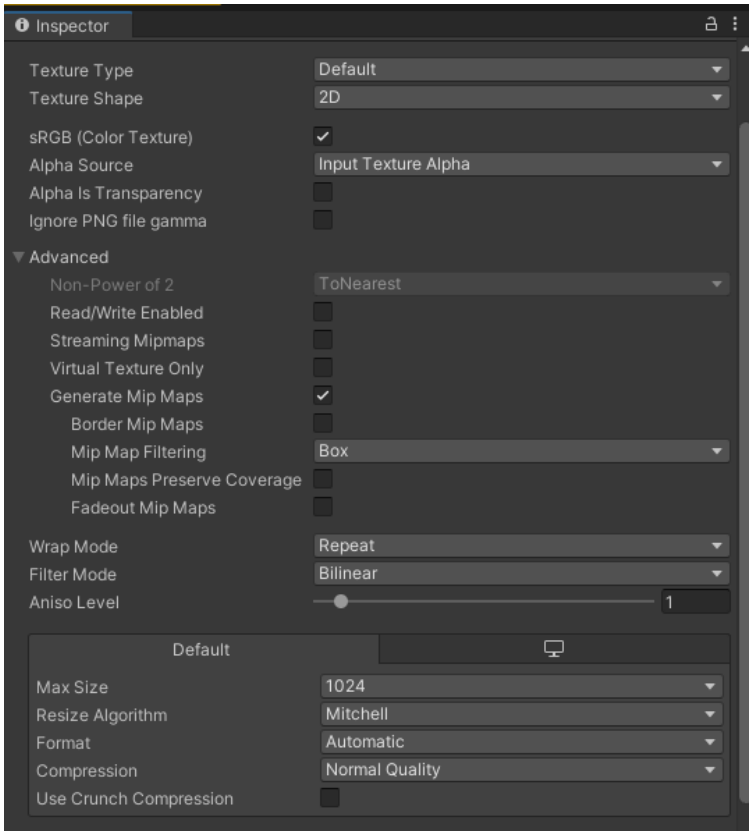
3.4.2. Atminties sunaudojimo patobulinimas

Nepaisant to, kad atminties sunaudojimo rodikliai atitiko iškeltus reikalavimus, juos galima pabandyti pagerinti panaudojant analizės metu įgytas žinias apie gerąsias žaidimo kūrimo praktikas. Dar kartą buvo apžvelgtos turimos tekstūros, kadangi jos sudarė apie pusę visos sunaudojamos atminties. Tekstūroms buvo uždėti maksimalios raiškos dydžių limitai ir išjungtas *Read/Write Enabled* nustatymas, nes jos žaidimo metu nebūna modifikuojamos. 31-ajame paveiksle matomi pasirinktos tekstūros nustatymai prieš patobulinimą. Tekstūra buvo 4096-ių pikselių dydžio ir turėjo įjungtą *Read/Write Enabled* nustatymą.



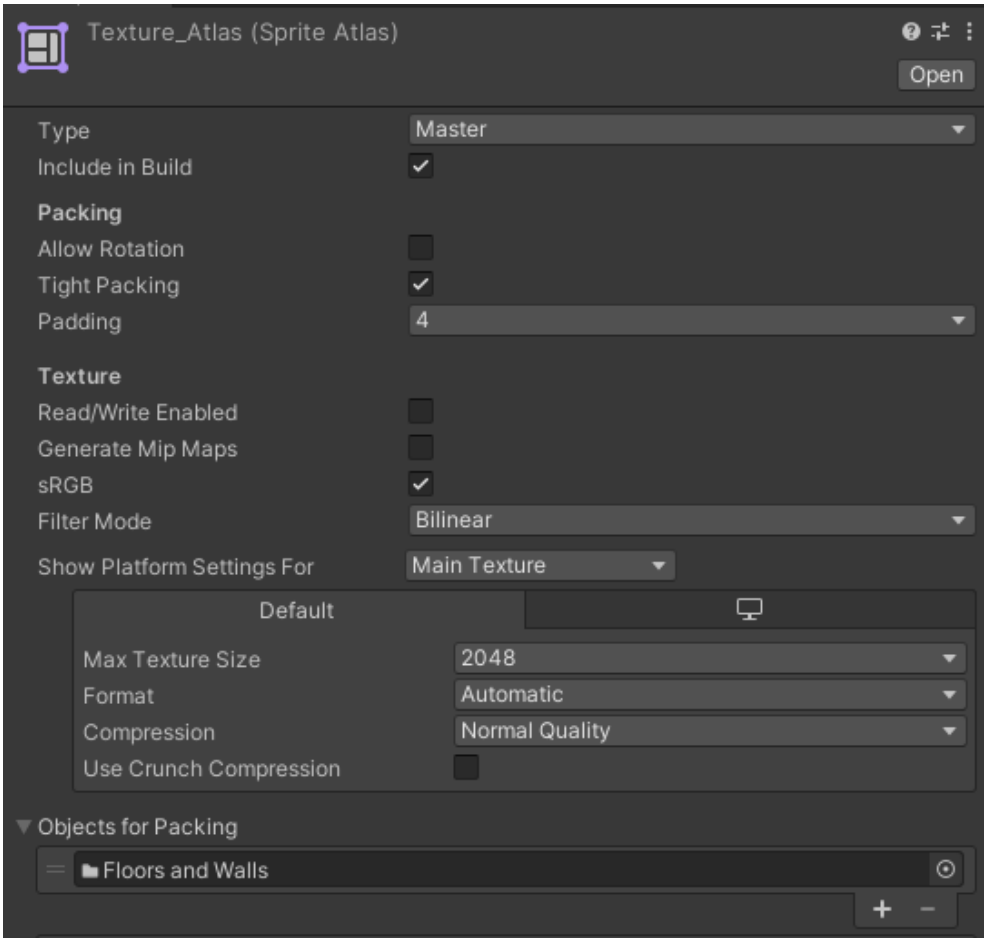
31 pav. Tekstūros nustatymai prieš patobulinimą

32-ajame paveiksle matomi tos pačios tekstūros parametrai po patobulinimų. Tekstūrai pakeistas dydžio limitas į 1024 pikselius ir išjungtas *Read/Write Enabled* nustatymas.



32 pav. Tekstūros nustatymai atlikus patobulinimus

Tokie patys nustatymai buvo pritaikyti ir kitoms naudojamoms tekstūroms. Taip pat, dažnai kartu naudojamos tekstūros buvo sugrupuotos ir sudėtos į tekstūrų atlasus (angl. *Sprite Atlas*), siekiant sumažinti piešimo užklausų (angl. *Draw calls*) kiekį. Atlaso pavyzdys pateikiamas 33-ajame paveiksle.



33 pav. Tekstūrų atlasas

Tekstūrų suspaudimo nustatymai šiame etape dar nebuvo naudojami, nes laikinojoje atmintyje laikomos tekstūros būna grąžinamos į pradinę kokybę ir dekompresuojamos išlaikant tik nustatytus maksimalius raiškos nustatymus.

3.5. Sukompiliuoto žaidimo užimamas dydis

Sukompiliuoto žaidimo dydžiui ypatingi reikalavimai nėra keliami, tačiau pageidautina, kad jis neviršytų 1 GB.

3.5.1. Sukompiliuoto žaidimo užimamo dydžio tyrimas

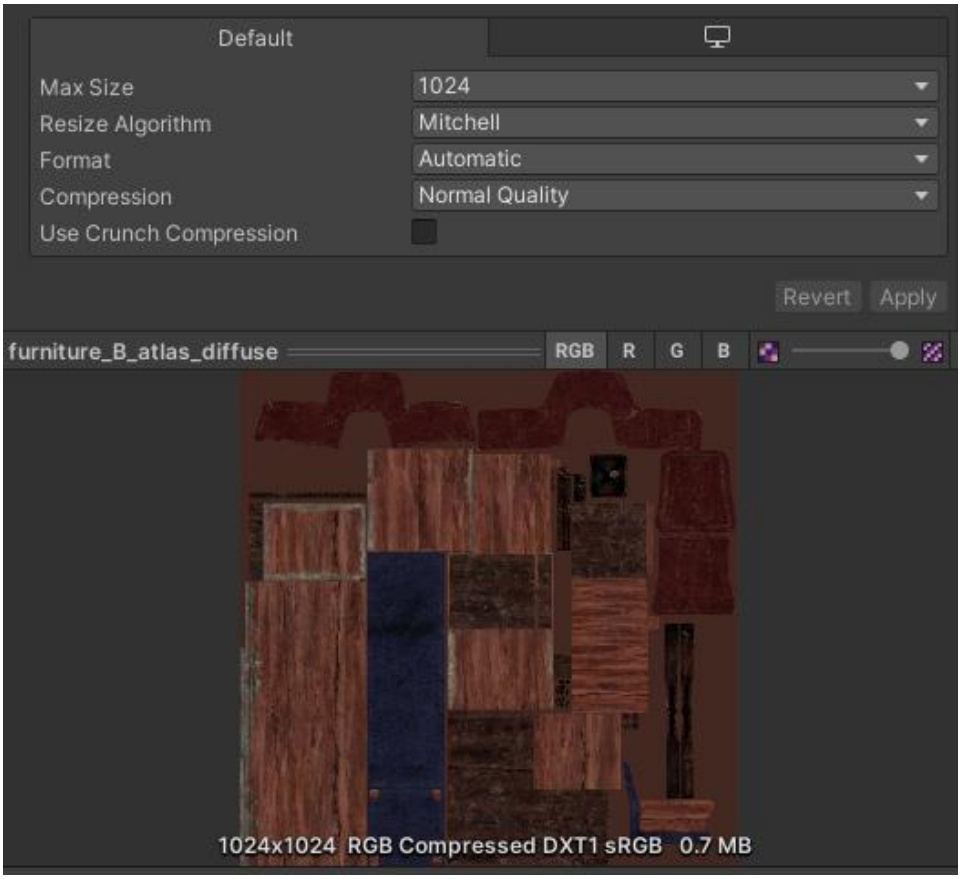
Sukompiliuoto žaidimo užimamo dydžio tyrimui buvo naudotas „*Build Report*“ įrankis. Jis ne tik pateikia galutinį žaidimo dydį, bet ir parodo kurios dalys ir naudoti resursai užima daugiausiai atminties – tai leidžia identifikuoti problemines vietas ir visą dėmesį skirti identifikuotų vietų taisymui. 34-ajame paveiksle matomi sukompiliuoto žaidimo dydžio duomenys.

34 pav. Sukompiliuoto žaidimo dydžio duomenys

Visas žaidimas užima 1,24 GB, iš kurių net 1 GB užima nepilnai sukompresuotos tekstūros. Likusi atmintis naudojama saugoti garsus, modelių geometrijos duomenis, sistemos bibliotekas, animacijas ir kitus resursus.

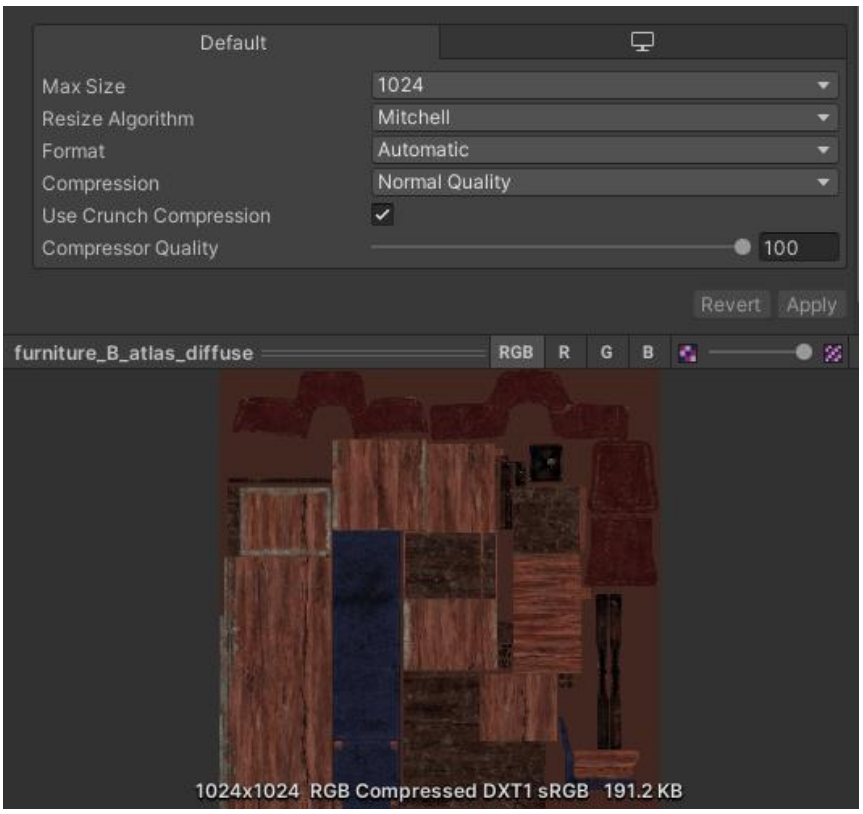
3.5.2. Sukompiliuoto žaidimo užimamo dydžio patobulinimas

Sukompiliuoto žaidimo užimamam dydžiui mažinti bus naudojama tekstūrų suspaudimo technika. 35-ajame paveiksle matomi pasirinktos tekstūros suspaudimo nustatymai prieš patobulinimą. Pagal nutylėjimą nėra naudojamas *Use Crunch Compression* parametras, tekstūra užima 0,7 MB atminties.



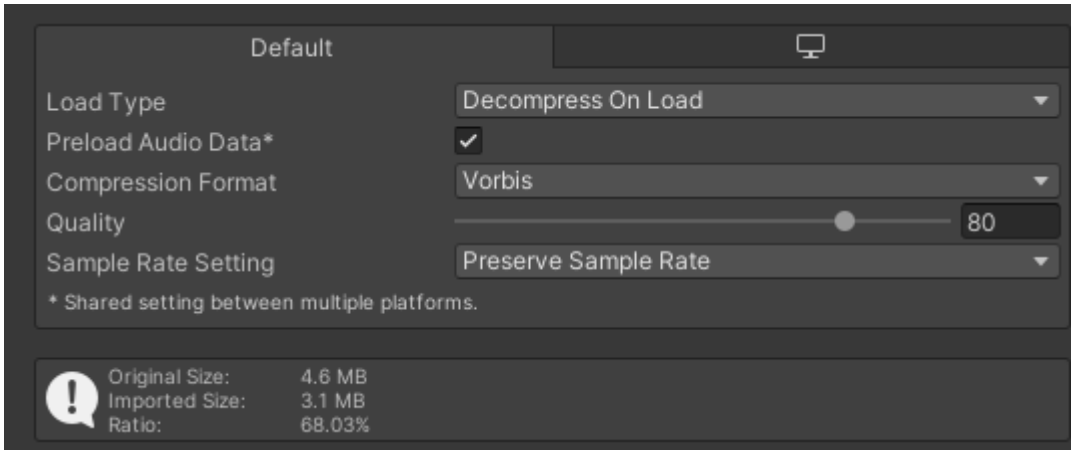
35 pav. Tekstūros dydis prieš suspaudimą

36-ajame paveiksle matomi pasirinktos tekstūros dydžio duomenys po pilno suspaudimo. Išlaikoma geriausia suspaudimo kokybė. Tekstūros dydis po suspaudimo – 191,2 KB.



36 pav. Tekstūros dydis po suspaudimo

Muzikos ir garso efektų užimamam dydžiui mažinti buvo pasirinkta šiek tiek pakeisti jų kokybę. Klausantis efekto sunku išgirsti ir rasti skirtumų tarp jo pilnos kokybės versijos ir versijos, kurios kokybė sumažinta iki 80%, tačiau tai turėtų padėti varikliui geriau suspausti failus sukompiliuotoje versijoje. Pasirinkto garso efekto nustatymai matomi 37-ajame paveikslėlyje.



37 pav. Garso efekto nustatymai

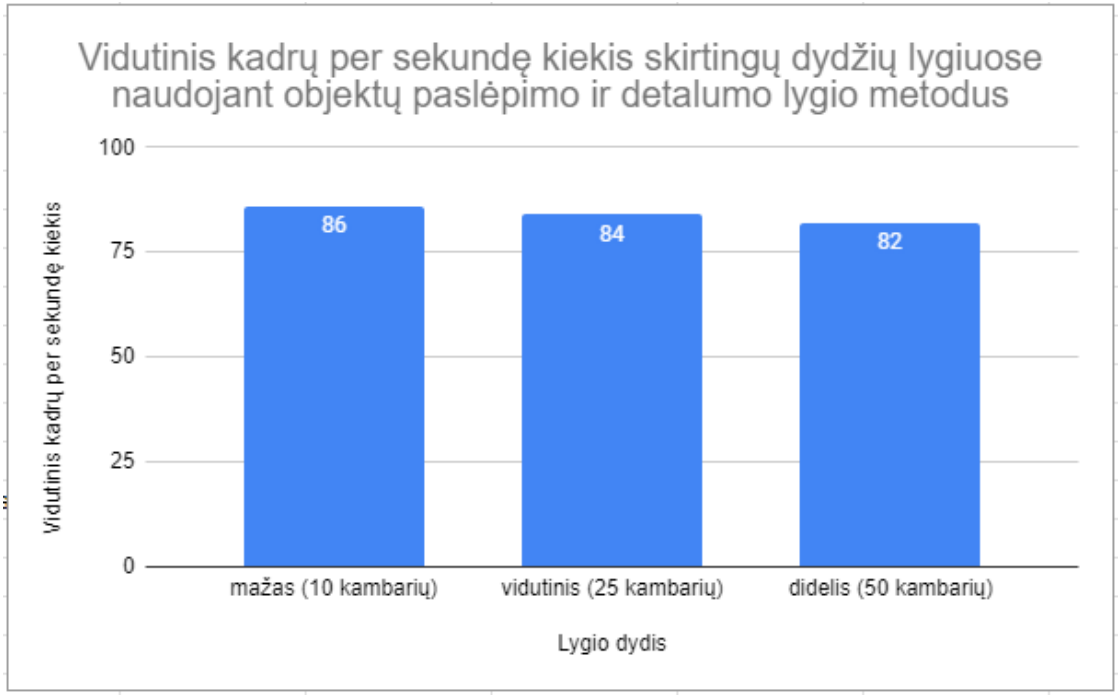
Taip pat, buvo pakeistas žaidimo kompiliavimo suspaudimo metodas iš „Numatytojo“ į „LZ4HC“, kuris yra rekomenduojamas žaidimo išleidimo versijoms kompiliuoti [26].

4. Eksperimentinė dalis

Eksperimentinėje dalyje atliekamas tyrimo metu pritaikytų našumo optimizacijos metodų ir technikų patobulinimų įvertinimas. Eksperimentai buvo atliekami su tokių pačių parametrų kompiuteriu kaip ir tyrimo dalyje.

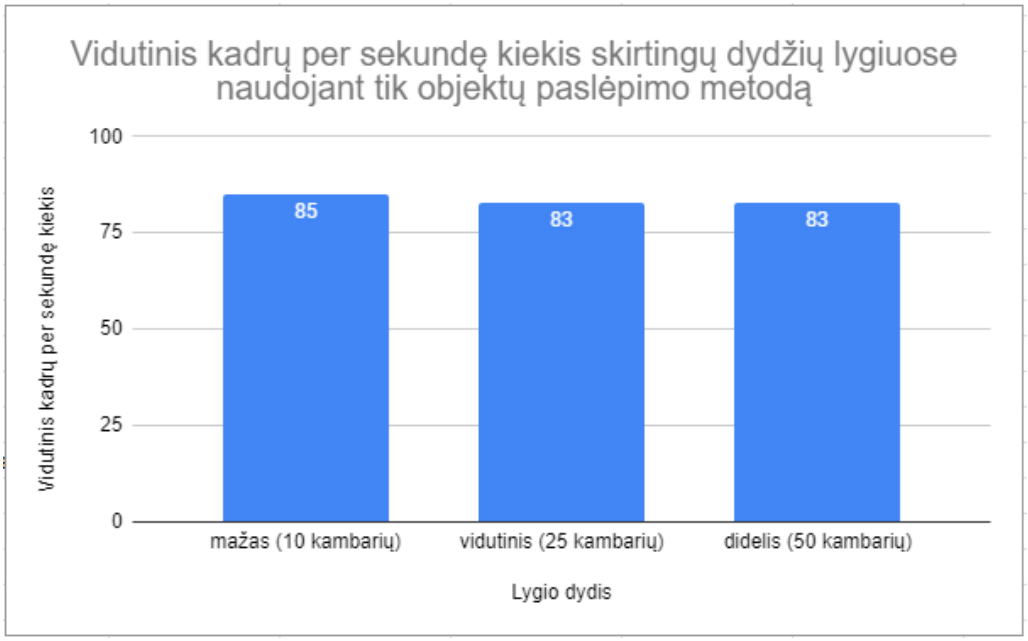
4.1. Kadrų per sekundę vidutinio kiekio eksperimentas

Buvo atliktas vidutinio kadrų per sekundę kiekio eksperimentinis tyrimas su įjungta ir patobulinta objektų paslėpimo sistema bei su įjungta objektų detalumo lygio valdymo sistema. 38-ajame paveiksle matomi šio kadrų per sekundę kiekio eksperimentinio tyrimo rezultatai. Kadrų per sekundę kiekio rodikliai tapo gerokai mažiau priklausomi nuo lygio dydžio – žaidimas vidutiniškai siekė apie 84 kadrus per sekundę. Tai galima paaiškinti tuo, kad objektų paslėpimo metodas sėkmingai paslepia žaidėjui nematomus daiktus ir grafikos plokštei nebereikia jų bereikalingai grafiškai apdoroti. Tad nepaisant koks yra žaidimo lygio dydis, grafiškai apdorojami tik žaidėjui tiesiogiai matomi objektai.



38 pav. Vidutinis kadrų per sekundę kiekis po patobulinimų

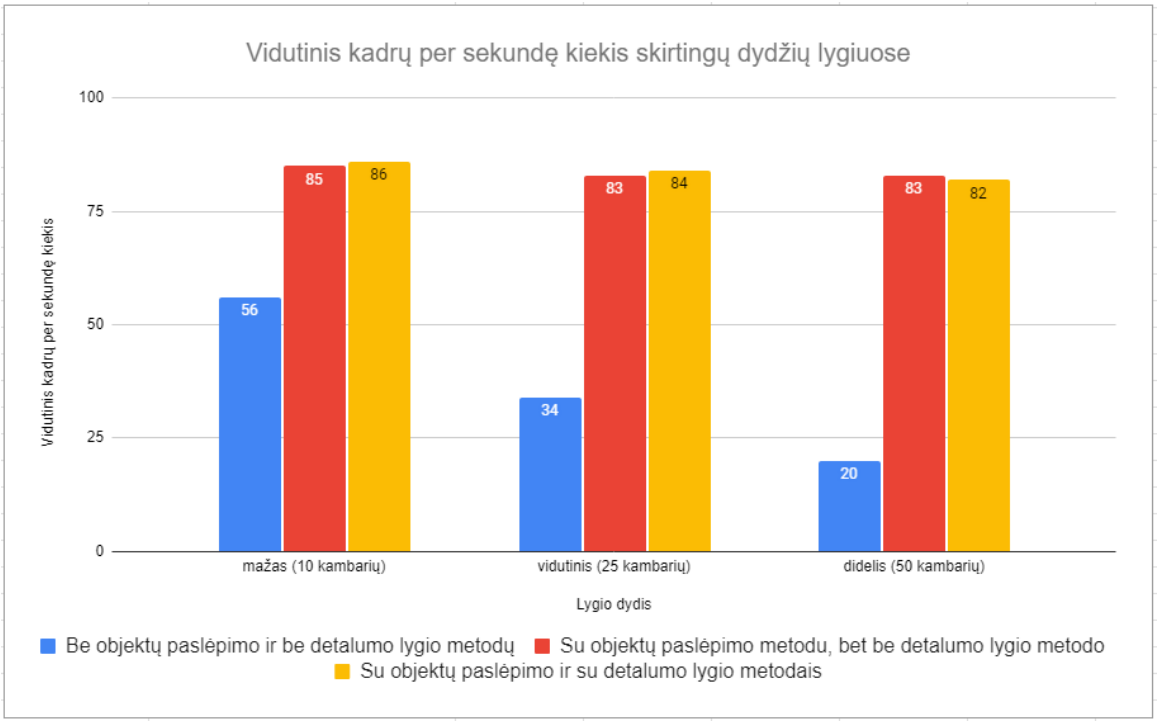
Tiriant žaidimo eigą, buvo pastebėta, kad kai kurie objektai netikėtai dingsta arba atsiranda priklausomai nuo žaidėjo atstumo iki jų. Išsiaiškinta, kad šį nesklandumą sukėlė detalumo lygio sistema, kuri išjungdavo šiuos objektus, kad jų nereikėtų grafiškai apdoroti dėl klaidingai nustatytos atstumo ribos. Kadangi žaidimas vyksta siauruose kambariuose, žaidėjui visi kambaryje matomi objektai būna pakankamai arti ir išlieka aukšto detalumo lygio. Žaidėjui pakankamai nutolus nuo objektų, pastarieji dažniausiai jau būna užstoti kitų objektų ar sienų. Tada suveikia objektų paslėpimo metodas ir objektai būna išjungiami bet kokiu atveju. Dėl to, detalumo lygio metodo įtaka našumui sumažėja. 39-ajame paveiksle pateikiami rezultatai išjungus detalumo lygio valdymo komponentus, tačiau su vis dar naudojamu objektų paslėpimo metodu.



39 pav. Vidutinis kadrų per sekundę kiekis be detalumo lygio metodo

Išjungus detalumo lygio metodą, vidutinis kadrų per sekundę kiekis iš esmės nepasikeitė. Galima teigti, kad šiam konkrečiam žaidimui detalumo lygio metodas našumo nepagerino.

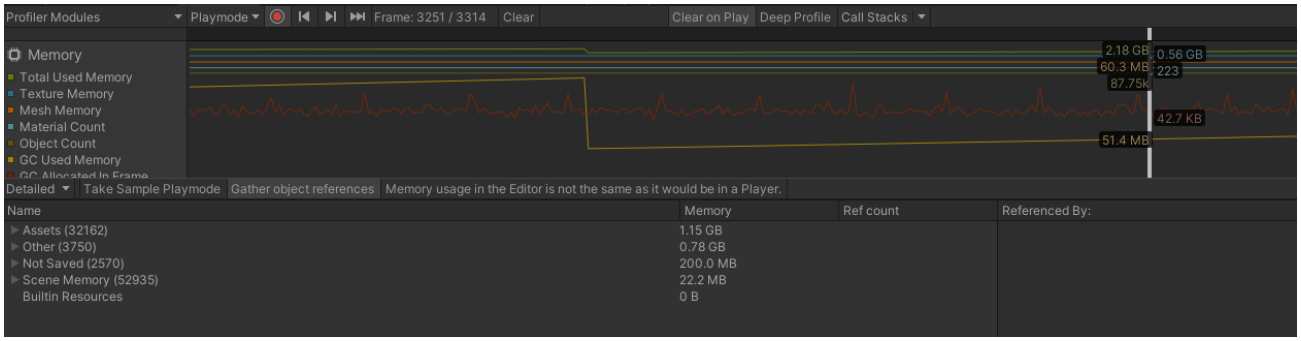
Toliau 40-ajame paveiksle pateikiamas bendras visų metodų daromos įtakos žaidimo kadrų per sekundę skaičiui palyginimas. Mažo dydžio lygiuose objektų paslėpimo metodas vidutinį kadrų per sekundę kiekį pagerino 52 procentais, vidutinio dydžio lygiuose – 144 procentais, o dideliuose lygiuose – 315 procentų. Detalumo lygio metodas įtakos nepadarė dėl sąlyginai mažų žaidimo kambarių dydžių – objektai niekada nebūna pakankamai toli, kad metodas galėtų gerinti našumą mažinant atvaizduojamų objektų detalumo lygį.



40 pav. Bendras visų metodų kadrų kiekio per sekundę palyginimas

4.2. Atminties sunaudojimo eksperimentas

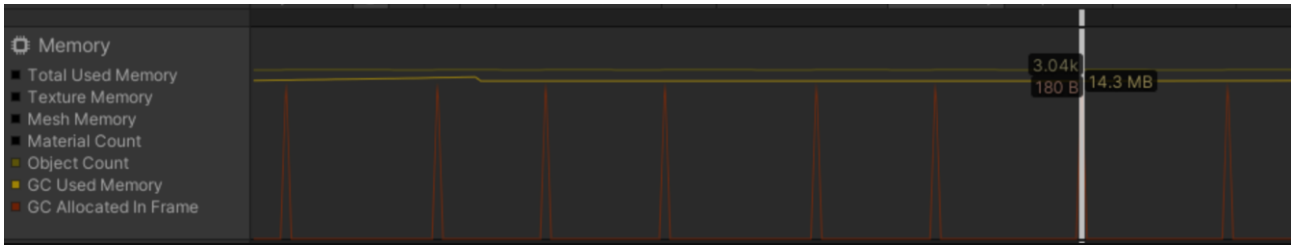
41-ajame paveiksle pateikiami žaidimo operatyviosios atminties matavimo rezultatai po patobulinimų.



41 pav. Operatyviosios atminties sunaudojimas po patobulinimų

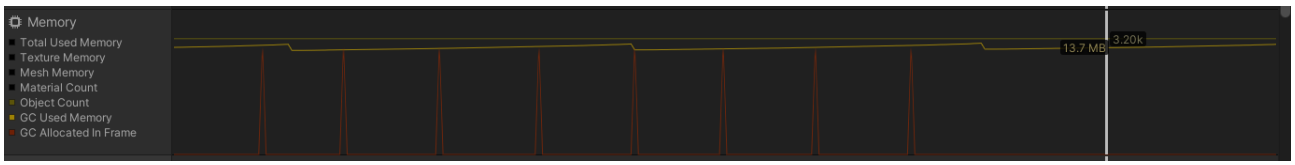
Po to, kai tekstūroms buvo uždėti dydžių limitai ir pritaikytas suspaudimo metodas, žaidimas naudojo 2,18 GB operatyviosios atminties. Daugiausiai atminties resursų vis dar naudojo tekstūros (apie 560 MB atminties). Šiek tiek daugiau atminties pradėjo naudoti garso klipai (apie 180 MB). Modeliai naudojo apie 60 MB atminties. Likusią atmintį naudojo pats „*Unity*“ redaktorius ir kiti įrankiai. Bendrą operatyviosios atminties sunaudojimą pavyko sumažinti 32 procentais – nuo 3,22 GB iki 2,18 GB.

Taip pat, buvo atliktas eksperimentas siekiant nustatyti objektų telkimo metodo naudojimo daromą įtaką atminčiai. Siekiant išvengti pašalinių faktorių įtakos, šiam eksperimentui atlikti buvo sukurta atskira testavimo scena, kurioje buvo imituojamas šaudymas ginklu. Buvo šaunama 100 kulkų, po vieną kulką kas 100 milisekundžių. Kulka praėjus sekundei po iššovimo buvo sunaikinama. 42-ajame paveiksle pateikiamas vaizdas iš „*Unity Profiler*“ apie atminties sunaudojimą nenaudojant objektų telkimo technikos kulkoms kurti. Matoma, kad kiekvieną kartą šaunant ar naikinant kulką, būdavo išskiriama 180 baitų šiukšlių rinkėjui (angl. *Garbage collector*), kuriais jis turėdavo pasirūpinti.



42 pav. Šiukšlių išskyrimas nenaudojant objektų telkimo metodo

43-ajame paveiksle pateikiamas vaizdas iš „*Unity Profiler*“ apie atminties sunaudojimą kulkų sukūrimui ir naikinimui naudojant objektų telkimo metodą. Naudojant šį metodą atminties šiukšlės išskiriamos tik pirmų kulkų iššovimo metu. Vėliau kulkoms skirta atmintis yra pakartotinai panaudojama ir atminties šiukšlės nėra išskiriamos.



43 pav. Šiukšlių išskyrimas naudojant objektų telkimo metodą

4.3. Sukompiliuoto žaidimo dydžio eksperimentas

44-ajame paveiksle pateikiami sukompiliuoto žaidimo dydžio rezultatai po patobulinimų.

44 pav. Žaidimo dydžio rezultatai po patobulinimų

Galima pastebėti, kad dėl tekstūrų suspaudimo metodo, gerokai sumažėjo jų užimamas dydis. Tekstūrų užimamas dydis sumažėjo net 87-iais procentais (nuo 1 GB iki 134,7 MB), garsų užimamas dydis sumažėjo 16-a procentų (nuo 112,1 MB iki 94,2 MB), o bendras sukompiliuoto žaidimo dydis sumažėjo 78-iais procentais (nuo 1,24 GB iki 271,9 MB).

Išvados

1. Atlikus žaidimų optimizacijos metodų, technikų ir geriausių praktikų analizę, buvo nuspręsta įgyvendinti objektų telkimo, nematomų objektų paslėpimo ir detalumo lygio metodus, nes pastebėta, kad šių metodų naudojimas gali padėti sumažinti operatyviosios atminties naudojimą bei pagerinti procesoriaus ir grafinės plokštės resursų našumą. Kuo procesoriaus ir grafinės plokštės našumas geresnis, tuo žaidimas yra geriau optimizuotas ir gali sugeneruoti daugiau kadrų per sekundę. Kuo daugiau sugeneruojama kadrų per sekundę, tuo žaidimas yra sklandesnis.
2. Tyrimo metu buvo nustatyta, kad iš pradžių sukurtas žaidimas neatitiko iškeltų našumo reikalavimų, todėl buvo nuspręsta įgyvendinti nematomų objektų paslėpimo metodą, nes šis metodas sumažina objektų kiekį kuriuos turi apdoroti grafikos plokštė. Įgyvendinus šį metodą, nepriklausomai nuo lygio dydžio žaidimas veikė bent 80-ies kadrų per sekundę ekrano atnaujinimo dažniu naudojant kompiuterį su reikalavimus atitinkančia aparatine įranga.
3. Eksperimentinėje dalyje atlikus našumo matavimus buvo pastebėta, kad kad tiriamam žaidimui nematomų objektų paslėpimo metodo naudojimas padidino vidutinį kadrų per sekundę kiekį 52 procentais mažo dydžio (sudarytame iš 10-ies kambarių) lygyje, vidutinio dydžio (sudarytame iš 25-ies kambarių) lygiuose – 144 procentais, o dideliuose lygiuose (sudarytame iš 50-ies kambarių) – 315 procentų. Taip pat, kadrų per sekundę kiekio rodikliai tapo gerokai mažiau priklausomi nuo lygio dydžio – žaidimas vidutiniškai siekė apie 84 kadrus per sekundę. Detalumo lygio metodo naudojimas įtakos vidutiniam kadrų per sekundę kiekiui nepadarė, todėl kad žaidimas vyksta siauruose kambariuose, kur žaidėjui visi kambaryje matomi objektai būna pakankamai arti ir išlieka aukšto detalumo lygio. Objektų telkimo metodas sumažino generuojamų atminties šiukšlių kiekį per 180 baitų kiekvieno šūvio iššovimo metu. Tekstūrų maksimalaus leidžiamo dydžio ribojimas sumažino operatyviosios atminties naudojimą 32 procentais. Sukompiliuoto žaidimo užimamą dydį pavyko sumažinti 78-iais procentais naudojant tekstūrų ir garso failų suspaudimo metodus.

Literatūros sąrašas

1. *Can you run it? Most popular PC Game Requirements* [interaktyvus]. 2023 [žiūrėta 2023-04-23]. Prieiga per: https://www.systemrequirementslab.com/cyri
2. *Understanding object pooling in Unity* [interaktyvus]. 2021 [žiūrėta 2023-04-23]. Prieiga per: https://unity.com/how-to/use-object-pooling-boost-performance-c-scripts-unity#understanding-object-pooling-unity
3. CHRISTOU, Ioannis T. ir EFREMIDIS, Sofoklis. *To pool or not to pool? revisiting an old pattern*. In arXiv.org [interaktyvus]. 2018 [žiūrėta 2023-04-23]. Prieiga per: https://arxiv.org/abs/1801.03763.
4. Unity. *Optimization tips for maximum performance – Part 1* [vaizdo medžiaga]. 2020 [žiūrėta 2023-04-23]. Prieiga per: https://www.youtube.com/watch?v=ZRDHEqy2uPI
5. *Managed memory in Unity* [interaktyvus]. 2021 [žiūrėta 2023-04-23]. Prieiga per: https://unity.com/how-to/use-object-pooling-boost-performance-c-scripts-unity#memory-allocation
6. KOULAXIDIS, Georgios. ir XINOGALOS, Stelios. *Improving Mobile Game Performance with Basic Optimization Techniques in Unity*. Modelling [interaktyvus]. 2022. Vol. 3, no. 2, p. 201–223 [žiūrėta 2023-04-23]. DOI 10.3390/modelling3020014.
7. *Unity – Manual: How occlusion culling works* [interaktyvus]. 2023 [žiūrėta 2023-05-06]. Prieiga per: https://docs.unity3d.com/2020.3/Documentation/Manual/OcclusionCulling.html
8. *Unity – Manual: Getting started with occlusion culling* [interaktyvus]. 2023 [žiūrėta 2023-05-06]. Prieiga per: https://docs.unity3d.com/Manual/occlusion-culling-getting-started.html
9. *Unity – Manual: Using occlusion culling with dynamic GameObjects* [interaktyvus]. 2023 [žiūrėta 2023-05-06]. Prieiga per: https://docs.unity3d.com/Manual/occlusion-culling-dynamic-gameobjects.html
10. *Unity – Manual: Occlusion Areas* [interaktyvus]. 2023 [žiūrėta 2023-05-06]. Prieiga per: https://docs.unity3d.com/Manual/class-OcclusionArea.html
11. *Unity – Manual: Occlusion Portals* [interaktyvus]. 2023 [žiūrėta 2023-05-06]. Prieiga per: https://docs.unity3d.com/Manual/class-OcclusionPortal.html
12. *Unity – Manual: Level of Detail (LOD) for meshes* [interaktyvus]. 2023 [žiūrėta 2023-05-06]. Prieiga per: https://docs.unity3d.com/Manual/LevelOfDetail.html
13. SACCO, Michael. *Optimize Unity Game Performance with Object Pooling: Best Practices, Benefits, and Techniques* [interaktyvus]. 2023 [žiūrėta 2023-05-06]. Prieiga per: https://www.occasoftware.com/blog/optimize-unity-game-performance-with-object-pooling-best-practices-benefits-and-techniques
14. *Unity – Manual: LOD Group* [interaktyvus]. 2023 [žiūrėta 2023-05-06]. Prieiga per: https://docs.unity3d.com/2020.3/Documentation/Manual/class-LODGroup.html
15. *Unity – Manual: Importing LOD Meshes* [interaktyvus]. 2023 [žiūrėta 2023-05-06]. Prieiga per: https://docs.unity3d.com/2020.3/Documentation/Manual/importing-lod-meshes.html
16. Unity Technologies. *Optimize your console and PC game performance* [interaktyvus]. 2021 [žiūrėta 2023-05-06]. Prieiga per: https://resources.unity.com/games/performance-optimization-e-book-console-pc
17. *Unity – Scripting API: MonoBehaviour* [interaktyvus]. 2023 [žiūrėta 2023-05-06]. Prieiga per: https://docs.unity3d.com/2020.3/Documentation/ScriptReference/MonoBehaviour.html
18. SIMONOV, Valentin. *10000 Update() calls* [interaktyvus]. 2015 [žiūrėta 2023-05-06]. Prieiga per: https://blog.unity.com/engine-platform/10000-update-calls

19. KROGH-JACOBSEN, Thomas. *Optimize your mobile game performance: Expert tips on graphics and assets* [interaktyvus]. 2021 [žiūrėta 2023-05-06]. Prieiga per: https://blog.unity.com/games/optimize-your-mobile-game-performance-expert-tips-on-graphics-and-assets
20. KROGH-JACOBSEN, Thomas. *Optimize your mobile game performance: Get expert tips on physics, UI, and audio settings* [interaktyvus]. 2021 [žiūrėta 2023-05-06]. Prieiga per: https://blog.unity.com/games/optimize-your-mobile-game-performance-get-expert-tips-on-physics-ui-and-audio-settings
21. *Optimization tips for Unity UI* [interaktyvus]. 2021 [žiūrėta 2023-05-06]. Prieiga per: https://unity.com/how-to/unity-ui-optimization-tips
22. *Steam Hardware & Software Survey: April 2023* [interaktyvus]. 2023 [žiūrėta 2023-05-13]. Prieiga per: https://store.steampowered.com/hwsurvey/Steam-Hardware-Software-Survey-Welcome-to-Steam
23. *Unity – Manual: Profiling your application* [interaktyvus]. 2023 [žiūrėta 2023-05-13]. Prieiga per: https://docs.unity3d.com/Manual/profiler-profiling-applications.html
24. Anamalous Underdog. *Build Report Tool* [interaktyvus]. 2023 [žiūrėta 2023-05-13]. Prieiga per: https://assetstore.unity.com/packages/tools/utilities/build-report-tool-8162#description
25. Frenchfaso Indie Apps. *InstantOC Dynamic Occlusion Culling* [interaktyvus]. 2016 [žiūrėta 2023-05-13]. Prieiga per: https://assetstore.unity.com/packages/tools/camera/instantoc-dynamic-occlusion-culling-lod-6391#description
26. *Unity - BuildOptions.CompressWithLz4* [interaktyvus]. 2023 [žiūrėta 2023-05-06]. Prieiga per: https://docs.unity3d.com/2020.3/Documentation/ScriptReference/BuildOptions.CompressWithLz4.html