**Kauno technologijos universitetas**

Informatikos fakultetas

# Dirbtinio intelekto metodų taikymas strateginiams žaidimams
# Application of Artificial Intelligence Methods for Strategy Games

T000M 257 Baigiamasis projektas (programinės įrangos įdiegimas)

Techninė dokumentacija

**Tadas Laurinaitis**

Projekto autorius

**Dr. Tomas Blažauskas**

Vadovas

**Kaunas, 2025**

**TURINYS**

# Contents

**Paveikslų sąrašas**

**Santrumpų ir terminų sąrašas**

PC (angl. *Non-playable character*) – kompiuterinio žaidimo veikėjas, kurį valdo kompiuteris.
AI (angl. *Artificial Intelligence*) – dirbtinis intelektas.
RTS (angl. *Real-time strategy*) – realiu laiku vykstanti strategija. Dažniausiai - žaidimo tipas.

# 1. PROJEKTO PARAIŠKA

## 1.1. Įvadas

Strateginiai kompiuteriniai žaidimai yra neatsiejama didelės grupės žmonių laisvalaikio dalis. Dirbtinis intelektas strateginiuose žaidimuose padeda sukurti gyviau reaguojančias aplinkas, prie žaidėjo prisitaikančias sistemas bei sudėtingesnius, labiau į tikrą gyvą žmogų panašius kompiuterio valdomus žaidėjus. Dirbtinio intelekto dėka, strateginiai žaidimai sukuria virtualius pasaulius, kurie duoda geresnę patirtį savo žaidėjams ir padeda labiau įsijausti į žaidimo pasaulį. Strateginiuose žaidimuose pačio žaidimo dirbtinio intelekto sudėtingumas žaidžia svarbią role žaidėjų įtraukime – jeigu žaidimas labai paprastas, jis greitai atsibos, o jeigu bus per daug sudėtingas, žaidėjui kils neigiami jausmai jį žaidžiant, ko pasekoje žaidėjas norės jį išjungti. Tinkamų dirbtinio intelekto metodų naudojimas kuriant strateginius žaidimus užtikrina, kad žaidimas bus tokio sunkumo, kad žaidėjui jis greitai neatsibostų ir pastoviai suteiktų įveikiamų iššūkių [1]. Šiuo metu beveik kiekvienas strateginis žaidimas naudoja tam tikrus dirbtinio intelekto metodus, kaip pavyzdžiui algoritmus priešininkų ar sąjungininkų elgsenos modeliavimui, kelio radimui, duomenų gavybai, procedūriniam turinio generavimui ir žaidėjo patirties modeliavimui.

Bėda ta, kad daugelis sukurtų algoritmų yra apsaugoti juos kuriančių kompanijų ar korporacijų ir nėra viešai prieinami.

Šio projekto tikslas yra ištirti esamus dirbtinio intelekto algoritmus ir metodikas naudojamas kompiuterinių žaidimų strategijai realizuoti, pasiūlyti patobulinimus šiems algoritmams ir metodikoms, ir įvertinti jų efektyvumą sukuriant algoritmus ir metodikas realizuojantį žaidimą. Projekto metu bus kuriamas strateginis žaidimas, realizuojantis kelis vėliau pasirinktus dirbtinio intelekto metodus. Kuriant šį žaidimą ir vykdant tyrimą, naudojami dirbtinio intelekto metodai bus patobulinami ir pasiūlomos alternatyvos jeigu šių dirbtinio intelekto metodų naudojimas nebus optimalus.

Projekto pasiūlymas suformuotas pagal Dr. Tomo Blažausko užsakymą. Žaidimas bus kuriamas kaip magistrinio darbo dalis.

Projektas planuojamas įvykdyti per metus su puse, iki 2025-05-01. Projektas apima žaidimą, tyrimą ir projekto ataskaitą. Projektui įgyvendinti naudojama naujausios technologijos, įranga bei sprendimai.

Projekto kaina – 35000 eurų.

## 1.2. Poreikis

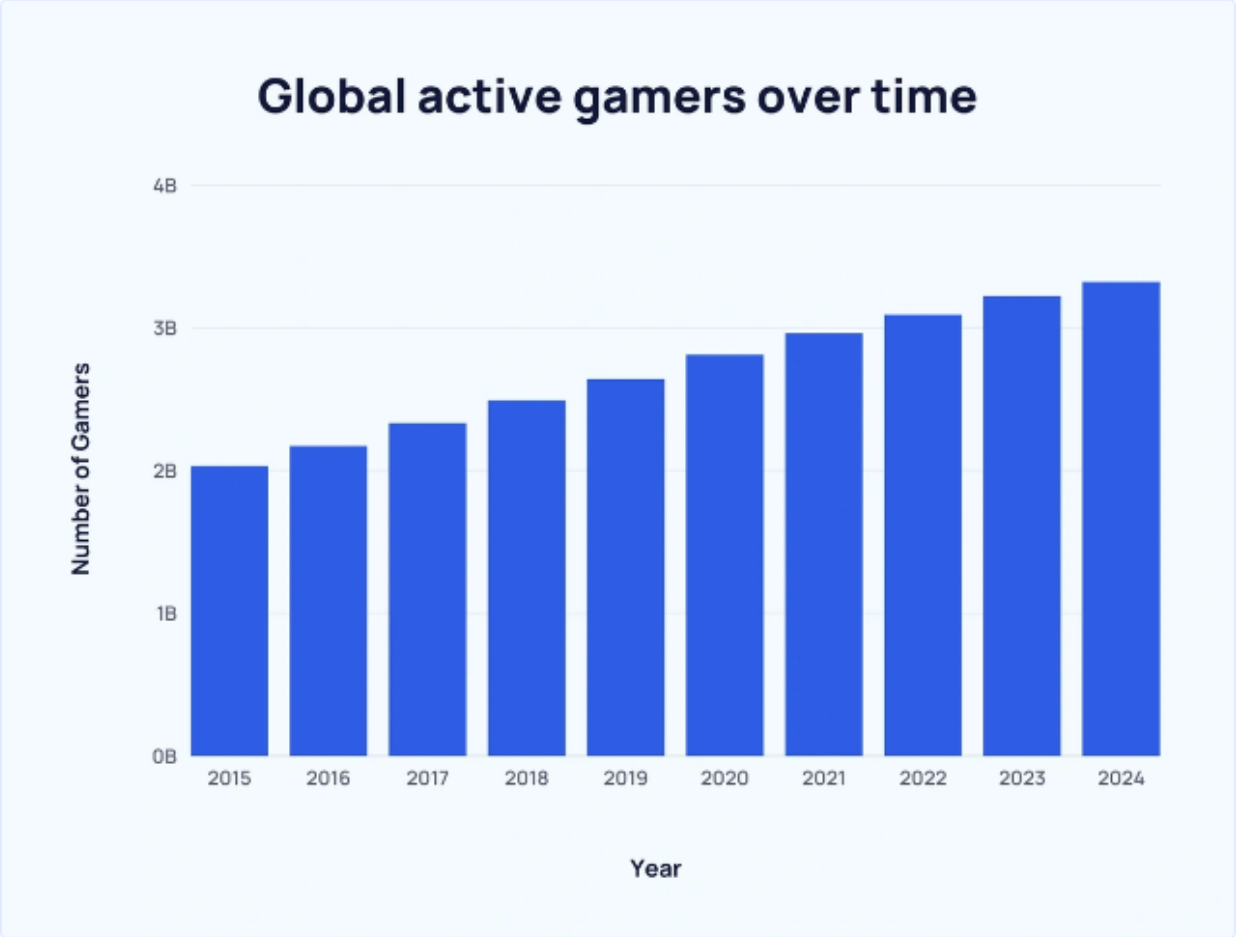
### 1.2.1. Projekto naudotojai ir klientai

Pagrindiniai žaidimo naudotojai yra jauni ir vidutinio amžiaus žmonės iš plačiosios visuomenės, praleidžiantys dalį savo laisvalaikio žaidžiant kompiuterinius žaidimus, naudojantys kompiuterinius žaidimus kaip veiklą užsidirbti ir mėgstantys strateginius kompiuterinius žaidimus. Taip pat tai žmonės, žaidžiantys žaidimus naudojantis asmeniniu kompiuteriu, o ne konsolėmis.

### 1.2.2. Naudotojų rolės ir tikslai

Vis didesnė dalis žmonių pradeda žaisti strateginius kompiuterinius žaidimus. Tačiau dalis žaidėjų pabandę pažaisti vieną ar kitą strateginį žaidimą, greitai nustoja. Taip yra dėl to, jog ne maža dalis žaidimų turi neįdomius ir iššūkio nesukeliančius kompiuterio bei dirbtinio intelekto valdomus žaidėjus, ko pasekoje žaidimai greitai nusibosta, yra neįtraukiantys ir neatitinka žaidėjų reikalavimų. Esami ilgamečiai strateginių žaidimų žaidėjai taip pat pastebi, kad dirbtinis intelektas kai kuriuose žaidimuose yra nuobodus, niekuo neišsiskiriantis ir nekeliantis entuziazmo. Tačiau yra ir kita monetos pusė – dalyje žaidimų dirbtinis intelektas atrodo kaip superkompiuteris, sekantis kiekvieną tikro žaidėjo veiksmą, gaunantis nesąžiningus bonusus ir sukeliantis žaidėjui atmetimo reakciją, kadangi niekas nenori žaisti žaidimo, kuris vietoj to kad sukeltų įdomų iššūkį, yra tiesiog atvirai nesąžiningas. Nors šios problemos yra realios, dalis žmonių net nesusimąsto apie gero dirbtinio intelekto svarbą kompiuteriniuose žaidimuose ir nesupranta, jog kad žaidimas būtų tikrai vertas laiko ir dėmesio, jis turi turėti ne tik gražią grafinę pusę [2]. Dirbtinis intelektas žaidime turi suteikti žaidėjui įveikiamą iššūkį, papildyti žaidime esančias aplinkas ir suteikti žaidėjui norą sekantį kartą sugrįžti. Dalis žaidimų šiuos aspektus įgyvendina sėkmingai ir sukuria žaidėjams nepamirštamus įspūdžius, tačiau dar dažniau kuriami žaidimai gaunasi pilki, neįsimintini bei nesukeliantys noro prie jų grįžti [3]. Prie šios problemos prisideda ir neteisingų dirbtinio intelekto metodų pasirinkimas, arba neteisingas teisingai pasirinktų dirbtinio intelekto metodų įgyvendinimas. Šios problemos taip pat nepadeda spręsti faktas, kad didžioji dalis dirbtinio intelekto metodų pavyzdžių, naudojamų vienuose ar kituose populiariuose ir sėkminguose strateginiuose kompiuteriniuose žaidimuose nėra laisvai prieinami plačiajai visuomenei, ko pasekoje mažoms, iš vieno ar kelių žmonių sudarytoms komandoms, norinčioms kurti šiuos žaidimus, yra sunku teisingai įgyvendinti dirbtinio intelekto dalį.

### 1.2.3. Rinkos tyrimas




**pav. 1 Kompiuterinių žaidimų žaidėjų kiekio pokytis nuo 2015 [4].**

Žaidimas skirtas žmonėms, kurie žaidžia kompiuterinius žaidimus. Žmonių, kurie žaidžia kompiuterinius žaidimus šiuo metu yra apie 3.09 milijardo [4]. Vien per paskutinius septynis metus, žmonių, žaidžiančių kompiuterinius žaidimus kiekis išaugo 1 milijardu [4]. Šių žmonių pasiskirstymas pasaulyje: Azija ir Ramiojo vandenyno regionas – 1.47 milijardo, Europa – 386 milijonai, Vidurio rytai ir Afrika – 377 milijonai, Pietų Amerika – 266 milijonai ir Šiaurės Amerika – 210 milijonų [5]. Matoma, kad didžiausias žaidėjų kiekis yra Azijoje ir Ramiojo vandenyno šalyse, kadangi iš ten yra kilusios tokios kompiuterinių žaidimų milžinės kaip „Nintendo“, „Sega“, „Sony“ ir tai yra tapę didele jų kultūros dalimi.

Strateginių žaidimų pasirinkimas labai platus, o žaidėjai dažniausiai renkasi žaidimus žaisti naudojantis žaidimų platformomis, tokiomis kaip „*Steam*“ ar „*Epic Games*“. Šioje skiltyje bus remiamasi „*Steam*“ platformos duomenimis, kadangi jie yra atvirai prieinami visiems *SteamCharts* tinklalapyje.

Šiuo metu šie strateginiai žaidimai, naudojantys įmantrias dirbtinio intelekto sistemas yra prieinami „Steam“ platformoje ir yra patys populiariausi:

Žaidimas „*Sid Meier's Civilization VI*“ šiuo metu turi 56751 žaidėją, o visų laikų aukščiausias pasiektas žaidėjų kiekis – 162314 žaidėjų [6].

Žaidimas „*Hearts of Iron IV*“ šiuo metu turi 37281 žaidėją, o visų laikų aukščiausias pasiektas žaidėjų kiekis – 70836 [6].

Strateginiai žaidimai rinka sudaro didelę dalį visos kompiuterinių žaidimų rinkos. Strateginių žaidimų rinkos dydis visame pasaulyje yra 24.6 milijardai dolerių 2022, o šios rinkos augimas per metus yra 8.84 procento [7].

### 1.2.4. Informacija apie klientus

Klientai yra dviejų tipų:

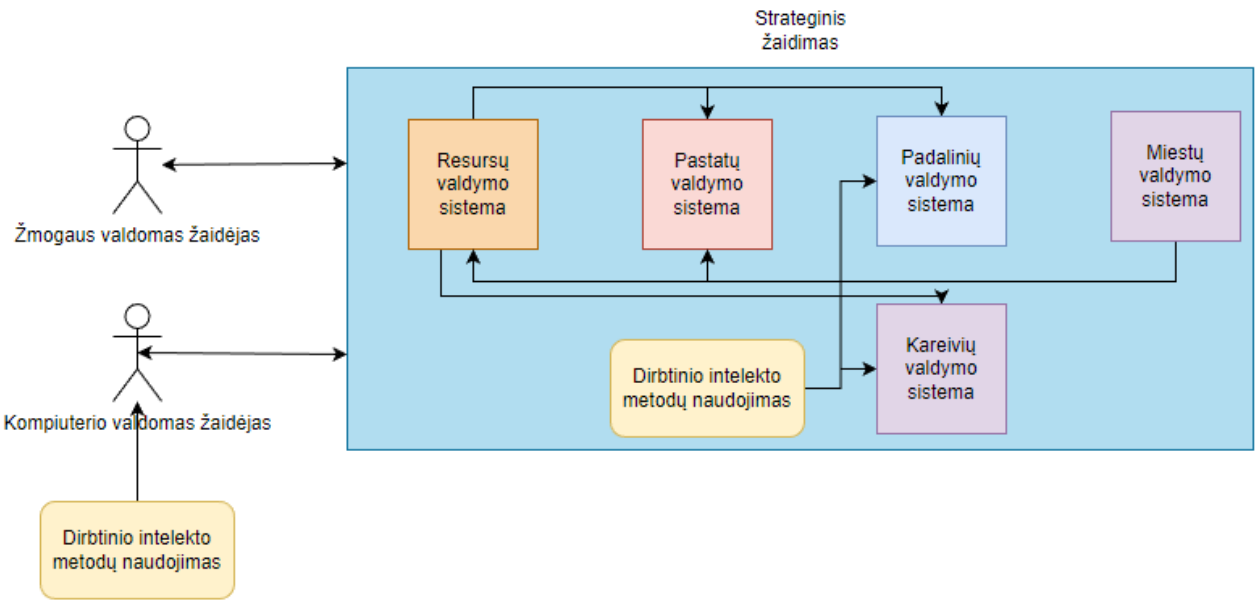
1. Naudotojai, kurie naudos sukurtą žaidimą. Užsakovo užsakymu, bus sukurtas žaidimas, kurį bus galima platinti arba naudoti kaip prototipą tolesniam platinimui.

2. Kompiuterinius žaidimus kuriančios komandos, kompanijos ar korporacijos, kurios pasinaudos žaidimo kūrimo metu atlikto tyrimo rezultatais ir naudos tam tikrus, ištirtus dirbtinio intelekto metodus.

## 1.3. Pasiūlymas

### 1.3.1. Produkto ar paslaugos apibūdinimas

Kadangi, kaip jau buvo minėta praeitame poskyryje, strateginių žaidimų rinka yra didelė ir auga dideliu tempu, bus kuriamas realaus laiko strateginis (*RTS*) žaidimas. Žaidėjas valdys savo pasirinktą civilizaciją, rinks resursus, apmokys padalinius ir kareivius, statys miestą ir kariaus su kompiuterio valdomais priešininkais. Pagrindinis žaidimo tikslas – išgyventi kuo ilgiau ir nugalėti priešus.




**pav. 2 Kuriamo strateginio žaidimo sistemų tarpusavio sąveika aukštame lygmenyje.**

Kiekvienas žaidimo aspektas bus išskirstytas į atskiras, už vieną dalyką atsakingas sistemas, kurios sąveikaus tarpusavyje (žr. pav. 2).

Žaidėjas turės keturis pagrindinius resursų tipus – medį, mineralus, auksą ir mokslą.

Žaidėjas galės įkurti savo miestą, jame pasirinkti įvairius pastatus, juos statyti, valdyti ir gerinti. Pastatai bus pasirenkami iš pastatų pasirinkimo meniu ir jų statymas bei gerinimas kainuos atitinkamą kiekį resursų. Figūruos keturi pagrindiniai pastatų tipai – resursų išgavimo pastatai, mokslo pastatai, kareivių ir padalinių rengimo pastatai ir gynybiniai įtvirtinimai.

Žaidėjas pasistatęs atitinkamus pastatus, galės apmokyti įvairius padalinius ir kareivius, kurių funkcija bus rinkti resursus, ginti miestą arba pulti priešininkus. Kiekvienas padalinys ir kareivis kainuos atitinkamą kiekį resursų, o naudojantis mokslo resursu, padaliniai ir kareiviai bus pagerinami.

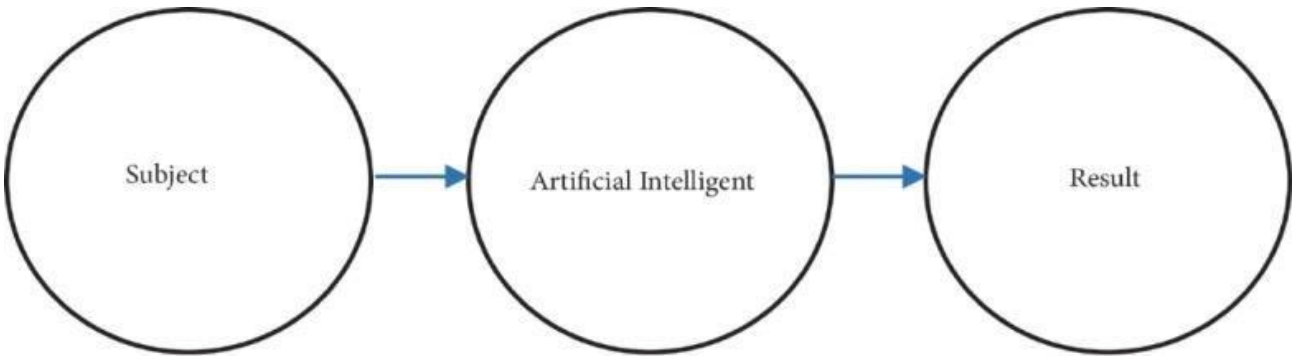
Žaidėjas galės duoti nurodymus savo padaliniams ir kareiviams. Padaliniai galės rinkti resursus, pagreitinti pastatų statymą, bei užsiimti mokslo resurso gaminimu atitinkamuose mokslo pastatuose.

Padaliniai turės automatinio valdymo funkciją, kuri veiks dirbtinio intelekto metodų pagrindu. Automatinio valdymo funkcija leis padaliniams atlikti veiksmus automatiškai, be žaidėjo įsikišimo, o veiksmų parinkimas bus nusprendžiamas pačio dirbtinio intelekto pagal esamą situaciją.

Priešininkai bus valdomi kompiuterio. Kompiuterio priešininkų valdymui bus panaudojami keli skirtingi dirbtinio intelekto metodai, kuriais naudojantis bus duodamos strategijos ir atliekamas skirtingų dirbtinio intelekto metodų palyginimas.

### 1.3.2. Sistemos kontekstas

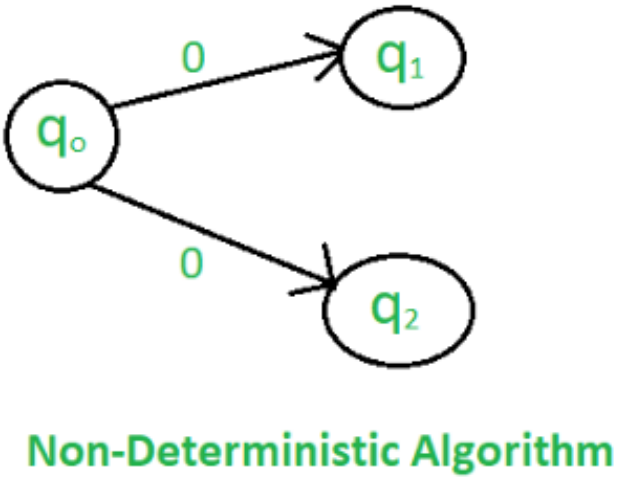
Žaidimas naudos dirbtinio intelekto metodus: kompiuterio valdomiems priešininkams valdyti ir padaliniams ir kareiviams, esantiems automatizuotame rėžime valdymui. Siekiant atrasti geriausius dirbtinio intelekto metodus, tinkančius šiam žaidimui, bus ekspermentuojama tiek su deterministiniais, tiek su nedeterministiniais dirbtinio intelekto metodais.




**pav. 3 Deterministinio dirbtinio intelekto metodo šablonas [8]**

Kadangi deterministiniai dirbtinio intelekto metodai yra greiti ir lengvai implementuojami, o jų veikimas yra konkretus ir nuspėjamas, jie bus pritaikomi padalinių ir kareivių automatizuoto rėžimo implementacijai [9]. Bus panaudoti keli skirtingi šios rūšies metodai, palyginamas jų veikimo optimalumas ir teisingumas. Taip pat bus bandoma kelių šios rūšies metodų kombinacija ir matuojamas šios kombinacijos optimalumas žaidimui.

Deterministinių dirbtinio intelekto metodų įgyvendinimui bus pasitelkiama C# programavimo kalba ir jos bibliotekos (priklausomai nuo metodo).




**pav. 4 Nedeterministinio dirbtinio intelekto metodo šablonas [10]**

Kadangi priešingai nei deterministiniuose dirbtinio intelekto metoduose, nedeterministiniuose metoduose yra tam tikras neaiškumo laipsnis, elgesys ir veikimas nėra nuspėjami, o patys metodai gali mokytis ir ekstrapoliuoti patys, šie metodai bus naudojami kompiuterio valdomiems priešininkams valdyti [11]. Bus panaudojami keli šios rūšies metodai, o jais naudojantis kompiuterio valdomiems priešininkams bus suteikiamas savarankiškumas, skirtingos strategijos ir nuo situacijos priklausanti elgsena.

Nedeterministinių dirbtinio intelekto metodų įgyvendinimui bus pasitelkiama C# programavimo kalba.

Žaidimas žaidėjams bus pasiekiamas „*Scratch.io*“ ir „*Steam*“ platformose, kuriose standartiniu būdų, žaidėjas jį galės parsisiųsti. Prototipas bus siūlomas nemokamai. Interneto ryšys žaidimui bus reikalingas tik parsisiuntimo metu.

### 1.3.3. Techninės galimybės

Reikalinga įranga žaidimui sukurti ir dirbtinio intelekto metodų analizei atlikti – kompiuteris su *Windows* 10+ 64 bitų operacine sistema, procesoriumi palaikančiu X64 architektūrą ir *SSE2* instrukcijų rinkinį, vaizdo plokštė palaikanti *DX11*, kadangi tokie yra *Unity Editor* sistemos reikalavimai [12].

Ekspermentai su dirbtinio intelekto metodais bus atliekami bandymo ir lyginimo būdu, išbandant įvairius galimus dirbtinio intelekto metodo implementacijos būdus ir dirbtinio intelekto metodų kombinacijas.

Duomenys nedeterministiniams dirbtinio intelekto metodams, bus renkami žaidimo kūrimo ir testavimo metu.

Baigiamojo projekto komisijai bus rodoma visas žaidimas ir jo veikimas, bei skirtingų dirbtinio intelekto metodų veikimas žaidime.

### 1.3.4. Rizika ir apribojimai

Rizikos:

- Galimi programinės įrangos – *Unity Editor* gedimai, dėl kurių įmanomas informacijos ar žaidimo kūrimo dalies progreso praradimas [13].
- Įgyvendinti dirbtinio intelekto metodai nustoja veikti korektiškai, ko pasekoje tęsti žaidimą yra neįmanoma.

Apribojimai:

- Žaidimo kūrimo ir dirbtinio intelekto metodų analizės metu būtinas interneto ryšys.
- Norint testuoti sukurtą žaidimą, būtinas kompiuteris su *Windows* 10+ operacine sistema.
- Žaidimas turi veikti 60 kadrų per sekundę rėžimu bent 1080p rezoliucijoje ir neturi turėti trukdžių.

### 1.3.5. Projekto įgyvendinimo planai ir kokybės vertinimas

Projekto metu bus sukurtas strateginis kompiuterinis žaidimas, atlikta dirbtinio intelekto metodų tinkamumo žaidimui analizė ir parengta ataskaita. Žaidimas bus kuriamas naudojantis Unity žaidimų variklį kartu su C# ir programavimo kalba. Dirbtinio intelekto dalis bus atliekama naudojantis C# programavimo kalba pačiame *Unity* žaidimų variklyje.

Projekto veiklos ir atitinka Magistrinio projekto įgyvendinimo akademinį grafiką.

Pabaigus kurti žaidimą, bus svarbu įvertinti jo kokybę. Šiam tikslui iškelti vertinimo kriterijai, pavaizduoti 1 lentelėje.

1 lentelė. Sistemos vertinimo kriterijai

| Nr. | Kriterijus | Pagrindimas |
|---|---|---|
| 1 | Greitaveika | Žaidimas veikia 60 kadrų per sekundę 1080p rezoliucijoje be strigimo. |
| 2 | Interaktyvumas | Žaidimas reaguoja į žaidėjo veiksmus ir duoda grįžtamąjį ryšį. |
| 3 | Dirbtinio intelekto panaudojimo žaidime optimalumas | Kompiuterio valdomi priešininkai sugeba savarankiškai žaisti žaidimą, naudoja skirtingas strategijas ir elgiasi adekvačiai. |
| 4 | Priklausomumas kitos programinės įrangos | Sistema veikia kompiuteriuose su Windows 10+ operacinėmis sistemomis. |

Projektą įgyvendinantis personalas:

- Vienas programuotojas (Tadas Laurinaitis), turintis Programų sistemų bakalauro laipsnį. Su vadovo pagalba aprašo, suprojektuoja, ištestuoja ir realizuoja žaidimą. Šiam asmeniui priklausys sukurto produkto autorinės teisės.
- Vadovas (Tomas Blažauskas), turintis daktaro laipsnį informacinių technologijų srityje. Vadovauja kūrimo procesui, padeda studentui su iškylančiomis problemomis ir klausimais.

Turimi resursai (patalpos, techninė ir programinė įranga, biudžetas). Gal reikės vykti į komandiruotę, ar įsigyti tam tikrus prietaisus ar programinę įrangą. Išlaidų pagrindimas.

Projekto biudžeto skaičiavimas pateiktas 2 lentelėje.

**2lentelė.** Projekto biudžetas

| **Išlaidos** | **Vienetas** | **Vienetų skaičius** | **Vieneto kaina (įskaitant mokesčius, Eur** | **Viso, Eur** |
|---|---|---|---|---|
| 1. Žmonių  ištekliai | | | | |
| Programuotojas | Mėnesis | 18 * 0.2 etato | 3450 | 12420 |
| Vadovas | Mėnesis | 18 * 0.1 etato | 3500 | 6300 |
| *Iš viso žmonių išteklių* | | | | 18720 |
| 2. Įranga ir prekės | | | | |
| Kompiuteris | Vienetas | 1 | 1500 | 1500 |
| Periferinė įranga kompiuteriui | Vienetas | 1 | 700 | 700 |
| *Iš viso Įranga ir prekės* | | | | 2200 |
| 3. Biuro išlaikymas | Mėnesis | 18 | 560 | 10080 |
| Elektros, interneto, šildymo,telefono,nuomos išlaidos | Vienetas | 18 | 200 | 3600 |
| *Iš viso biuro išlaikymas* | | | | 13680 |
| 4. Programinė įranga | | | | |
| Visual Studio 2022 Professional | Vienetas | 1 | 89 | 89 |
| *Iš viso programinė įranga* | | | | 89 |
| 5. Viso tiesioginiai projekto kaštai | | | | 34689 |

## 1.4. Nauda

Projekto laukiami rezultatai:

Strateginis žaidimas leis žaidėjams praleisti bent 100 valandų laisvalaikio žaidžiant.

Bent 3 žaidimo kūrimo ir dirbtinio intelekto metodų tyrimo ir analizės metu ištirti metodai, jų implementacijos, pastabos, patobulinimai ir pritaikymo būdai bus atvirai prieinami naudoti kitiems žaidimų kūrėjams ar kompanijoms.

## 1.5. Konkurencija ir alternatyvos

Konkurentų analizė ir alternatyvos neaktualios, kadangi pagrindinis projekto tikslas yra vykdyti dirbtinio intelekto metodų, naudojamų strateginiuose žaidimuose analizę, juos implementuoti strateginiame žaidime ir padaryti juos viešai prieinamus kitiems žaidimų kūrėjams ar kompanijoms.

**3 lentelė.** Alternatyvos

| **Palyginimo kriterijus** | **Mūsų žaidimas** | **Sid Meier‘s Civilization V** | **Northgard** |
|---|---|---|---|
| Realaus laiko strategija | Taip | Taip | Taip |

| | | | |
|---|---|---|---|
| Civilizacijos/žaidėjo pasirinkimas | Taip | Taip | Ne |
| Padalinių/kareivių automatizavimas | Taip | Ne | Ne |
| Skirtingais dirbtinio intelekto metodais įgyvendinti kompiuterio valdomi žaidėjai | Taip | Ne | Ne |
| Kaina | Nemokamas | 30€ | 26.99€ |

## 1.6. Santrauka

**Vertėmis grįstas pasiūlymas:** Dirbtinio intelekto metodų taikymas strateginiams žaidimams

Strateginių žaidimų žaidėjams ir strateginių žaidimų kūrėjų komandoms ir įmonėms, kuriems įdomus ar komerciškai naudingas dirbtinio intelekto metodų naudojimas strateginiuose žaidimuose. Kurie nori linksmai praleisti laisvalaikį arba sukurti kompiuterinį strateginį žaidimą, kuris naudotų dirbtinio intelekto metodus, kurių implementacijos nėra viešai prieinamos, dėl žaidimo kūrimo kompanijų ir korporacijų, kurios nepadaro savo kodo atviro.

Mūsų siūlomas sprendimas: laisvai prieinamas atviro kodo strateginis žaidimas ir jame panaudotų dirbtinio intelekto metodų analizės rezultatas su alternatyvomis.

Suteikiamas įtraukiantis strateginis žaidimas, kuris žaidėjui leis jį peržaisti bent kelis kartus, dėl jame įgyvendinto dirbtinio intelekto. Taip pat suteikiama panaudotų dirbtinio intelekto metodų analizė, leisianti žaidimus kuriančioms komandoms ar įmonėms panaudoti šiuos sprendimus jų pačių kuriamuose žaidimuose.

# 2. PROJEKTAVIMO METODOLOGIJOS IR TECHNOLOGIJŲ ANALIZĖ

## 2.1. Įvadas

Projektas – Dirbtinio intelekto metodų taikymas strateginiams žaidimams.

Dirbtinis intelektas padeda sukurti labiau reaguojančius, prisitaikančius ir sudėtingesnius strateginius žaidimus. Dirbtinio intelekto dėka, strateginiai žaidimai sukuria geresnę patirtį savo žaidėjams kurdami gyvenimiškus situacinius pokyčius žaidimo progrese bei suteikdami jaudulio jausmą [1]. Didėjantis strateginių žaidimų dirbtinio intelekto sudėtingumas užtikrina, kad žaidėjai labiau įsijaus į patį žaidimą, o pats žaidimas jiems greitai neatsibos. Šiuo metu yra daug strateginių žaidimų, kuriuose taikomi įvairūs algoritmai priešininkų ar sąjungininkų elgsenos modeliavimui, kelio radimui, duomenų gavybai, procedūriniam turinio generavimui ir žaidėjo patirties modeliavimui. Bėda ta, kad daugelis sukurtų algoritmų yra apsaugoti juos kuriančių kompanijų ar korporacijų ir nėra viešai prieinami.

Raktiniai žodžiai – Dirbtinis intelektas, strateginius žaidimus, strateginiai žaidimai, žaidėjams, dirbtinio intelekto sudėtingumas, modeliavimui, algoritmų.

## 2.2. Tikslas

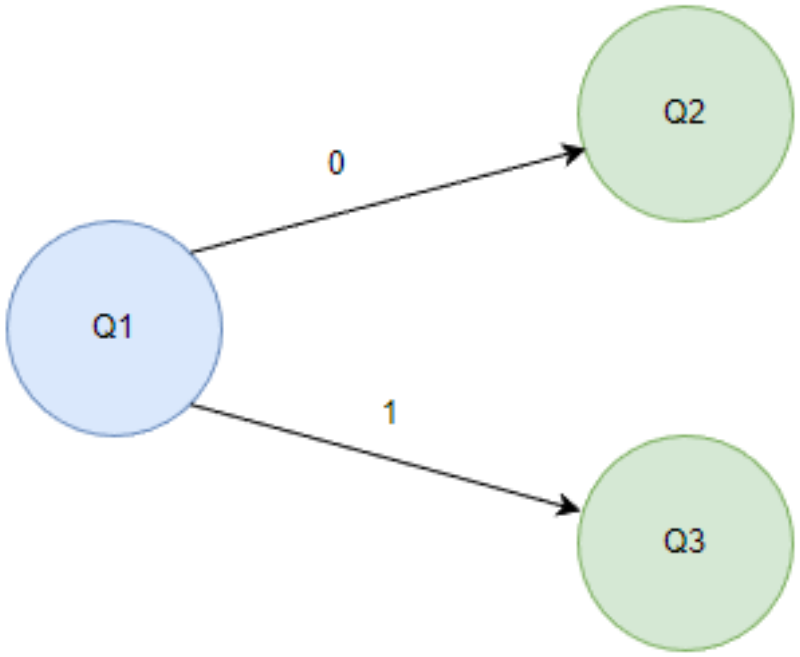
Šio darbo tikslas yra ištirti esamus dirbtinio intelekto algoritmus ir metodikas naudojamas kompiuterinių žaidimų strategijai realizuoti, pasiūlyti patobulinimus šiems algoritmams ir metodikoms ir įvertinti jų efektyvumą sukuriant algoritmus ir metodikas realizuojantį žaidimą.

## 2.3. Srities apžvalga ir egzistuojantys sprendimai

Strateginiai žaidimai yra ideali aplinka dirbtinio intelekto galimybių tyrinėjimui, kadangi bendrai, žaidimai suteikia uždarą aplinką su tam tikromis, kiekvienam žaidimui specifinėmis taisyklėmis, kurioje gali būti išbandoma ir įvertinama plati aibė problemų sprendimų būdų ir technikų, prieš pritaikant šiuos būdus ir technikas sprendžiant realaus pasaulio problemas. Pirmas dirbtinio intelekto pritaikymas žaidime buvo 1951 metais sukurtame, matematiniame strateginiame žaidime „*Nim*“ [2], kurio tikslas buvo dviems žaidėjams vienam paskui kitą nuiminėti daiktus iš dviejų krūvų. Šiame žaidime dirbtinis intelektas sugebėdavo reguliariai laimėti prieš žmones. Tais pačiais metais Mančesterio universitete buvo sukurti du kompiuteriniai žaidimai, kurie taip pat naudojo dirbtinį intelektą – šaškės ir šachmatai. Per ateinančius dešimtmečius dirbtinio intelekto algoritmai naudojami šiuose žaidimuose buvo tobulinami, o šio tobulinimo kulminacija buvo pasiekta 1997 metais, kai *IBM Deep Blue* kompiuteris nugalėjo tuometinį šachmatų „*Grandmaster*“ titulą laikantį Garry Kasparov [3]. Dar praeito amžiaus aštuntajame dešimtmetyje dirbtinis intelektas tapo integralia kompiuterinių žaidimų dizaino dalimi. 1978 metais sukurtas „*Space Invaders*“ demonstravo vis sunkėjančius lygius su skirtingais judėjimo modeliais, priklausančiais nuo žaidėjo judėjimo, naudojantis elementariu dirbtiniu intelektu saugomų judesių modelių pavidalu [4]. Nuo to laiko dirbtinis intelektas buvo bandomas pritaikyti daugumos žanrų žaidimams. Dirbtinis intelektas sporto žaidimuose pasirodė dar nuo praeito amžiaus devinto dešimtmečio, išleidus tokius žaidimus, kaip „*Earl Weaver Baseball*“, „*Madden Football*“ ir „*Tony La Russa Baseball*“, kuriuose priešininkai bandė atkartoti realių trenerių ir vadovų vadovavimo stilius ir naudojamas taktikas, o vėlesniuose sporto žaidimuose netgi atsirado galimybės sukurti dirbtinio intelekto

priešininkus naudojant žaidėjo nustatytus kintamuosius. Formalūs dirbtinio intelekto įrankiai, tokie kaip baigtinių būsenų mašinos, buvo pradedami naudoti dar 1990-taisiais įvairiuose, ypač naujuose žaidimų žanruose. Dirbtinis intelektas buvo pradedamas naudoti realaus laiko strateginiuose žaidimuose kelio radimo problemoms spręsti, atlikti realaus laiko sprendimus ar netgi planuoti kompiuterinio priešininko ekonomiką. Tačiau pirmųjų žaidimų dirbtinis intelektas nebuvo tobulas – pvz. 1990 Žaidimo „*Herzog Zwei*“ kelio radimas buvo praktiškai neveiksnus ir naudojo paprastą trijų baigtinių būsenų mašiną padalinių kontrolei, o 1992 žaidimo „*Dune II*“ dirbtinis intelektas sukčiaudavo, kad gautų nesąžiningą persvarą prieš žaidėją. Laikui bėgant, o dirbtinio intelekto metodams ir algoritmams tampant labiau rafinuotiems, didžioji dalis šių problemų buvo išspręsta, o vėlesni žaidimai pradėjo naudoti „*bottom-up*“ (liet. Iš apačios į viršų) dirbtinio intelekto metodus, kurie sugeba įvertinti žaidėjo veiksmus ir pagal juos nusprendžia tolesnius veiksmus [5]. Šiuo metu, kompiuteriniuose žaidimuose, o tuo pačiu ir strateginiuose žaidimuose naudojamas dirbtinis intelektas dažniausiai yra skirstomas į dvi rūšis – deterministinį (angl. *Deterministic*) ir nedeterministinį (angl. *Nondeterministic*).

### 2.3.1. Deterministinis dirbtinis intelektas




**pav. 5 Deterministinio dirbtinio intelekto veikimo principas**

Deterministinio dirbtinio intelekto elgesys ir veikimas yra konkretus ir nuspėjamas, nėra neaiškumo. Deterministiniai dirbtinio intelekto metodai yra kompiuterinių žaidimų duona. Šie algoritmai yra nuspėjami, greiti, lengvai implementuojami, lengvai suprantami ir lengvai ištestuojami. Deterministiniai metodai nors ir naudingi, tačiau ant žaidimo kūrėjo užkrauna naštą – žaidimo kūrėjas turi žinoti, numatyti ir implementuoti visus įmanomus veiksmus. Taip pat, deterministiniai metodai neapima mokymosi ir tobulėjimo, todėl žaidėjui šiek tiek pažaidus žaidimą, jie tampa nuspėjami, ko pasekoje sutrumpėja žaidimo gyvavimo ciklas. Paprastas deterministinio intelekto pavyzdys yra persekiojimo algoritmas. Parašytas algoritmas nurodytų personažui, kad jam reikia judėti link tikslinio taško x ir y koordinačių ašimis, kol personažo x

ir y koordinatės sutaps su tiksline vieta [6]. 1 pav. matoma paprasta deterministinio dirbtinio intelekto reprezentacija, kur parodoma, jog iš Q1 galimi perėjimai į Q2 ir Q3, priklausomai nuo to ar Q1 reikšmė yra 0 ar 1.

### 2.3.2. Nedeterministinis dirbtinis intelektas

Priešingai nei deterministiniuose dirbtinio intelekto metoduose, nedeterministiniuose metoduose yra tam tikras neaiškumo laipsnis, o elgesys ir veikimas yra nenuspėjamas (nenuspėjamumo laipsnis labai priklauso nuo panaudoto dirbtinio intelekto metodo ir jo supratimo lygio). Nedeterministiniai metodai taip pat gali mokytis ir ekstrapoliuoti patys, skatinti atsirandantį naują elgesį, arba elgesį, kuris atsiranda be aiškių nurodymų. Žaidimų kūrėjai tradiciškai žiūrėdavo į nedeterministinius metodus kaip į nepatikimus, nors paskutiniais metais šis požiūris pradėjo keistis. Panaudojus nedeterministinį metodą jį sunku ištestuoti ir užtikrinti, jog neatsiras nenorima elgsena, todėl esant trumpam žaidimo kūrimo ciklui, sunku garantuoti jog žaidimas bus pilnai ištestuotas ir atitiks kokybės standartus, kurių tikisi potencialūs žaidėjai. Puikus nedeterministinio metodo pavyzdys būtų kompiuterio valdomo priešininko mokymasis ir prisitaikymas prie žaidėjo kovos taktikų, o tokį prisitaikymą galima būtų pasiekti naudojantis neuroniniu tinklu, Bayeso technika arba genetiniu algoritmu [7].

### 2.3.3. Nusistovėjęs dirbtinis intelektas žaidimuose

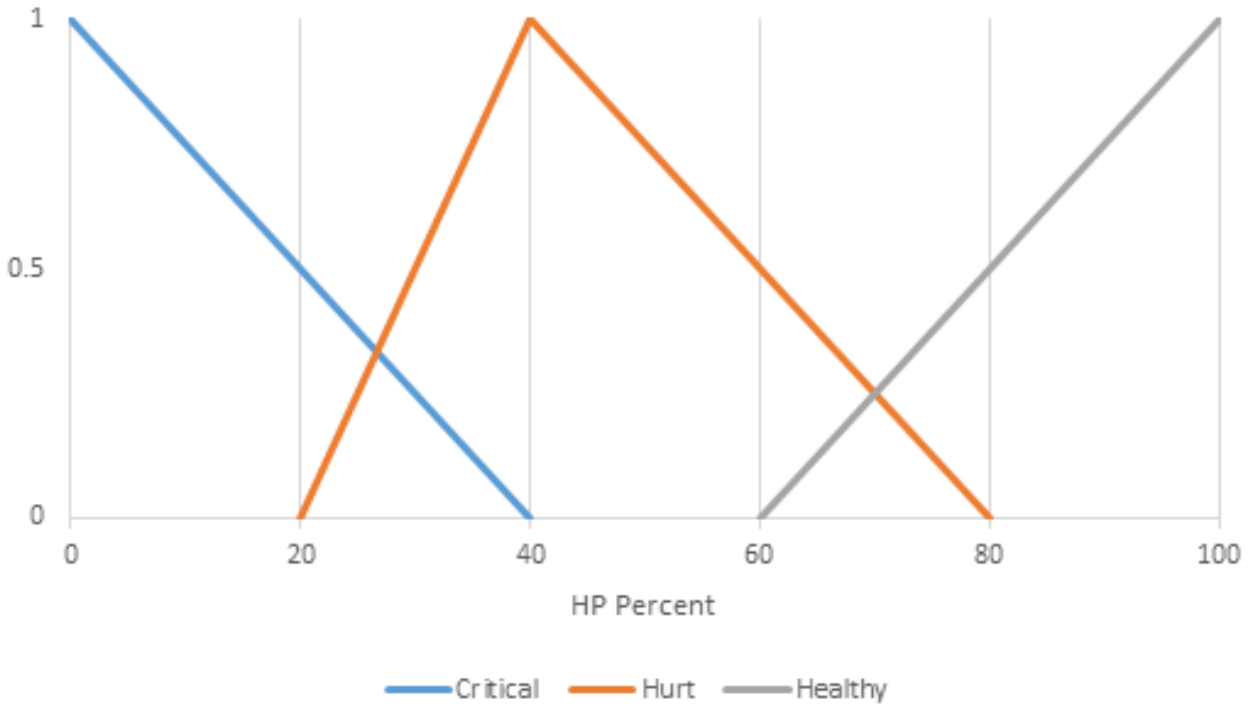
Dažniausiai žaidimuose suteikti dirbtiniam intelektui žmogišką intelektą nėra pagrindinis kurėjų tikslas. Dažnu atveju rašomas kodas būna skirtas kontroliuoti personažą, kuris nėra žmogus, kaip pvz. robotas, drakonas ar katė. Nors dažnu atveju, dirbtinis intelektas žaidimuose pasitelkiamas išspręsti įvairias sudėtingas problemas, kartais kuriami veikėjai, kurie nepasižymi labai protingomis tendencijomis taip pat yra tikslas, nes tai duoda žaidimui turiniui įvairovės. Taip pat dirbtinis intelektas pasitelkiamas kaip įrankis duoti ne žaidėjo valdomiems personažams skirtingas asmenybes, emocijas ar polinkius – pavaizduoti juos laimingus, išsigandusius, suirzusius ir t.t.

Dar viena iš labiausiai paplitusių dirbtinio intelekto metodikų žaidimuose yra sukčiavimas. Sukčiavimas yra labai įprastas būdas, padedantis kompiuterio valdomam žaidėjui suteikti pranašumą prieš protingus tikrus žaidėjus. Ši metodika pasireiškia įvairiausiais būdais, tačiau vienas iš pavyzdžių būtų kariniame strateginiame žaidime kompiuterio valdomam žaidėjui visos informacijos susijusios su tikru žaidėju suteikimas – tikro žaidėjo bazės vieta, valdomų personažų vietos ir kiekiai, jų tipai ir t.t. Tačiau kompiuterio valdomo veikėjo sukčiavimas gali būti ir blogas dalykas – jeigu tikram žaidėjui yra aišku, jog kompiuteris sukčiauja, tikram žaidėjui taps nuobodu ir jam atrodys, jog visos jo dedamos pastangos yra beprasmės. Taip pat nesubalansuotas kompiuterio sukčiavimas gali kompiuterio valdomam žaidėjui suteikti per daug galios, ko pasekoje žaidimas taps neįmanomas, todėl ir šiuo, ir praeitu atveju, kompiuterio valdomo žaidėjo sukčiavimas turi būti subalansuotas taip, kad tikram žaidėjui būtų sudaromas toks iššūkis, kad žaidimas atrodytų įdomus ir smagus [8].

## 2.4. Taikomi metodai, jų veikimo principai ir apribojimai

Šiuolaikiniuose žaidimuose naudojamas platus spektras dirbtinio intelekto metodų - paprastos baigtinių būsenų mašinos ir *fuzzy* logika, įvairūs kelio radimo algoritmai, ekspertų sistemos, sprendimų medžiai, gilūs neuroniniai tinklai, genetiniai algoritmai, paskatinimo principu veikiantys mokymosi algoritmai ir daug kitų. Šiame poskyryje apžvelgsime viena deterministinį metodą – fuzzy logika ir du nedeterministinius metodus – sprendimų medžius ir giliuosius neurononinius tinklus.

### 2.4.1. Fuzzy logika




**pav. 6 Personažo gyvybės atvaizduotos fuzzy grafu [10]**

*Fuzzy* logika yra įprastinės logikos plėtinys, kur objekto priklausomybės laipsnis aibėje nėra apibrėžtas sveikaisiais skaičiais 0 (objektas aibei nepriklauso) ir 1 (objektas aibei priklauso), bet gali būti bet kuris skaičius iš intervalo nuo 0 iki 1 [9]. *Fuzzy* logika dažniausiai yra naudojama *fuzzy* sistemos forma, o ją sudaro fuzzy kintamieji, *fuzzy* taisyklės ir *fuzzy* išvadų variklis. Naudojantis *fuzzy* logika, kintamieji yra paverčiami į *fuzzy* kintamuosius naudojantis narystės funkcijų rinkinį pagal galimą jų verčių diapazoną. Narystės funkcijos priskiria aiškią kintamojo reikšmę *fuzzy* etiketei su tiesos laipsnius, kuris dažniausiai yra tarp 0 ir 1. Pavyzdžiui, kaip matome 2 pav., personažo gyvybės žaidime gali būti padalintos į tris dalis – *Critical, Hurt* ir *Healthy*, kaip matoma paveikslėlyje aukščiau. Kintamieji gali būti traktuojami kaip įvestis, pavyzdžiui personažo parametrai, arba išvestis. *Fuzzy* taisyklės gali būti aprašomos naudojantis aprašytais *fuzzy* kintamaisiais ar jų aibe. Taisyklės dažniausiai yra jeigu – tai forma, o jų gramatika panaši į *boolean* logiką. Po to kai taisyklės yra aprašytos, *fuzzy* išvadų variklis naudojamas, kad gauti išvadas apie esamų kintamųjų reikšmes pagal sukurtas taisykles, ko pasekoje išvesties kintamiesiems suteikiamos *fuzzy* reikšmės. Galiausiai, *fuzzy* kintamieji gali būti atverčiami atgal į paprastus kintamuosius. Kaip ir dauguma kitų dirbtinio intelekto technikų, *fuzzy* logika buvo testuojama žaidimuose, norint sukurti paprastą dizainą, su

kombinuotą su inteligentiškais agentais. *Fuzzy* logika patenkina šiuos reikalavimus, kadangi ji yra pakankamai paprasta naudoti, primena tikrą kalbą ir turi patogią ir greitą dizaino metodologiją. *Fuzzy* logika gali būti naudojama žaidimo kompiuterio valdomų veikėjų sprendimų priėmimo procese, kaip pavyzdžiui daikto ar ginklo pasirinkimas arba kitų veikėjų valdymas, tai pat rizikų vertinimui. *Fuzzy* logikos naudojimas žaidimų kūrėjams atneša daug teigiamų dalykų, kadangi jos naudojimas reikalauji tik paprastos logikos supratimo, todėl tiek *fuzzy* taisyklės, tiek *fuzzy* kintamieji gali būti kuriami visų, dalyvaujančių žaidimo kūrimo procese. Tradicinis kompiuterio valdomų žaidėjų sprendimų priėmimas gali būti kiek per daug staigus naudojant tradicinius metodus, todėl *fuzzy* yra puikus kelias padaryti šiuos sprendimus labiau nuosekliais. Tačiau kaip ir visa kita, *fuzzy* logika nėra tobula – ji reikalauja visiškai teisingų įvesties ir išvesties kintamųjų ribų nustatymo, o sistemose su šimtais skirtingų žaidimo agentų veikiančių vienu metu, gali suprastinti žaidimo veikimą, kadangi šimtai taisyklių patikrinimų veiks kiekvieną sekundę [11].

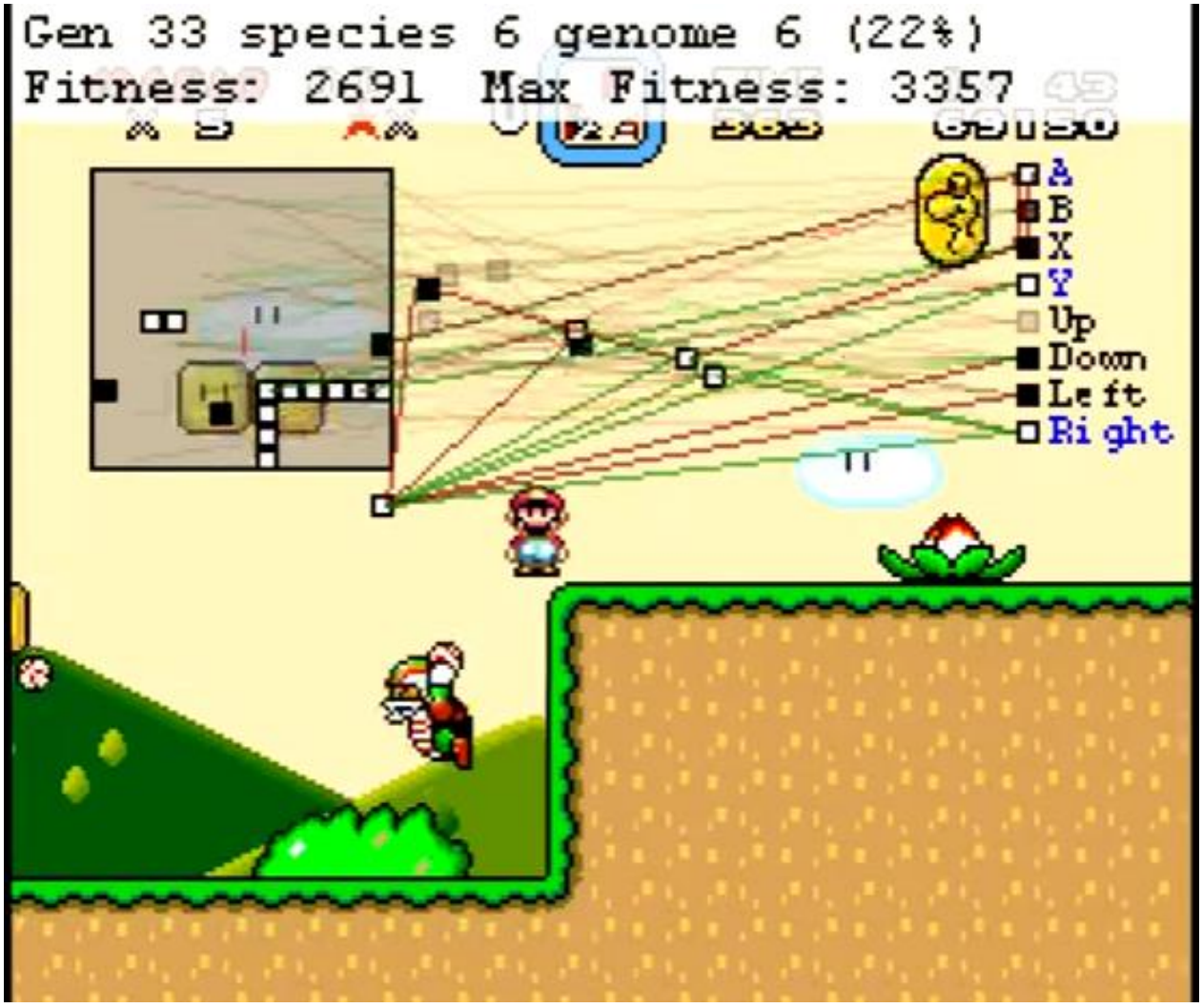
### 2.4.2. Sprendimų medžiai


**pav. 7 Sprendimų medžio pavyzdys [14]**

Sprendimų medžiai, tai prižiūrimo mašininio mokymosi metodai, kurie gali būti apmokomi atlikti klasifikaciją ir regresiją. Tai yra vieni iš paprasčiausių mašininio mokymosi algoritmų, naudojamų žaidimų kūrime. Jie padeda įvertinti kintamųjų reikšmių svarbą, nustatydami tam tikras pasirinkimų taisykles atsižvelgdami į duomenų charakteristikas. Sprendimų medžiai yra lengvai suprantami ir interpretuojami, o pasirinkimai neturėtų būti sunkiai įvertinami. Žaidimų dizaino stadijoje, naudojamos sprendimų lentelės [13]. Sprendimų medžiai yra sudaryti iš šakninio lapo, kuris dažniausiai yra situacija, kurioje žaidėjas/veikėjas yra iš pradžių, ir lapų, kurie reprezentuoja kiekvieną tolimesnį pasirinkimą. Kiekvienas lapas dažniausiai turi ribotą kiekį lapų, arba jų neturi išvis. Sprendimų medžiu yra laikoma, kai daugiau nei vienas pasirinkimas yra duodamas tikram žaidėjui ir kai žaidėjas kažkurį iš jų pasirenka, jam parodoma dar pasirinkimų, kurie susiję su praeitu jo pasirinkimu. Sprendimų medžiai leidžia daug skirtingų pasirinkimų ir jų kombinacijų, iš kurių kiekviena yra unikali ir atskirta viena nuo kitos. Sprendimų medžiai leidžia žaidėjui susikurti savo istoriją ar išeigas priklausomai nuo žaidėjo

padarytų pasirinkimų. Vienintelis skirtumas tarp sprendimų medžių ir visiškos laisvės žaidėjui yra tai, kad pasirinkimų rezultatai yra iš anksto žinomi. Modernūs kompiuteriniai žaidimai dažniausiai turi sprendimų medžius, apie kuriuos žaidėjas nieko nežino. Kompiuteriniai žaidimai dažniausiai turi tam tikras užuominas, kurios priveda prie tam tikro rezultato ir pasako žaidėjui koks tas rezultatas bus dar prieš padarant pasirinkimą. Sprendimų medžiai duoda žaidėjui įžvalgų, kas jos laukia žaidime, bei duoda užuominų kuris kelias yra teisingas. Pavyzdys – žaidime „*Star Wars Jedi: Fallen Order*“ žaidėjui žaidžiant žaidimą duodamos įžvalgos, kokia būtų ateitis, jeigu tam tikros aplinkybės susiklostytų. Taip pat, žaidime „*The Witcher 3: Wild Hunt*“, žaidimas turi skirtingas pabaigas priklausomai nuo žaidėjo padarytų veiksmų – įvykdytų misijų, neskaitant pagrindinės misijos, kiekio ir pasirinkimų, padarytų vykdant misijas [12]. 3 pav. matomas paslėptas sprendimų medis, kuriuo kompiuterio valdomas personažas gali priimti tam tikrus sprendimus, priklausomai nuo medžio šakos, kurioje jis šiuo metu yra, lygio.

### 2.4.3. Gilieji neuroniniai tinklai




**pav. 8 Konvoliucinis neuroninis tinklas mokosi žaisti žaidimą „*Mario*“ [16]**

Dirbtiniai neuroniniai tinklai yra algoritmai, kurių struktūra yra šiek tiek panaši į žmogaus smegenų struktūrą, ir kurie gali išmokti įvairias charakteristikas iš apmokymui skirtų duomenų

ir modeliuoti net labai sudėtingas tikro pasaulio ar kompiuterinio žaidimo situacijas. Palyginus su klasikiniais dirbtinio intelekto metodais, neuroniniai tinklai gali pereiti tam tikras spragas žaidimo dizaine. Neuroniniai tinklai yra prisitaikantys ir sugeba pasitaisyti priklausomai nuo besikeičiančių žaidimo sąlygų realiu laiku. Atitinkami mokymo metodai gali būti naudojami priklausomai nuo to kokį žaidimą ir kokį neuroninį tinklą žaidimo kūrėjas bando realizuoti. Dirbtinio intelekto agentai, naudojantys giliuosius neuroninius tinklus strateginiuose žaidimuose gali greitai pakeisti savo žaidimo strategijas, kad neatsiliktų nuo tikrų žaidėjų, kadangi jie turi savybę mokytis ir tobulėti. Gilieji neuroniniai tinklai yra puikus būdas užtikrinti, kad žaidimas išliks sunkus netgi po ilgo žaidimo laiko. Gilieji neuroniniai tinklai šiuo metu yra labai populiarūs ir paklausūs kaip žaidimo agentų dizaino įrankis. Gilieji neuroniniai tinklai naudoja kelis neuroninių tinklų sluoksnius, kurie naudojantis žaidimo suteikiamais įvesties duomenimis tobulėja ir mokosi žaisti. Kontroliuodami vieną ar kelis žaidimo agentus, gilieji neuroniniai tinklai pasiekia geresnius rezultatus kompiuterio kontroliuojamo žaidėjo rezultatų pasiekime, nei kiti dirbtinio intelekto metodai, kas leidžia jiems būti tiek kompiuterio kontroliuojamais žaidėjais, tiek žaidimo aplinkos valdytojais [13]. Gilieji neuroniniai tinklai mokantis gali nekreipti dėmesio į nereikšmingus ir tik žmogui skirtus veiksnius – oro sąlygas žaidime, besikeičiančias spalvas grafinėje sąsajoje ar aplinkoje, todėl gali greitai pasiekti įspūdingus rezultatus [15]. 4 pav. matomas konvoliucinis neuroninis tinklas, kuris mokosi žaisti žaidimą „*Mario*“, naudodamasis valdymo komandomis kaip įvesties duomenimis, ir artėjimu prie tikslo kaip paskatinimu.

## 2.5. Egzistuojančių rinkoje strateginiu žaidimų dirbtinio intelekto naudojimo analizė

Šiuo metu rinkoje egzistuoja labai didelis kiekis įvairiausio tipo strateginių žaidimų – nuo paprasčiausių žaidimų, tokių kaip šaškės, iki kompleksiškų, iš ne vienos dirbtinio intelekto sistemos sudarytų grandiozinės strategijos žaidimų, tokių kaip „*Sid Meier's Civilization V*“ ar „*Crusader Kings II*“. Žaidimai naudoja dirbtinio intelekto metodus, kad padidintų žaidimo patrauklumą potencialiems žaidėjams, kad sukeltų tikro, „gyvo“ ir protingo priešininko jausmą ir kad užtikrintų, jog žaidimą galima būtų pereiti ne vieną kartą. Šiam tikslui pasiekti, žaidimų kūrėjai ieško vis naujesnių ir labiau priimtinų dirbtinio intelekto metodų, tačiau ne visada renkasi patį naujausią, geresnį, tačiau mažiau ištirtą ir ištestuotą metodą, kadangi tai padidina kuriamo žaidimo tiek piniginius tiek laiko kaštus. Dažnu atveju liekama prie galbūt senesnių, tačiau labiau ištirtų ir laiko patikrintų metodų, kadangi jie siūlo labiau nuspėjamą kompiuterinių žaidimų kūrimo procesą ir lengviau nuspėjamus tiek piniginius tiek laiko kaštus. Šiame skyriuje atlikta dviejų pasirinktų, vis dar aktualių ir populiarių strateginių žaidimų ir juose naudojamų dirbtinio intelekto metodų analizė – „*Sid Meier's Civilization V*“ ir „*Age of Empires III*“.

### 2.5.1. Sid Meier‘s Civilization V


**pav. 9 Strateginio žaidimo „Sid Meier‘s Civilization V“ pagrindinio žaidimo lango vaizdas [17]**

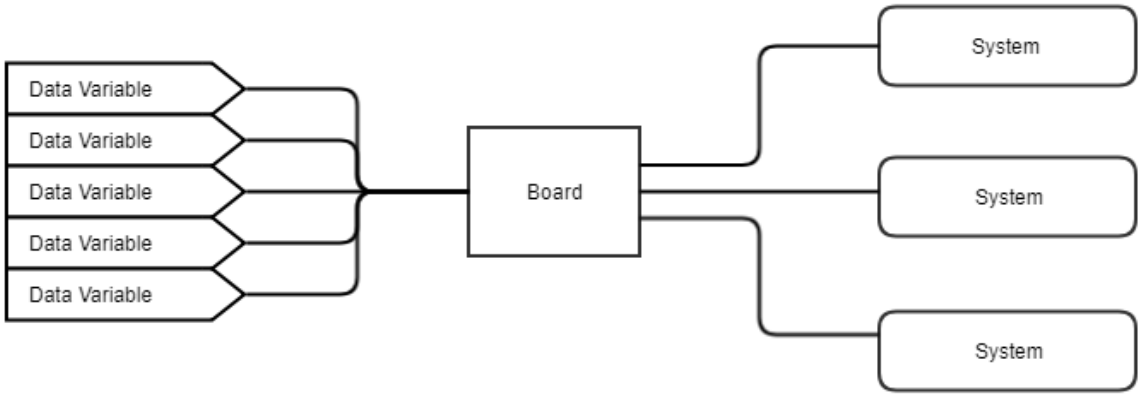
„*Sid Meier‘s Civilization V*“ yra strateginis kompiuterinis žaidimas, sukurtas *Firaxis Games*, kombinuojantis tiek ėjimais grįstą žaidimo principą, tiek realaus laiko strategiją. Žaidimo tikslas yra nuo senų iki naujų laikų pastatyti savo civilizaciją ir pasiekti vieną iš galimų pergalių tipų naudojantis mokslu, plėtimusi, diplomatija, tyrinėjimu, ekonominiu vystymusi arba kariuomene. Žaidėjas valdo vieną iš 43 galimų visų laikų civilizacijų, ir rungtyniauja su kompiuterio valdomomis civilizacijomis arba kitais tikrais žaidėjais tinklo rėžimu. Žaidime figūruoja skirtingos eros, kurios eina žaidėjui tobulinant savo civilizaciją. Žaidimas pastatytas naudojantis nauju žaidimų varikliu, ir naudoja šešiakampius laukelius žaidimo žemėlapio objektams [18]. Žaidimo naudojamas dirbtinis intelektas susideda iš kelių skirtingų dalių, kurių visuma vadinasi *Blackboard* sistema [19]. Kartu su *Blackboard* sistema, žaidimas naudoja sukčiavimą (2.3 poskyris šiame dokumente). Šios sistemos dalys sudaro *Blackboard* visumą:

1. Karo dirbtinis intelektas – atsakingas už kareivių valdymą, naujų kareivių paruošimą, kareivių prioriteto nustatymą, skirtingų kareivių tipų parinkimą atitinkamiems veiksmams.
2. Prekiavimo dirbtinis intelektas – atsakingas už prekiavimą su kitais kompiuterio ir tikrais žaidėjais, resursų vystymą, laukelių gerinimą.
3. Mokslo dirbtinis intelektas – atsakingas už mokslo civilizacijoje plėtojimą, mokslinės šakos pasirinkimus.
4. Tyrinėjimo dirbtinis intelektas – atsakingas už žemėlapio naršymą, žvalgų siuntimą.

### 2.5.2. Blackboard sistemos žaidimuose

*Blackboard* sistema yra vienas iš dirbtinio intelekto realizavimo žaidimuose būdų. Tai metodas, kuris susideda iš žinomų faktų bazės, dar kitaip vadinamos lenta, kuri yra iteratyviai atnaujinama grupės žinių šaltinių, pradedant nuo problemos specifikacijos ir baigiant problemos sprendimu. Kiekvienas iš šaltinių atnaujina lentoje esamą informaciją galimais ne

pilnais sprendimais, kai žinių šaltinio vidiniai apribojimai pasiekia lentos būseną [20]. *Blackboard* sistema padeda atsieti skirtingas funkcijas, taip sukuriant solidžią architektūrą su lengva implementacija ir palaikymu.




**pav. 10 Elementari Blackboard sistemos schema [21]**

*Blackboard* sistema yra žinučių siuntimo ir duomenų sistema. Žaidimo metu, užkraunamos lentos, kurios turi sąrašą duomenų, nuo kurių priklausomai iškviečiami įvykiai, kuriuos lentą stebinčios sistemos gali panaudoti keičiant duomenis. Lentoje kiekviena eilutė turi unikalų Id, kuris naudojamas prieiti prie atitinkamų duomenų arba iškviesti atitinkamą įvykį. Lenta laiko duomenis atskirais kintamaisiais, o jie patys yra indeksuojami, kartu su jų raktais, todėl lenta veikia kaip kontroleris. Nors *Blackboard* sistema yra puikus būdas atsieti funkcijas ir pagreitinti skirtingų dirbtinio intelekto sistemų valdymą, tai nėra magiškas atsakymas į visus žaidimo reikalavimus ar problemas [21].

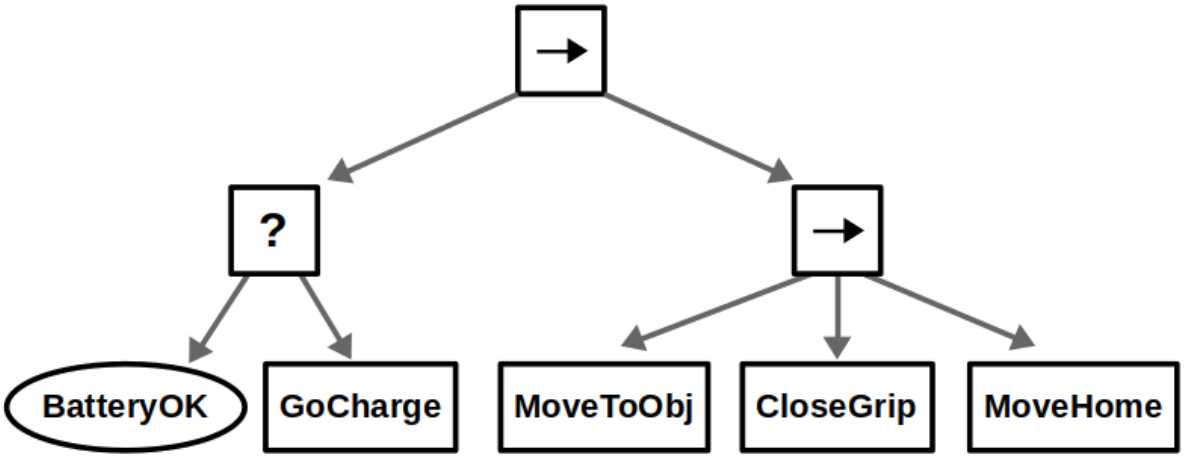
### 2.5.3. Age of Empires III


**pav. 11 Žaidimo „*Age of Empires III*“ pagrindinio žaidimo lango vaizdas, kuriame matomi skirtingi pastatai bei padaliniai [22].**

„*Age of Empires III*“ yra realaus laiko strateginis kompiuterinis žaidimas sukurtas *Microsoft*. Žaidime vaizduojama Amerikų kolonizacija, o žaidėjas gali pasirinkti vieną iš 14 civilizacijų už kurias jis gali žaisti. Žaidimas atvedė į strateginių žaidimų žanrą, tokių kaip pagrindinis miestas ir puikiai kombinuoja realaus laiko strategiją su rolių žaidimo atributais. Žaidėjo tikslas pasiekti pergalę prieš kompiuterio valdomus arba realius žaidėjus, statant savo miestą, išgaunant resursus, apmokant kareivius ir statant pastatus, įtvirtinimus ir vystant ekonomiką. Žaidimas suskirstytas į įvairius amžius, kurie eina žaidėjui tobulėjant ir tobulinant savo miestą [23]. Dirbtinis intelektas žaidime valdo kompiuterio valdomus žaidėjus, jų įsakymus ir priešų ir draugų padalinių sąveiką. Dirbtinis intelektas leidžia kompiuterio žaidėjams turėti skirtingus prioritetus, statyti įmantrias gynybines bazes ir jas pastoviai gerinti. Dirbtinis intelektas kompiuterio žaidėjams taip pat suteikia skirtingas taktikas ir požiūrius, priklausomai nuo to, už kurią civilizaciją jis žaidžia [24]. Žaidimo dirbtinis intelektas pagrinde naudoja elgsenų medžius įkomponuotus kartu su sprendimų medžiais (2.4.2 poskyris šiame dokumente) bei sukčiavimą (2.3 poskyris šiame dokumente).

### 2.5.4. Elgsenų medžiai

Elgsenų medžiai šiuo metu yra dominuojanti technika, naudojama žaidimų kūrėjų kaip dirbtinio intelekto metodas jų kuriamuose žaidimuose. Ši technika pagrinde buvo sukurta kaip patobulinimas baigtinių būsenų mašinoms, kadangi šių pagrindinis trūkumas yra, jog esant šimtams būsenų, baigtinių būsenų mašinos tampa vis sunkiau ir sunkiau testuojamos. Pagrindinė idėja – elgsenos yra vykdomos arba po vieną, arba kelios paraleliai, o sprendimas, kurią elgseną vykdyti, atliekamas kas tam tikrą laiko tarpą, pavyzdžiui kas sekundę. Elgsenų medžiai taip pat leidžiam pakeisti elgsenų vykdymo seką, jeigu paskutinę sekundę buvo priimtas sprendimas vykdyti elgseną, kurios nėra sekoje. Elgsenų medžiai suteikia karkasą lengviau suprantamoms ir lengviau skaitomoms dirbtinio intelekto sistemoms, o vizualiai reprezentuoti medžiai leidžia lengvai juos testuoti [25]. Tačiau kaip ir visa kita, elgsenų medžiai turi savo trūkumų – labai dideli elgsenų medžiai turi labai didelius vertinimų kaštus žaidimo veikimo greičiui, ir nors elgsenų medžiai yra geras metodas elgsenų organizavimui, jie nesuteikia dirbtiniam intelektui labiau optimizuoto sprendimų atlikimo metodo.




**pav. 12 Primityvus valdymo elgsenų medis su dviem šakomis [25]**

## 2.6. Išvados

Buvo atlikta įvairių literatūros šaltinių, susijusių su magistrinio darbo tema, analizė.

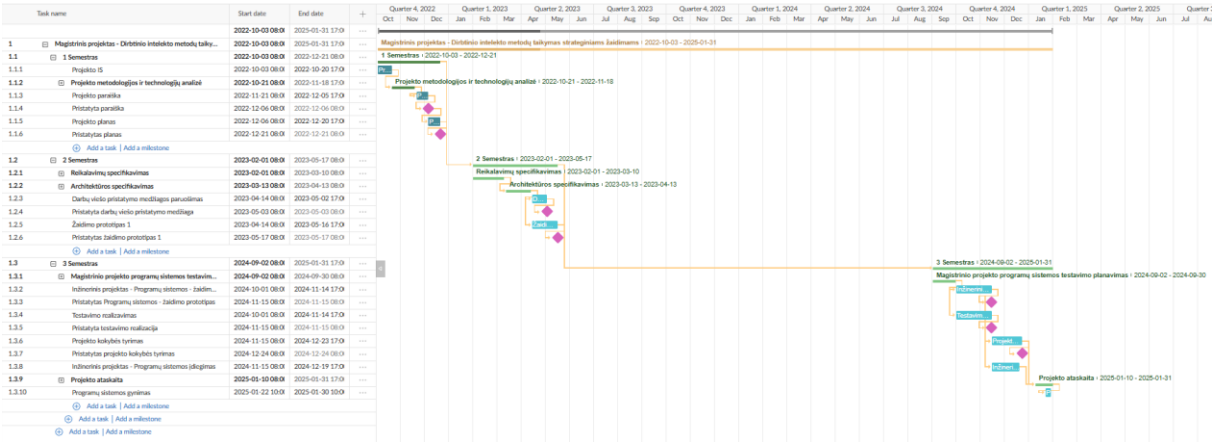
Išanalizavus dirbtinio intelekto naudojimą kompiuteriniuose žaidimuose pastebėtas ryški pažanga šioje srityje – per paskutinius 70 metų dirbtinis intelektas kompiuteriniuose žaidimuose patobulėjo nuo paprastų statinių metodų iki sudėtingų neuroninių tinklų ir tapo neatskiriama kompiuterinių žaidimų dalimi.

Pastebėta, kad dirbtinis intelektas kompiuteriniuose žaidimuose išskiriamas į dvi rūšis, o šios rūšys turi daug skirtingų porūšių, kurie tiek pavieniai, tiek kartu su kitais naudojami kompiuteriniuose žaidimuose.

Atliekant esamų sprendimų analizę, išsiaiškinta, jog tiek populiarūs, tiek nelabai populiarūs žaidimai naudoja dirbtinį intelektą vienokia ar kitokia forma ir padeda kompiuterio valdomiems žaidėjams imituoti tikrus žaidėjus ir tikriems žaidėjams užtikrinti įdomiai praleistą laiką.

Pastebėta tendencija, kad didžioji dauguma net ir nesenai išleistų kompiuterinių žaidimų vis dar naudoja kiek pasenusius, tačiau per laiką išdirbtus dirbtinio intelekto metodus, o tik maža dalis naujų kompiuterinių žaidimų bando įtraukti sudėtingesnius ir ne tiek išdirbtus dirbtinio intelekto metodus, dėl augančios žaidimo kainos ir testavimo laiko.

# 3. PROJEKTO PLANAS




**pav. 9 Projekto planas**

# 4. REIKALAVIMŲ SPECIFIKACIJA

## 4.1. Sistemos paskirtis

### 4.1.1. Projekto kūrimo pagrindas

Strateginiai kompiuteriniai žaidimai yra neatsiejama didelės grupės žmonių laisvalaikio dalis. Dirbtinis intelektas strateginiuose žaidimuose padeda sukurti gyviau reaguojančias aplinkas, prie žaidėjo prisitaikančias sistemas bei sudėtingesnius, labiau į tikrą gyvą žmogų panašius kompiuterio valdomus žaidėjus. Dirbtinio intelekto dėka, strateginiai žaidimai sukuria virtualius pasaulius, kurie duoda geresnę patirtį savo žaidėjams ir padeda labiau įsijausti į žaidimo pasaulį. Tinkamų dirbtinio intelekto metodų naudojimas kuriant strateginius žaidimus užtikrina, kad žaidimas bus tokio sunkumo, kad žaidėjui jis greitai neatsibostų ir pastoviai suteiktų įveikiamų iššūkių. Šiuo metu beveik kiekvienas strateginis žaidimas naudoja tam tikrus dirbtinio intelekto metodus, kaip pavyzdžiui algoritmus priešininkų ar sąjungininkų elgsenos modeliavimui, kelio radimui, duomenų gavybai, procedūriniam turinio generavimui ar žaidėjo patirties modeliavimui.

Problema - daugelis sukurtų algoritmų ir metodikų yra apsaugoti juos kuriančių kompanijų ar korporacijų ir nėra viešai prieinami.

Dėl šių priežasčių yra vykdomas tyrimas, kurio tikslas - ištirti esamus dirbtinio intelekto algoritmus ir metodikas naudojamas kompiuterinių žaidimų strategijai realizuoti, pasiūlyti patobulinimus šiems algoritmams ir metodikoms, ir įvertinti jų efektyvumą sukuriant algoritmus ir metodikas realizuojantį žaidimą. Projekto metu bus kuriamas strateginis žaidimas, kuris realizuos kelis vėliau pasirinktus dirbtinio intelekto metodus. Kuriant šį žaidimą ir vykdant tyrimą, naudojami dirbtinio intelekto metodai bus patobulinami ir pasiūlomos alternatyvos jeigu šių dirbtinio intelekto metodų naudojimas nebus optimalus.

### 4.1.2. Sistemos tikslai

- Atrasti optimaliausius dirbtinio intelekto taikymo metodus įvairiems scenarijams, su kuriais susiduriama strateginiuose žaidimuose
- Pasiūlyti alternatyvas mažiau optimaliems dirbtinio intelekto metodams arba patobulinti esamus
- Patobulinti esamus strateginių žaidimų dirbtinio intelekto panaudojimo būdus

## 4.2. Užsakovai, pirkėjai ir kiti sistema suinteresuoti asmenys

### 4.2.1. Užsakovas

Šis projektas tiesioginio užsakovo neturi, tačiau kaip užsakovą įvardiname šio projekto vadovą.

### 4.2.2. Pirkėjas

Žaidimas bus nemokamas, todėl pirkėjo nėra.

### 4.2.3. Kiti suinteresuoti asmenys

Sistemos programuotojas yra atsaking už visus sprendimus, priimamus projekto kūrimo metu – projekto planavimas, architektūra ir implementacija.

## 4.3. Vartotojai

Sistemos vartotojai skiriami į dvi skirtingas grupes: žmones mėgstančius žaisti strateginius žaidimus ir strateginių žaidimų kūrėjus.

Informacija apie žmonių mėgstančių žaisti strateginius žaidimus naudotojų grupę

**Priklauso:** kompiuterinių žaidimų žaidėjai.

**Tikslai:** gerai praleisti laiką.

**Bendrosios charakteristikos:** neapibrėžtos, kadangi galimas platus individų spektras.

**Patirtis dalykinėje srityje:** Žemas – aukštas.

**Patirtis technologinėje srityje:** Žemas – aukštas.

**Prioritetas:** Žemas.

**Apmokymas:** Nereikalingas.

Informacija apie strateginių žaidimų kūrėjų naudotojų grupę

**Priklauso:** kompiuterinių žaidimų kūrėjai.

**Tikslai:** gauti žinių apie dirbtinio intelekto metodų panaudojimą kuriamuose žaidimuose.

**Bendrosios charakteristikos:** neapibrėžtos, kadangi galimas platus individų spektras.

**Patirtis dalykinėje srityje:** Aukštas.

**Patirtis technologinėje srityje:** Aukštas.

**Prioritetas:** Aukščiausias.

**Apmokymas:** Žinių apie dirbtinio intelekto modelių veikimo principus įgavimas.

## 4.4. Įpareigojantys apribojimai

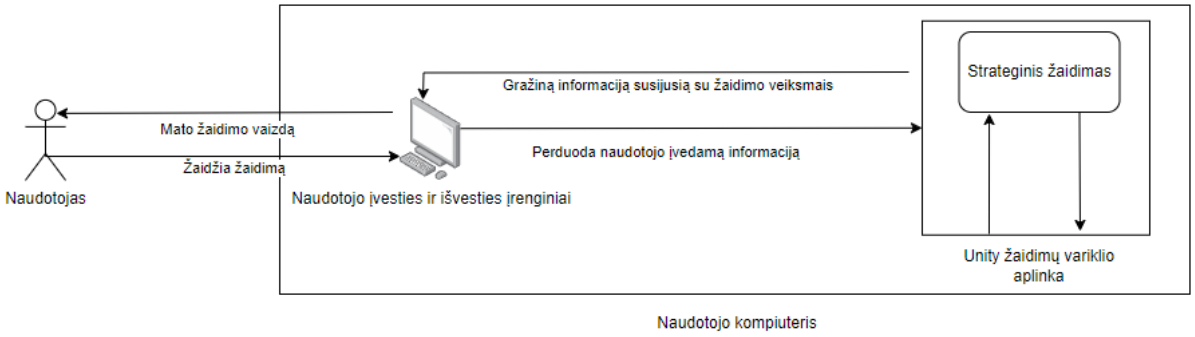
### 4.4.1. Apribojimai sprendimui

Apribojimas: sistema turi būti pritaikyta įrenginiams naudojantiems Windows 10 arba naujesnę Windows operacinę sistemą.

Pagrindimas: Didžioji dauguma naudotojų naudoja stacionarius kompiuterius, turinčius aukščiau minėtą operacinę sistemą, todėl sistema turi tinkamai veikti juose.

Atitikimo kriterijai: Sistema tinkamai veikia stacionariuose kompiuteriuose su Windows 10 ar naujesnėmis Windows operacinėmis sistemomis.

### 4.4.2. Diegimo aplinka




**1 pav. Sistemos diegimo aplinka.**

### 4.4.3. Bendradarbiaujančios sistemos

Sistema neturi jokų bendradarbiaujančių sistemų.

### 4.4.4. Komerciniai specializuoti programų paketai

Žaidimo veikimas priklauso nuo Unity žaidimų variklio aplinkos, kuri naudotojams bus įdiegta žaidimo įrašymo metu. Ši sistema reikalinga, kadangi žaidimas kuriamas naudojantis Unity žaidimų varikliu.

### 4.4.5. Numatoma darbo vietos aplinka

Darbuotojai dirba nuotoliniu būdų. Darbuotojų kompiuteriai turi turėti šias arba aukštesnes specifikacijas: 16GB RAM, 50GB laisvos atminties diske, 8 branduolių procesorių ir vaizdo plokštę geresnę nei Nvidia RTX 2060 arba AMD RX 5700.

### 4.4.6. Sistemos kūrimo terminai

Sistemos veikiantis prototipas turi būti įgyvendintas iki 2024 pavasario pabaigos. Pilnai veikianti sistema turi būti įgyvendinta iki 2024 žiemos vidurio.

### 4.4.7. Sistemos kūrimo biudžetas

Galutinė sistemos kaina neturi viršyti 35,000€.

## 4.5. Svarbūs faktai ir prielaidos

Sistema ir tyrimas bus orientuoti į pasaulinę rinką.

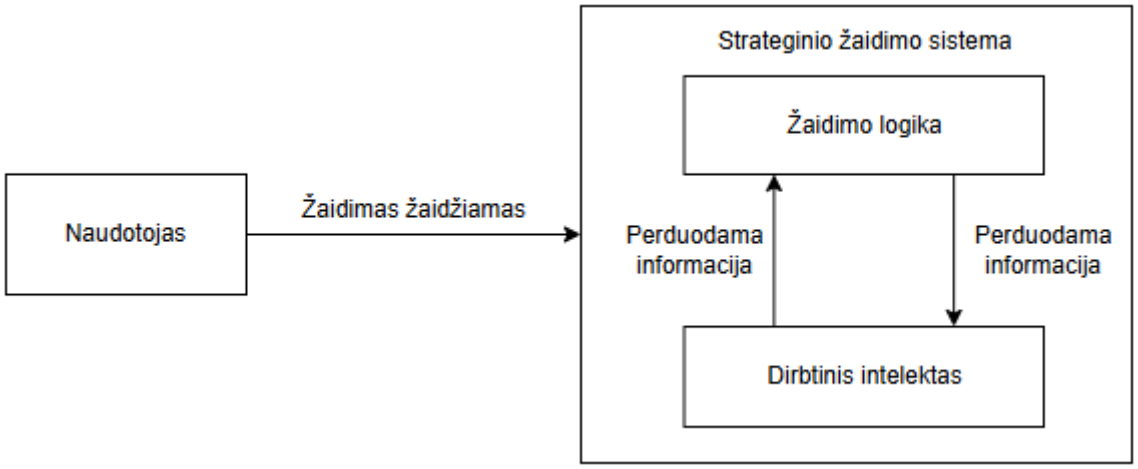
Žaidimas bus įgyvendintas naudojantis C# programavimo kalba ir Unity žaidimų varikliu.

Dirbtinio intelekto įgyvendinimui gali būti pasitelkti algoritmai ir metodikos, įgyvendinamos C# programavimo kalba ir įvariomis šios kalbos bibliotekomis ir karkasais.

Projektas turi būti įgyvendintas iki 2025 metų pavasario galo.

## 4.6. Funkciniai reikalavimai

### 4.6.1. Veiklos kontekstas




**2 pav. Veiklos konteksto diagrama**

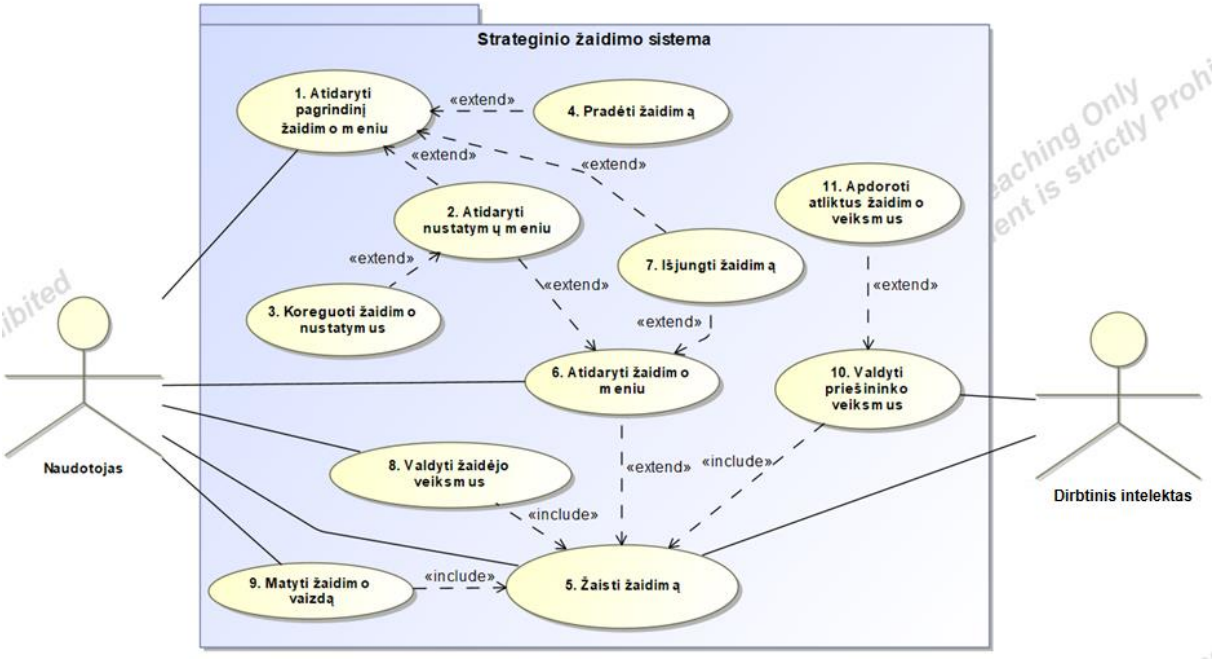
### 4.6.2. Veiklos padalinimas

**1 Lentelė.** Veiklos įvykių sąrašas

| Eil. Nr. | Įvykio pavadinimas | Įeinantys/Išeinantys informacijos srautai |
|---|---|---|
| 1 | Žaidžiamas žaidimas | Žaidėjo įvestis naudojantis įvesties įrenginiais (pelė ir klaviatūra) (įeinantis) |
| 2 | Pateikiama vaizdinė žaidimo informacija | Žaidėjui matomas žaidimo langas, rodos išvesties įrenginyje (monitoriuje) (išeinantis) |
| 3 | Apdorojami žaidėjo padaryti veiksmai | Žaidimo logikos ir panaudotų dirbtinio intelekto algoritmų apdorojami informacijos srautai, gauti iš žaidėjo padarytų veiksmų (įeinantis) |
| 4 | Priimami sprendimai | Žaidimo logikos ir dirbtinio intelekto įvestis į žaidimo sistemą, pagrįsta praietais veiksmais (išeinantis) |

## 4.7. Sistemos sudėtis

### 4.7.1. Sistemos ribos




**3 pav. Strateginio žaidimo panaudojimo atvejų diagrama**

### 4.7.2. Panaudojimo atvejų sąrašas

**2 Lentelė.** Panaudojimo atvejis nr. 1

| |
|---|
| 1. PANAUDOJIMO ATVEJIS: Atidaryti pagrindinį žaidimo meniu<br>**Vartotojas/Aktorius:** Naudotojas<br>**Aprašas:** Atidaromas pagrindinis žaidimo meniu.<br>**Prieš sąlyga:** Naudotojas įjungė žaidimą.<br>**Sužadinimo sąlyga:** Naudotojas paspaudė pagrindinio žaidimo meniu įjungimo mygtuką.<br>**Po-sąlyga:** Atidaromas pagrindinis žaidimo meniu. |

**3 Lentelė.** Panaudojimo atvejis nr. 2

| |
|---|
| 2. PANAUDOJIMO ATVEJIS: Atidaryti nustatymų meniu<br>**Vartotojas/Aktorius:** Naudotojas<br>**Aprašas:** Atidaromas nustatymų meniu langas.<br>**Prieš sąlyga:** Naudotojas atidarė pagrindinį žaidimo meniu.<br>**Sužadinimo sąlyga:** Naudotojas paspaudė nustatymų meniu įjungimo mygtuką.<br>**Po-sąlyga:** Atidaromas nustatymų meniu langas. |

**4 Lentelė.** Panaudojimo atvejis nr. 3

| |
|---|
| 3. PANAUDOJIMO ATVEJIS: Koreguoti žaidimo nustatymus<br>**Vartotojas/Aktorius:** Naudotojas<br>**Aprašas:** Koreguojami žaidimo nustatymai – garsumas, grafikos lygis ir rezoliucija.<br>**Prieš sąlyga:** Naudotojas atidarė nustatymų meniu langą. |

| | |
|---|---|
| **Sužadinimo sąlyga:** | Naudotojas pakeitė garso, grafikos arba rezoliucijos lygį naudodamasis grafine sąsaja. |
| **Po-sąlyga:** | Žaidimo nustatymai pakeičiami į naujai pasirinktus. |

**5 Lentelė.** Panaudojimo atvejis nr. 4

| | |
|---|---|
| 4. PANAUDOJIMO ATVEJIS: Pradėti žaidimą | |
| **Vartotojas/Aktorius:** | Naudotojas |
| **Aprašas:** | Pradedamas žaidimas. |
| **Prieš sąlyga:** | Naudotojas atidarė pagrindinį žaidimo meniu. |
| **Sužadinimo sąlyga:** | Naudotojas paspaudė žaidimo pradėjimo mygtuką. |
| **Po-sąlyga:** | Įjungiamas pagrindinis žaidimo langas ir pradedamas žaidimas. |

**6 Lentelė.** Panaudojimo atvejis nr. 5

| | |
|---|---|
| 5. PANAUDOJIMO ATVEJIS: Žaisti žaidimą | |
| **Vartotojas/Aktorius:** | Naudotojas, Dirbtinis intelektas |
| **Aprašas:** | Žaidžiamas žaidimas tarp žaidėjo ir dirbtinio intelekto. |
| **Prieš sąlyga:** | Naudotojas atidarė pagrindinį žaidimo meniu. |
| **Sužadinimo sąlyga:** | Naudotojas paspaudė žaidimo pradėjimo mygtuką. |
| **Po-sąlyga:** | Žaidžiamas žaidimas tarp žaidėjo ir dirbtinio intelekto. |

**7 Lentelė.** Panaudojimo atvejis nr. 6

| | |
|---|---|
| 6. PANAUDOJIMO ATVEJIS: Atidaryt žaidimo meniu | |
| **Vartotojas/Aktorius:** | Naudotojas |
| **Aprašas:** | Atidaromas meniu žaidimo metu, leidžiantis keisti žaidimo nustatytumus ir išjungti žaidimą. |
| **Prieš sąlyga:** | Naudotojas atidarė pagrindinį žaidimo meniu. |
| **Sužadinimo sąlyga:** | Naudotojas paspaudė žaidimo pradėjimo mygtuką. |
| **Po-sąlyga:** | Įjungiamas žaidimo meniu žaidimo metu. |

**8 Lentelė.** Panaudojimo atvejis nr. 7

| | |
|---|---|
| 7. PANAUDOJIMO ATVEJIS: Išjungti žaidimą | |
| **Vartotojas/Aktorius:** | Naudotojas |
| **Aprašas:** | Išjungiamas žaidimas. |
| **Prieš sąlyga:** | Naudotojas atidarė žaidimo meniu. |
| **Sužadinimo sąlyga:** | Naudotojas paspaudė žaidimo išjungimo mygtuką. |
| **Po-sąlyga:** | Žaidimas išjungiamas. |

**9 Lentelė.** Panaudojimo atvejis nr. 8

| | |
|---|---|
| 8. PANAUDOJIMO ATVEJIS: Valdyti žaidėjo veiksmus | |
| **Vartotojas/Aktorius:** | Naudotojas |
| **Aprašas:** | Valdomi žaidėjo veiksmai, naudojantis įvesties įrenginiais ir atsižvelgiant į matomą žaidimo vaizdą. |
| **Prieš sąlyga:** | Žaidimas yra pradėtas. |
| **Sužadinimo sąlyga:** | Naudotojas paspaudė bent vieną iš registruotų žaidėjo valdymo klavišų klaviatūroje arba pelėje. |
| **Po-sąlyga:** | Valdomi žaidėjo veiksmai priklausomai nuo paspaustų įvesties įrenginių klavišų. |

**10 Lentelė.** Panaudojimo atvejis nr. 9

| | |
|---|---|
| 9. PANAUDOJIMO ATVEJIS: Matyti žaidimo vaizdą | |
| **Vartotojas/Aktorius:** Naudotojas | |
| **Aprašas:** | Matomas žaidimo vaizdas išvesties įrenginyje (monitoriuje). |
| **Prieš sąlyga:** | Naudotojas įjungė žaidimo aplikaciją. |
| **Sužadinimo sąlyga:** | - |
| **Po-sąlyga:** | Matomas visas žaidimo vaizdas bei pasinkti meniu. |

**11 Lentelė.** Panaudojimo atvejis nr. 10

| | |
|---|---|
| 10. PANAUDOJIMO ATVEJIS: Valdyti priešininko veiksmus | |
| **Vartotojas/Aktorius:** Dirbtinis intelektas | |
| **Aprašas:** | Valdomi priešininko veiksmai atsižvelgiant į apdorotus jau atliktus žaidimo veiksmus. |
| **Prieš sąlyga:** | Žaidimas yra pradėtas. |
| **Sužadinimo sąlyga:** | Naudotojas paspaudė žaidimo pradėjimo mygtuką. |
| **Po-sąlyga:** | Priešininkas yra valdomas atsižvelgiant į veiksmus, kuriuos reikia atlikti. |

**12 Lentelė.** Panaudojimo atvejis nr. 11

| | |
|---|---|
| 11. PANAUDOJIMO ATVEJIS: Apdoroti atliktus žaidimo veiksmus | |
| **Vartotojas/Aktorius:** Dirbtinis intelektas | |
| **Aprašas:** | Apdorojami žaidėjo ir dirbtinio intelekto modelio atlikti veiksmai bei apskaičiuojami geriausi ir naudingiausi sekantys veiksmai. |
| **Prieš sąlyga:** | Žaidimas yra pradėtas. |
| **Sužadinimo sąlyga:** | Naudotojas paspaudė žaidimo pradėjimo mygtuką. |
| **Po-sąlyga:** | Pastoviai apdorojama žaidėjo ir dirbtinio intelekto modelio veiksmų informacija ir parenkami geriausi ir naudingiausi sekantys veiksmai. |

## 4.8. Funkciniai reikalavimai ir reikalavimai duomenims

### 4.8.1. Funkciniai reikalavimai

**13 Lentelė.** Funkcinis reikalavimas nr. 1

| | | | | | |
|---|---|---|---|---|---|
| Reikalavimas: | F-1 | Reikalavimo tipas: | 9 | PA: | 5, 10, 11 |
| Aprašymas | Dirbtinis intelektas sugeba priimti tinkamus sprendimus žaidimo metu ir prisideda prie žaidėjo įtraukimo į žaidimą | | | | |
| Pagrindimas | Dirbitinis intelektas pagyvina žaidimą žaidėjui, kad šiam kuo ilgiau būtų įdomu žaisti. | | | | |
| Šaltinis | Sistemos užsakovas | | | | |
| Tinkamumo kriterijai | Dirbtinis intelektas padaro, kad žaidėjui būtų įdomu žaisti žaidimą. | | | | |
| Užsakovo patenkinimas | 5 | | Užsakovo nepatenkinimas | 5 | |
| Priklausomybės | Nėra | | Konfliktai | Nėra | |

## 4.9. Nefunkciniai reikalavimai

### 4.9.1. Reikalavimai sistemos išvaizdai

**12 Lentelė.** Nefunkcinis reikalavimas nr. 1

| Reikalavimas: | NF-1 | Reikalavimo tipas: | 9 | PA: | Visi |
|---|---|---|---|---|---|
| Aprašymas | Žaidimo grafinėje sąsajoje turi dominuoti pastelinės spalvos su viduramžių tematika | | | | |
| Pagrindimas | Švarus ir žaidėjui patrauklus žaidimo grafinės sąsajos stilius. | | | | |
| Šaltinis | Sistemos užsakovas | | | | |
| Tinkamumo kriterijai | - | | | | |
| Užsakovo patenkinimas | 4 | | Užsakovo nepatenkinimas | 3 | |
| Priklausomybės | Nėra | | Konfliktai | Nėra | |

### 4.9.2. Reikalavimai panaudojamumui

**13 Lentelė.** Nefunkcinis reikalavimas nr. 2

| Reikalavimas: | NF-2 | Reikalavimo tipas: | 9 | PA: | 1, 2, 3, 6 |
|---|---|---|---|---|---|
| Aprašymas | Žaidimo grafinė sąsaja lengvai suprantama be jokio išankstinio apmokymo | | | | |
| Pagrindimas | Lengvai suprantama grafinė sąsaja palengvina naujam žaidėjui greitai suprasti žaidimo veikimo principą. | | | | |
| Šaltinis | Sistemos užsakovas | | | | |
| Tinkamumo kriterijai | Žaidimo grafinė sąsaja yra greitai perprantama naujam žaidėjui. | | | | |
| Užsakovo patenkinimas | 4 | | Užsakovo nepatenkinimas | 3 | |
| Priklausomybės | Nėra | | Konfliktai | Nėra | |

### 4.9.3. Reikalavimai vykdymo charakteristikoms

**14 Lentelė.** Nefunkcinis reikalavimas nr. 3

| Reikalavimas: | NF-3 | Reikalavimo tipas: | 9 | PA: | 1, 2, 3, 6 |
|---|---|---|---|---|---|
| Aprašymas | Žaidimo grafinėje sąsajos elementai turi įsijungti per mažiau nei 1s nuo paspaudimo. | | | | |
| Pagrindimas | Žaidimo grafinė sąsaja turi veikti greitai, kad naudotojas nepatirtų diskomforto dėl trikdžių. | | | | |
| Šaltinis | Sistemos užsakovas | | | | |
| Tinkamumo kriterijai | - | | | | |
| Užsakovo patenkinimas | 3 | | Užsakovo nepatenkinimas | 4 | |
| Priklausomybės | Nėra | | Konfliktai | Nėra | |

**15 Lentelė.** Nefunkcinis reikalavimas nr. 4

| Reikalavimas: | NF-4 | Reikalavimo tipas: | 9 | PA: | Visi |
|---|---|---|---|---|---|

| | | | |
|---|---|---|---|
| Aprašymas | Žaidimo valdymas turi būti greitas ir sklandus, o naudotojo valdomi objektai turi valdytis be trikdžių. | | |
| Pagrindimas | Žaidimas turi veikti sklandžiai, kad naudotojas norėtų jį žaisti kuo daugiau. | | |
| Šaltinis | Sistemos užsakovas | | |
| Tinkamumo kriterijai | - | | |
| Užsakovo patenkinimas | 5 | Užsakovo nepatenkinimas | 5 |
| Priklausomybės | Nėra | Konfliktai | Nėra |

### 4.9.4. Reikalavimai veikimo sąlygoms

**16 Lentelė.** Nefunkcinis reikalavimas nr. 5

| | | | | | |
|---|---|---|---|---|---|
| Reikalavimas: | NF-5 | Reikalavimo tipas: | 9 | PA: | Visi |
| Aprašymas | Žaidimą turi būti patogu žaisti tiek naudotojams naudojantiems nešiojamus kompiuterius, tiek stacionarius kompiuterius. | | | | |
| Pagrindimas | Pritraukiamas didesnis naudotojų kiekis apimant šias dvi naudotojų grupes. | | | | |
| Šaltinis | Sistemos užsakovas | | | | |
| Tinkamumo kriterijai | - | | | | |
| Užsakovo patenkinimas | 1 | | Užsakovo nepatenkinimas | 4 | |
| Priklausomybės | Nėra | | Konfliktai | Nėra | |

### 4.9.5. Reikalavimai sistemos priežiūrai

**17 Lentelė.** Nefunkcinis reikalavimas nr. 6

| | | | | | |
|---|---|---|---|---|---|
| Reikalavimas: | NF-6 | Reikalavimo tipas: | 9 | PA: | Visi |
| Aprašymas | Du pirmus žaidimo gyvavimo metus bus leidžiami atnaujinimai bei žaidimo sistemos tvarkymai. | | | | |
| Pagrindimas | Pastoviai atnaujinamas ir tvarkomas žaidimas pritraukia didesnį naudotojų kiekį. | | | | |
| Šaltinis | Sistemos užsakovas | | | | |
| Tinkamumo kriterijai | - | | | | |
| Užsakovo patenkinimas | 3 | | Užsakovo nepatenkinimas | 5 | |
| Priklausomybės | Nėra | | Konfliktai | Nėra | |

### 4.9.6. Reikalavimai saugumui (Security)

**18 Lentelė.** Nefunkcinis reikalavimas nr. 7

| | | | | | |
|---|---|---|---|---|---|
| Reikalavimas: | NF-7 | Reikalavimo tipas: | 9 | PA: | Visi |
| Aprašymas | Žaidimo duomenys neturi būti galimi pakeisti iš išorės (neturi veikti sukčiavimo programos) | | | | |
| Pagrindimas | Skatinamas sąžiningas žaidimas. | | | | |
| Šaltinis | Sistemos užsakovas | | | | |

| | | | |
|---|---|---|---|
| Tinkamumo kriterijai | - | | |
| Užsakovo patenkinimas | 1 | Užsakovo nepatenkinimas | 2 |
| Priklausomybės | Nėra | Konfliktai | Nėra |

## 4.10. Projekto išeiga

### 4.10.1. Atviri klausimai

Sistemos realizavimo metu bus norima išbandyti kuo daugiau skirtingų dirbtinių intelekto algoritmų, kurie šiuo metu jau yra naudojami kituose, populiariuose strateginiuose žaidimuose. Problema – didžioji dauguma populiarių strateginių žaidimų neviešina savo žaidimų kodo, todėl bus sunku surasti jau implementuotų pavzdžių, kuriuos galima būtų greitai perkelti į savo kuriamą žaidimą.

### 4.10.2. Pagamintos sistemos, kurios gali būti nupirktos

Sistemos kūrimui galima būtų nupirkti žaidimų variklio Unity, *premium* Unity Pro versiją, kuri šiek tiek paspartintų žaidimo kūrima, kadangi Pro versija turi papildomų produktyvumą didinančių įrankų bei plėtinių. Unity Pro kaina per mėnesį – nuo 185$ vienam programuotojui.

## 4.11. Naujos problemos

### 4.11.1. Problemos diegimo palinkai

Kompiuteris kuriame bus bandomas diegti žaidimas, gali turėti 32 bitų architektūra grįstą procesorių, todėl įdiegimas gali būti neįmanomas.

### 4.11.2. Įtaka jau instaliuotoms sistemoms

Strateginio žaidimo sistema yra niekaip nesusijusi su jokiomis kitomis sistemomis esančiomis naudotojo kompiuteryje, todel įtaka nedaroma.

### 4.11.3. Neigiamas vartotojų nusiteikimas

Neveikiant arba neteisingai veikiant strateginio žaidimo sistemai, naudotojams reikia pranešti pranešimu pagrindiniame žaidimo meniu lange, kad kai kuriuos funkcijos yra apribotos dėl vykdomų tvarkymų.

### 4.11.4. Kliudantys diegimo aplinkos apribojimai

Kadangi žaidimo naudotojų kompiuteriai nėra vienodi, reikia atsižvelgti, kad žaidimo duomenys po instaliacijos neužpildytų likusiuos laisvos disko vietos, jeigu jos yra likę nedaug.

### 4.11.5. Galimos naujos sistemos sukeltos problemos

Strateginio žaidimo sistema reikalaus nemažai procesoriaus ir vaizdo plokštės resursų, todėl svarbu numatyti, kad šie komponentai nebūtų naudojami 100%, kad naudotojas galėtų turėti įjungtas ir kitas aplikacijas vienu metu.

## 4.12. Uždaviniai

### 4.12.1. Sistemos pateikimo žingsniai (etapai)

Šiam projektui parenkamas tradicinis Krioklio (kaskadinis) proceso modelis. Šis modelis parenkamas todėl, nes kuriamas projektas, kurio visi keliami reikalavimai yra iš anksto žinomi, jų keitimosi tikimybė yra minimali, o pats projekto uždavinys nėra išskirtinis, ir jo sprendimo metodika yra žinoma ir aiški. Šis modelis turi privalumus puikiai tinkančius šio projekto pobūdžiui - modelio procesas yra aiškus ir lengvai koordinuojamas, o reikalaujamos darbo sąnaudos yra minimalios jeigu nenumatomi grįžimai į praeitus modelio etapus.

Modelis bus skirstomas į 4 pagrindinius etapus:

1. Reikalavimų surinkimas ir analizė
2. Planavimas ir projektavimas
3. Kodavimas ir testavimas
4. Integravimas, eksploatacija ir palaikymas

### 4.12.2. Vystymo etapai

Strateginio žaidimo sistemos  planavimo ir kūrimo grafikas išskirstytas į 4 pagrindinius etapus.

**19 Lentelė.** Sistemos vystymo etapų grafikas

| **Vystymo etapas** | **Aprašymas** | **Terminas** |
|---|---|---|
| Reikalavimų ir architektūros specifikavimas | Sistemos funkcinių, nefunkcinių ir techninių reikalavimų rinkimas ir analizė, sistemos projektavimas. | 2023-09-01 – 2024-04-14 |
| Realizacija | Sistemos kodo implementacija pagal surinktus reikalavimus ir architektūros specifikaciją | 2024-04-14 – 2024-10-01 |
| Testavimas | Sistemos funkcijų testavimas | Vykdomas iteratyviai kartu su realizacijos etapu. |
| Dokumentacija | Sistemos funkcijų aprašymas ir įvairių scenarijų dokumentavimas. | 2024-10-01 – 2024-12-01 |

## 4.13. Pritaikymas

### 4.13.1. Reikalavimai esamų duomenų perkėlimui

Kuriamas naujas žaidimas, todėl esamų duomenų, kuriuos reikėtų perkelti, nėra.

### 4.13.2. Reikalingas duomenų transformavimas perkeliant į naują sistemą

Kuriamas naujas žaidimas, todėl esamų duomenų, kuriuos reikėtų transformuoti, nėra.

## 4.14. Rizikos

### 4.14.1. Galimos sistemos kūrimo rizikos

**20 Lentelė.** Galimos sistemos kūrimo rizikos

| Rizikos faktorius | Tikimybinis įvertinimas[1] |
|---|---|
| Reikalavimų specifikacijos pasikeitimai realizavimo fazėje | 8 |
| Architektūriniai pasikeitimai realizavimo fazėje | 8 |
| Nepakankamas kiekis dirbtinio intelekto modelių pavyzdžių rastų realizavimo fazėje | 4 |
| Sistemos greitaveikos reikalavimų neišpildymas realizavimo fazėje | 6 |

### 4.14.2. Atsitiktinumų (rizikų) planas

**21 Lentelė.** Atsitiktinumų (rizikų) planas

| Rizikos faktorius | Problemos sprendimas |
|---|---|
| Reikalavimų specifikacijos pasikeitimai realizavimo fazėje | Kad keisti reikalavimus tokioje fazėje jie turi būti tikrai labai svarbūs |
| Architektūriniai pasikeitimai realizavimo fazėje | Kad keisti architektūrą šioje fazėje, pasikeitimai turėtų kelis kartus pagerinti sistemos veikimą |
| Nepakankamas kiekis dirbtinio intelekto modelių pavyzdžių rastų realizavimo fazėje | Giliau patobulinti rastus pavyzdžius, pabandyti skirtingas dirbtinio intelekto modelių kombinacijas |
| Sistemos greitaveikos reikalavimų neišpildymas realizavimo fazėje | Labiausiai greitaveiką įtakojančių sistemos dalių atradimas, svarbiausių sistemos dalių prioritezavimas, lėtai veikiančių dalių perrašymas. |

## 4.15. Kaina

**22 Lentelė.** Detalus kainos išskirstymas

| Išlaidos | Vienetas | Vienetų skaičius | Vieneto kaina (įskaitant mokesčius, Eur | Viso, Eur |
|---|---|---|---|---|
| 1. Žmonių ištekliai | | | | |
| Programuotojas | Mėnesis | 18 * 0.2 etato | 3450 | 12420 |
| Vadovas | Mėnesis | 18 * 0.1 etato | 3500 | 6300 |
| *Iš viso žmonių išteklių* | | | | 18720 |
| 2. Įranga ir prekės | | | | |
| Kompiuteris | Vienetas | 1 | 1500 | 1500 |
| Periferinė įranga kompiuteriui | Vienetas | 1 | 700 | 700 |

| | | | | |
|---|---|---|---|---|
| *Iš viso Įranga ir prekės* | | | | 2200 |
| 3. Biuro išlaikymas | Mėnesis | 18 | 560 | 10080 |
| Elektros, interneto, šildymo,telefono,nuomos išlaidos | Vienetas | 18 | 200 | 3600 |
| *Iš viso biuro išlaikymas* | | | | 13680 |
| 4. Programinė įranga | | | | |
| Visual Studio 2022 Professional | Vienetas | 1 | 89 | 89 |
| *Iš viso programinė įranga* | | | | 89 |
| 5. Viso tiesioginiai projekto kaštai | | | | 34689 |

### 4.16. Vartotojo dokumentacija ir apmokymas

Reikalinga parengti detalią strateginio žaidimo sistemos grafinės vartotojo sąsajos dokumentaciją su detaliai aprašytais veikimo principais ir diagramomis, parodančiomis kaip kiekviena sąsajos dalis siejasi su šalia jos esančiomis dalimis.

Žaidimo veikimo principus detaliai apibūdinančią dokumentaciją parengs ir naujins programuotojas Tadas Laurinaitis.

Dokumentacija iš pradžių bus aprašyta PDF formatu, su planu vėliau ją perkelti į atskirą wikipedia subdomeną.

## 5. ARCHITEKTŪROS SPECIFIKACIJA

### 5.1. Įvadas

#### 5.1.1. Dokumento paskirtis

Šio dokumento paskirtis yra apibrėžti detalią, kuriamo projekto – strateginio žaidimo, architektūros specifikaciją. Šiame dokumente pateikiamas projekto išanalizuotos reikiamos logikos architektūrinis aprašas tolimesniems projekto vykdymo darbams. Dokumento vartotojai – projektą vykdantys programuotojai ir architektai. Dokumentas vartotojams padės sklandžiai komunikuoti tarpusavyje dėl architektūrinių sprendimų.

#### 5.1.2. Apžvalga

Šis architektūros specifikacijos dokumentas pateikia projekto metu atliktus architektūrinius sprendimus.

2. Architektūros pateikimas - nurodomi architektūros pateikimo aprašymai ir reikalingi elementai.

3. Architektūros tikslai ir apribojimai - aprašomi projekto programinės įrangos tikslai, reikalavimai ir apribojimai.

4. Panaudojimo atvejų vaizdas – pateikiama panaudojimo atvejų diagrama ir aprašomas kiekvienas panaudojimo atvejis lentelių forma.

5. Sistemos statinis vaizdas – sistemos išskaidymas paketų diagramomis. Kiekvienas paketas turi esminių klasių diagramą ir trumpą aprašymą.

6. Sistemos dinaminis vaizdas – sistemos esminio funkcionalumo aprašymas bei specifikavimas naudojantis veiklos, būsenų ir sekų diagramomis.

7. Išdėstymo vaizdas – aprašymas ir sistemos komponentų diagrama.

8. Duomenų vaizdas – tuščias skyrius, kadangi strateginio žaidimo sistema nenaudos tradicinių duomenų bazių.

9. Kokybė – nurodomas galimas projekto metu kuriamos strateginio žaidimo sistemos išplečiamumas ir kokybės užtikrinimo būdai.

## 5.2. Architektūros pateikimas

Architektūros specifikacija pateikiama naudojantis žmonėms skaitomu tekstu, aprašytu Microsoft Word įrankiu, bei UML kalbos diagramomis, naudojant MagicDraw grafinio atvaizdavimo įrankį.

Šiuo būdu bus pateikiami žemiau nurodyti vaizdai su diagramomis:

- Panaudojimo atvejų vaizdas:
  - Panaudojimo atvejų diagrama
- Statinis vaizdas:
  - Paketų diagrama
  - Klasių diagramomis.
- Dinaminis vaizdas:
  - Veiklos diagramomis
  - Būsenų diagrama
  - Sekų diagramomis.
- Išdėstymo vaizdas:
  - Išdėstymo diagrama.

## 5.3. Architektūros tikslai ir apribojimai

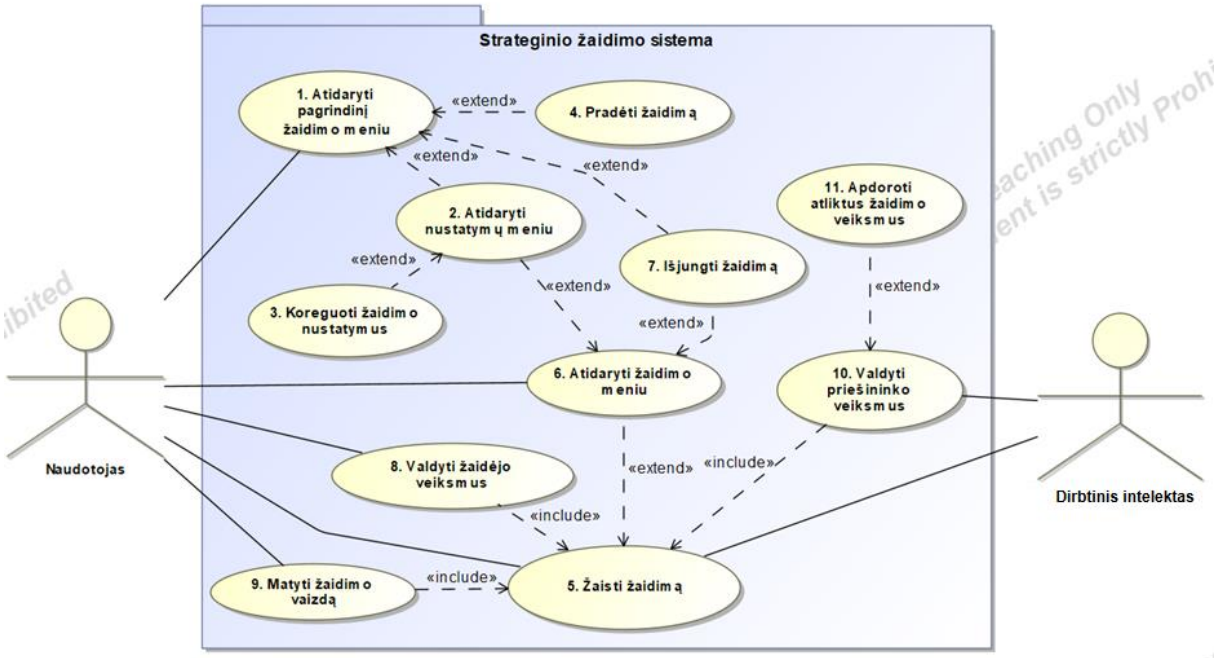
Reikalavimai, įtakojantys projekto architektūrą:

- Sistema turi būti pritaikyta įrenginiams naudojantiems Windows 10 arba naujesnę Windows operacinę sistemą
- Žaidimo grafinėje sąsajos elementai turi įsijungti per mažiau nei 1s nuo paspaudimo.
- Dirbtinio intelekto modeliai ar jų kombinacijos sugeba priimti tinkamus sprendimus žaidimo metu.
- Žaidimo valdymas turi būti greitas ir sklandus, o naudotojo valdomi objektai turi valdytis be trikdžių.
- Žaidimą turi būti patogu žaisti tiek naudotojams naudojantiems nešiojamus kompiuterius, tiek stacionarius kompiuterius.
- Du pirmus žaidimo gyvavimo metus bus leidžiami atnaujinimai bei žaidimo sistemos tvarkymai.

– Žaidimo duomenys neturi būti galimi pakeisti iš išorės (neturi veikti sukčiavimo programos)

## 5.4. Panaudojimo atvejų vaizdas

Žemiau pateiktame paveikslėlyje (**1 pav.**) pateikiami esminiai projekto sistemos panaudojimo atvejai ir jų ryšiai. Po paveikslėliu pateiktose lentelėse (**1-11 Lentelės**), tie patys panaudojimo atvejai ir ryšiai tarp jų yra aprašomi detaliau.




**1 pav. Strateginio žaidimo panaudojimo atvejų diagrama**

**1 Lentelė.** Panaudojimo atvejis nr. 1

| |
|---|
| 1. PANAUDOJIMO ATVEJIS: Atidaryti pagrindinį žaidimo meniu<br>**Vartotojas/Aktorius:** Naudotojas<br>**Aprašas:** Atidaromas pagrindinis žaidimo meniu.<br>**Prieš sąlyga:** Naudotojas įjungė žaidimą.<br>**Sužadinimo sąlyga:** Naudotojas paspaudė pagrindinio žaidimo meniu įjungimo mygtuką.<br>**Po-sąlyga:** Atidaromas pagrindinis žaidimo meniu |

**2 Lentelė.** Panaudojimo atvejis nr. 2

| |
|---|
| 2. PANAUDOJIMO ATVEJIS: Atidaryti nustatymų meniu<br>**Vartotojas/Aktorius:** Naudotojas<br>**Aprašas:** Atidaromas nustatymų meniu langas.<br>**Prieš sąlyga:** Naudotojas atidarė pagrindinį žaidimo meniu.<br>**Sužadinimo sąlyga:** Naudotojas paspaudė nustatymų meniu įjungimo mygtuką.<br>**Po-sąlyga:** Atidaromas nustatymų meniu langas. |

**3 Lentelė.** Panaudojimo atvejis nr. 3

| |
|---|
| 3. PANAUDOJIMO ATVEJIS: Koreguoti žaidimo nustatymus |

**Vartotojas/Aktorius:** Naudotojas

**Aprašas:** Koreguojami žaidimo nustatymai – garso, grafikos ir sunkumo lygiai.

**Prieš sąlyga:** Naudotojas atidarė nustatymų meniu langą.

**Sužadinimo sąlyga:** Naudotojas pakeitė garso, grafikos arba sunkumo lygį naudodamasis grafine sąsaja.

**Po-sąlyga:** Žaidimo nustatymai pakeičiami į naujai pasirinktus.

**4 Lentelė.** Panaudojimo atvejis nr. 4

4. PANAUDOJIMO ATVEJIS: Pradėti žaidimą

**Vartotojas/Aktorius:** Naudotojas

**Aprašas:** Pradedamas žaidimas.

**Prieš sąlyga:** Naudotojas atidarė pagrindinį žaidimo meniu.

**Sužadinimo sąlyga:** Naudotojas paspaudė žaidimo pradėjimo mygtuką.

**Po-sąlyga:** Įjungiamas pagrindinis žaidimo langas ir pradedamas žaidimas.

**5 Lentelė.** Panaudojimo atvejis nr. 5

5. PANAUDOJIMO ATVEJIS: Žaisti žaidimą

**Vartotojas/Aktorius:** Naudotojas, dirbtinis intelektas

**Aprašas:** Žaidžiamas žaidimas tarp žaidėjo ir dirbtinio intelekto.

**Prieš sąlyga:** Naudotojas atidarė pagrindinį žaidimo meniu.

**Sužadinimo sąlyga:** Naudotojas paspaudė žaidimo pradėjimo mygtuką.

**Po-sąlyga:** Žaidžiamas žaidimas tarp žaidėjo ir dirbtinio intelekto.

**6 Lentelė.** Panaudojimo atvejis nr. 6

6. PANAUDOJIMO ATVEJIS: Atidaryt žaidimo meniu

**Vartotojas/Aktorius:** Naudotojas

**Aprašas:** Atidaromas meniu žaidimo metu, leidžiantis keisti žaidimo nustatytumus ir išjungti žaidimą.

**Prieš sąlyga:** Naudotojas atidarė pagrindinį žaidimo meniu.

**Sužadinimo sąlyga:** Naudotojas paspaudė žaidimo pradėjimo mygtuką.

**Po-sąlyga:** Įjungiamas žaidimo meniu žaidimo metu.

**7 Lentelė.** Panaudojimo atvejis nr. 7

7. PANAUDOJIMO ATVEJIS: Išjungti žaidimą

**Vartotojas/Aktorius:** Naudotojas

**Aprašas:** Išjungiamas žaidimas.

**Prieš sąlyga:** Naudotojas atidarė žaidimo meniu.

**Sužadinimo sąlyga:** Naudotojas paspaudė žaidimo išjungimo mygtuką.

**Po-sąlyga:** Žaidimas išjungiamas.

**8 Lentelė.** Panaudojimo atvejis nr. 8

8. PANAUDOJIMO ATVEJIS: Valdyti žaidėjo veiksmus

**Vartotojas/Aktorius:** Naudotojas

**Aprašas:** Valdomi žaidėjo veiksmai, naudojantis įvesties įrenginiais ir atsižvelgiant į matomą žaidimo vaizdą.

**Prieš sąlyga:** Žaidimas yra pradėtas.

**Sužadinimo sąlyga:** Naudotojas paspaudė bent vieną iš registruotų žaidėjo valdymo klavišų klaviatūroje arba pelėje.

**Po-sąlyga:** Valdomi žaidėjo veiksmai priklausomai nuo paspaustų įvesties įrenginių klavišų.

**9 Lentelė.** Panaudojimo atvejis nr. 9

| 9. PANAUDOJIMO ATVEJIS: Matyti žaidimo vaizdą |
|---|
| **Vartotojas/Aktorius:** Naudotojas |
| **Aprašas:** Matomas žaidimo vaizdas išvesties įrenginyje (monitoriuje). |
| **Prieš sąlyga:** Naudotojas įjungė žaidimo aplikaciją. |
| **Sužadinimo sąlyga:** - |
| **Po-sąlyga:** Matomas visas žaidimo vaizdas bei pasinkti meniu. |

**10 Lentelė.** Panaudojimo atvejis nr. 10

| 10. PANAUDOJIMO ATVEJIS: Valdyti priešininko veiksmus |
|---|
| **Vartotojas/Aktorius:** Dirbtinis intelektas |
| **Aprašas:** Valdomi priešininko veiksmai atsižvelgiant į apdorotus jau atliktus žaidimo veiksmus. |
| **Prieš sąlyga:** Žaidimas yra pradėtas. |
| **Sužadinimo sąlyga:** Naudotojas paspaudė žaidimo pradėjimo mygtuką. |
| **Po-sąlyga:** Priešininkas yra valdomas atsižvelgiant į veiksmus, kuriuos reikia atlikti. |

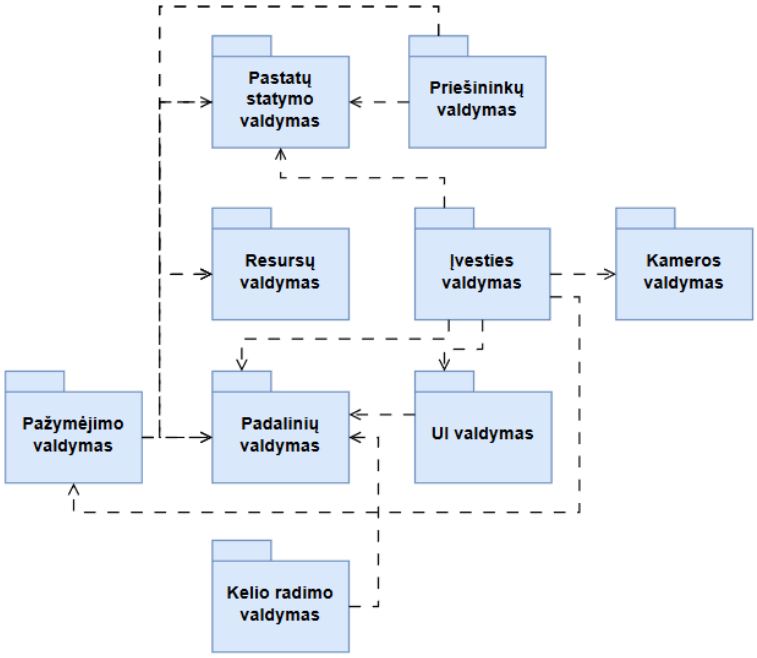
**11 Lentelė.** Panaudojimo atvejis nr. 11

| 11. PANAUDOJIMO ATVEJIS: Apdoroti atliktus žaidimo veiksmus |
|---|
| **Vartotojas/Aktorius:** Dirbtinis intelektas |
| **Aprašas:** Apdorojami žaidėjo ir dirbtinio intelekto atlikti veiksmai bei apskaičiuojami geriausi ir naudingiausi sekantys veiksmai. |
| **Prieš sąlyga:** Žaidimas yra pradėtas. |
| **Sužadinimo sąlyga:** Naudotojas paspaudė žaidimo pradėjimo mygtuką. |
| **Po-sąlyga:** Pastoviai apdorojama žaidėjo ir dirbtinio intelekto veiksmų informacija ir parenkami geriausi ir naudingiausi sekantys veiksmai. |

## 5.5. Sistemos statinis vaizdas

### 5.5.1. Apžvalga

Projekto statinis vaizdas skiriamas į paketus, matomus žemiau pateiktame paveikslėlyje (**2 pav.**). 5.2 skyriuje “Paketų detalizavimas”, šie paketai bus detalizuojami aprašymais ir klasių diagramomis.

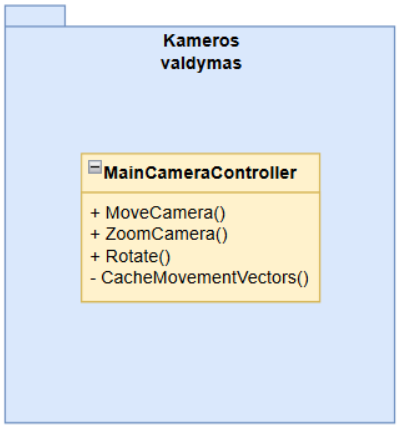



**2 pav. Sistemos paketų diagrama**

### 5.5.2. Paketų detalizavimas

**Paketas “Kameros valdymas”**

Šio paketo klasės yra atsakingos už pagrindinės žaidimo kameros valdymą. Paketo klasių struktūra pateikiama žemiau esančiame paveikslėlyje (**3 pav.**).

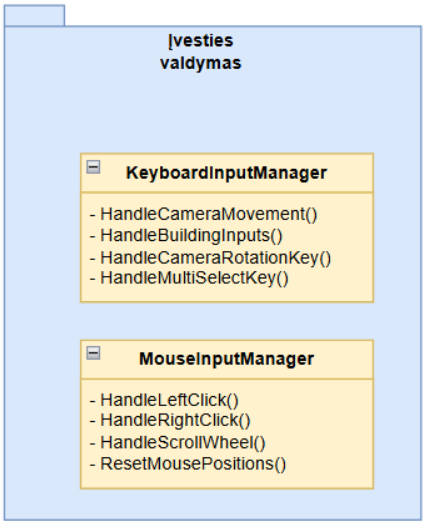



**3 pav. Paketo „Kameros valdymas“ klasių diagrama**

Paveikslėlyje parodyta klasė atlieka šias funkcijas: kameros pozicijos keitimas, kameros rotacijos keitimas ir kameros vaizdo padidinimas.

**Paketas „Įvesties valdymas“**

Šio paketo klasės yra atsakingos už žaidėjo įvesties signalų, pateikiamų įvesties įrenginiais, apdorojimą. Paketo klasių struktūra pateikiama žemiau esančiame paveikslėlyje (**4 pav.**)

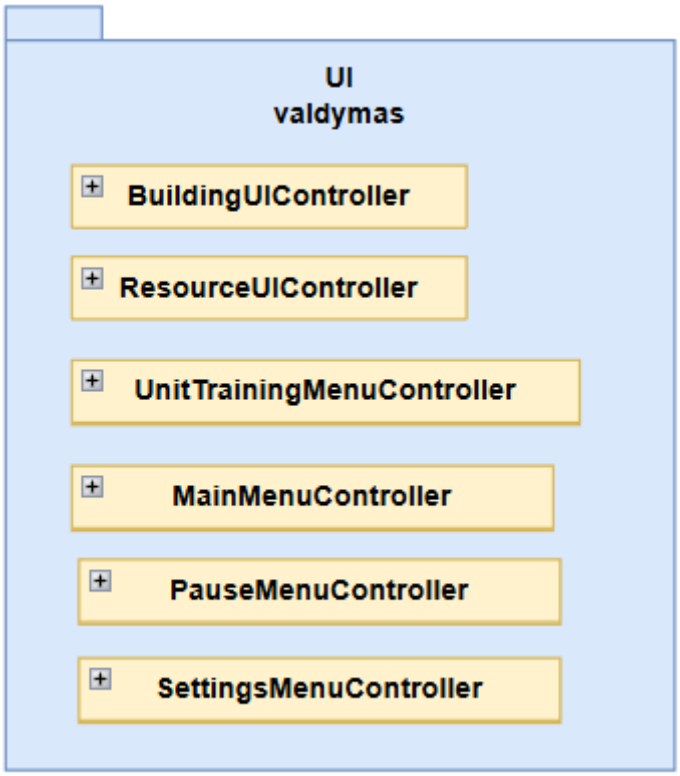



**4 pav. Paketo „Įvesties valdymas“ klasių diagrama**

Aukščiau pateiktame paveikslėlyje parodytos klasės atlieka šias funkcijas: gauna žaidėjo įvestį, atsikiria įvesties tipą, prilyginą šį tipa norimam atlikti veiksmui ir galiausiai siunčia veiksmus į kituose paketuose esančius valdiklius.

### Paketas „UI valdymas“

Šio paketo klasės yra atsakingos vartotojo grafinės sąsajos valdymą. Unity žaidimų variklyje, grafinės vartotojo sąsajos įgyvendinimas vykdomas drobių (angl. Canvas) pagrindu. Kiekviena drobė turi savo elementus, kurie gali būti skirtingų tipų. Paketo klasių struktūra pateikiama žemiau esančiame paveikslėlyje (**5 pav.**).

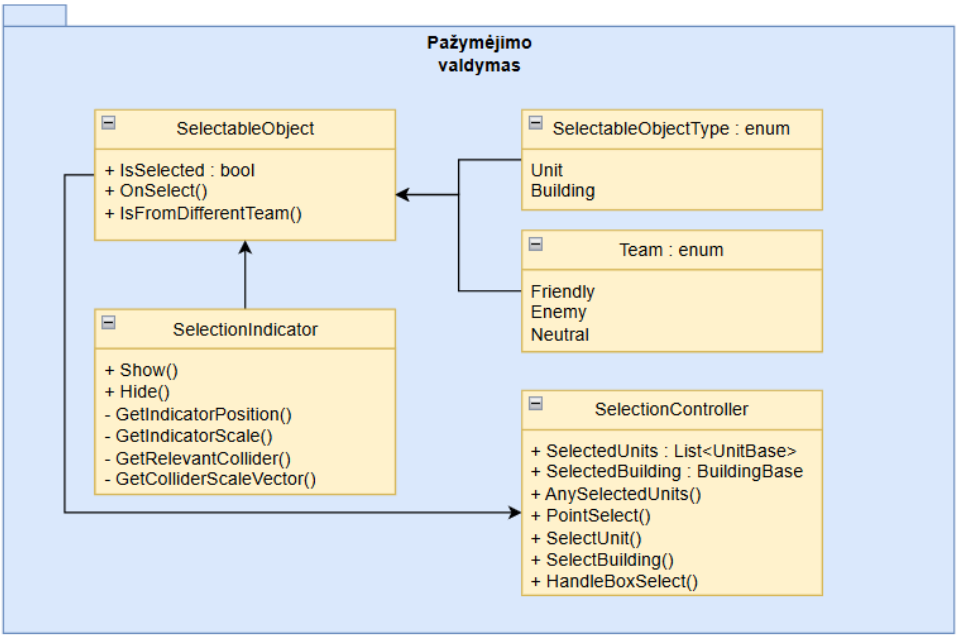



**5 pav. Paketo „UI“ klasių diagrama**

Aukščiau pateiktame paveikslėlyje parodytos klasės atlieka šias funkcijas: atlieka drobių kūrimą, atnaujinimą bei keitimą, drobių elementų kūrimą ir atnaujinimą. Išskiriami šeši drobių elementų tipai: tekstinis langas (*angl. TextBox*), mygtukas (*angl. Button*), mygtukas su tekstiniu langu (*angl. ButtonWithTextBox*), nustatymų langas (*angl. SettingsBox*), slankiojimo elementas (*angl. Slider*) ir slinkimo elementas (*angl. Scroller*).

### Paketas „Pažymėjimo valdymas“

Šio paketo klasės yra atsakingos už padalinių ir pastatų pažymėjimo logikos valdymą. Šią pažymėjimo logiką taip pat naudoja ir kiti paketai, pagrinde padalinių ir pastatų valdymo paketai. Paketo klasių struktūra pateikiama žemiau esančiame paveikslėlyje (**6 pav.**).

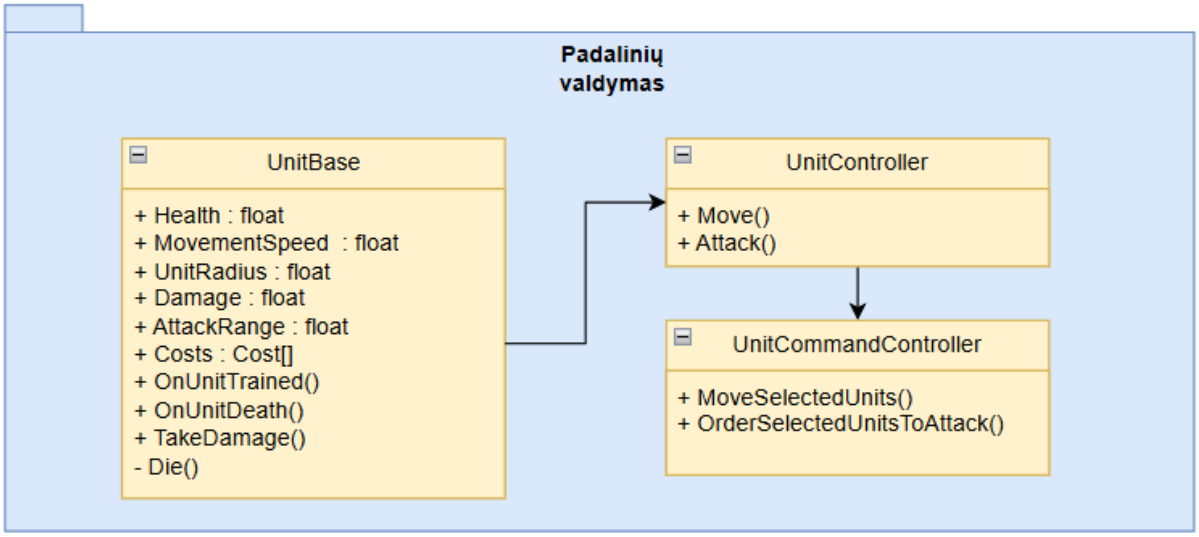



**6 pav. Paketo „Pažymėjimo valdymas“ klasių diagrama**

Aukščiau parodytoje diagramoje matoma, kad klasės valdo tiek padalinių, tiek pastatų pažymėjima. Surinkti pažymėjimų duomenys yra perduodami į kituose paketuose esančias klases.

## Paketas „Padalinių valdymas“

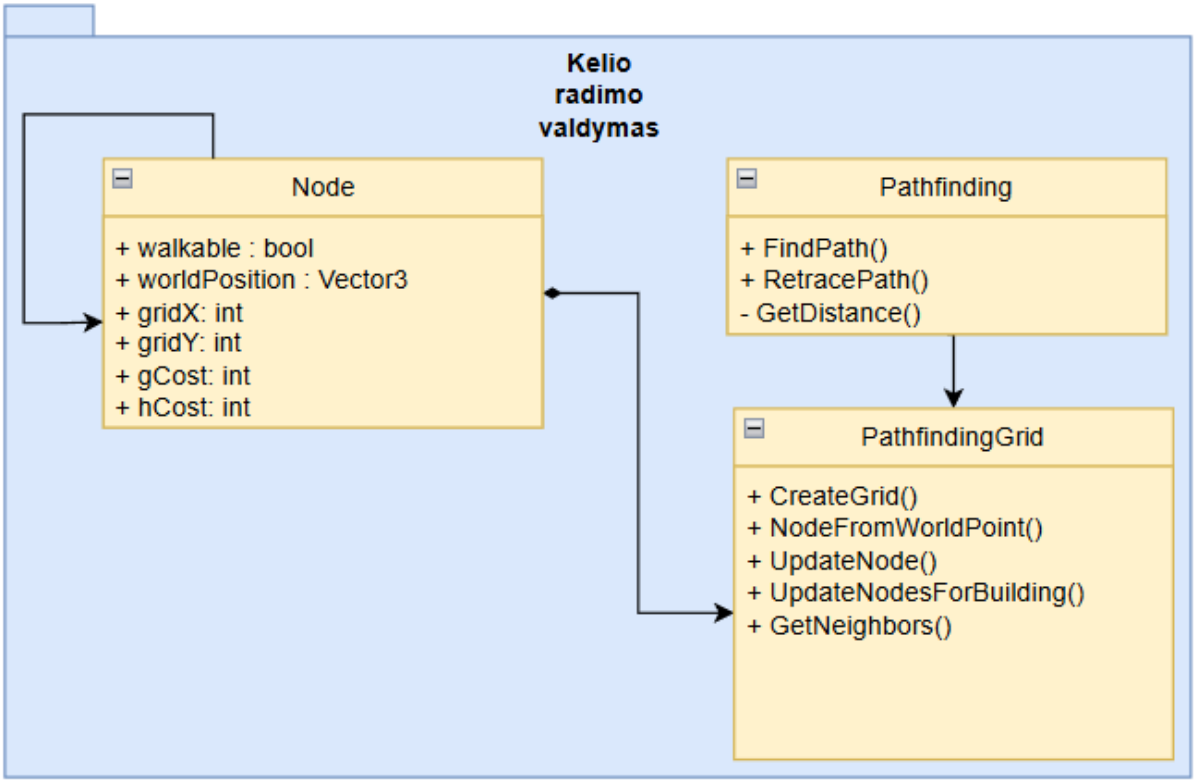
Šio paketo klasės yra atsakingos už padalinių vaikščiojimo ir atakavimo logikos valdymą. Paketo klasių struktūra pateikiama žemiau esančiame paveikslėlyje (**7 pav.**).




**7 pav. Paketo „Padalinių valdymas“ klasių diagrama**

## Paketas „Kelio radimo valdymas“

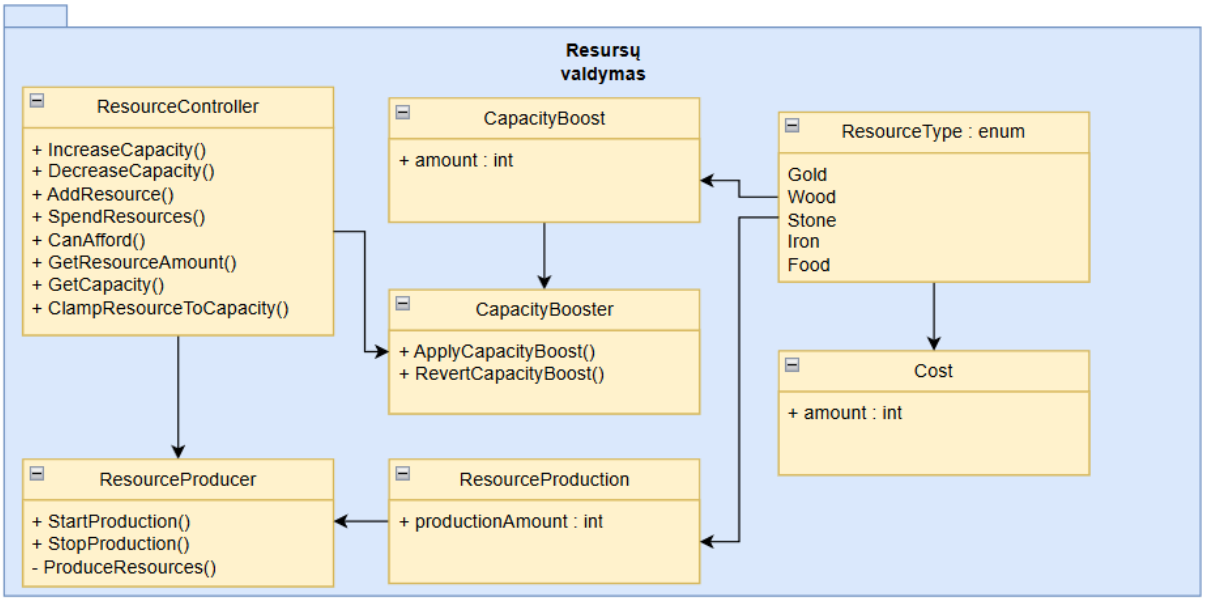
Šio paketo klasės yra atsakingos už kelio radimą žemėlapyje. Pakete esančių klasių duomenis naudoja kiti paketai, pagrinde padalinių ir pastatų paketai. Paketo klasių struktūra pateikiama žemiau esančiame paveikslėlyje (**8 pav.**).




**8 pav. Paketo „Kelio radimo valdymas“ klasių diagrama**

### Paketas „Resursų valdymas“

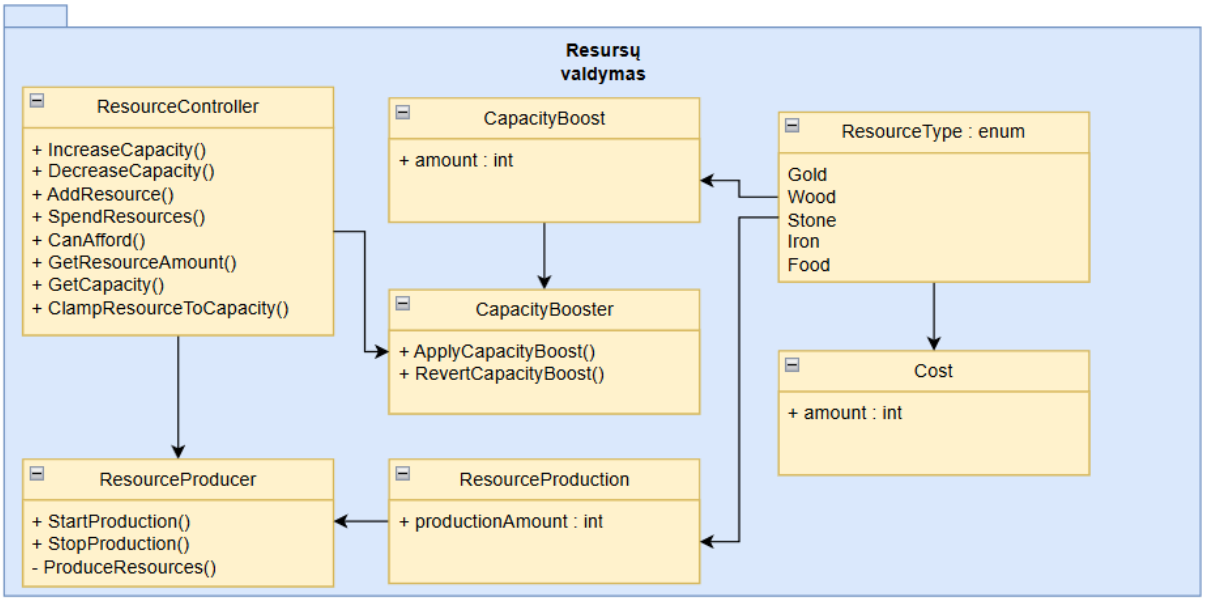
Šio paketo klasės yra atsakingos už viską, kas susiję su resursais – resursų gaminimą, resursų vietos plėtimą, resursų leidimą ir resursų kiekio tikrinimą. Paketo klasės yra plačiai naudojamos kituose paketuose. Paketo klasių struktūra pateikiama žemiau esančiame paveikslėlyje (**9 pav.**).




**9 pav. Paketo „Kelio radimo valdymas“ klasių diagrama**

## Paketas „Resursų valdymas“

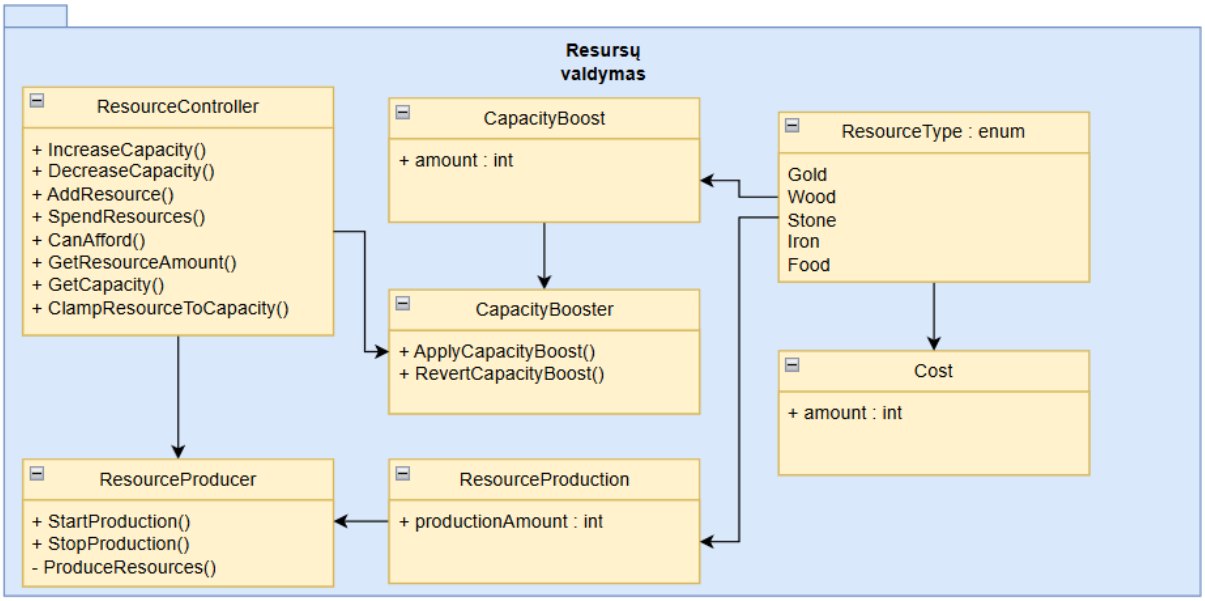
Šio paketo klasės yra atsakingos už viską, kas susiję su resursais – resursų gaminimą, resursų vietos plėtimą, resursų leidimą ir resursų kiekio tikrinimą. Paketo klasės yra plačiai naudojamos kituose paketuose. Paketo klasių struktūra pateikiama žemiau esančiame paveikslėlyje (**10 pav.**).




**10 pav. Paketo „Kelio radimo valdymas“ klasių diagrama**

## Paketas „Pastatų statymo valdymas“

Šio paketo klasės yra atsakingos už viską, kas susiję su pastatais – jų statymą, vietos ieškojimą, sugriovimą, veiksmų, tokių kaip padalinių gaminimas, atlikimą. Paketo klasių struktūra pateikiama žemiau esančiame paveikslėlyje (**11 pav.**).



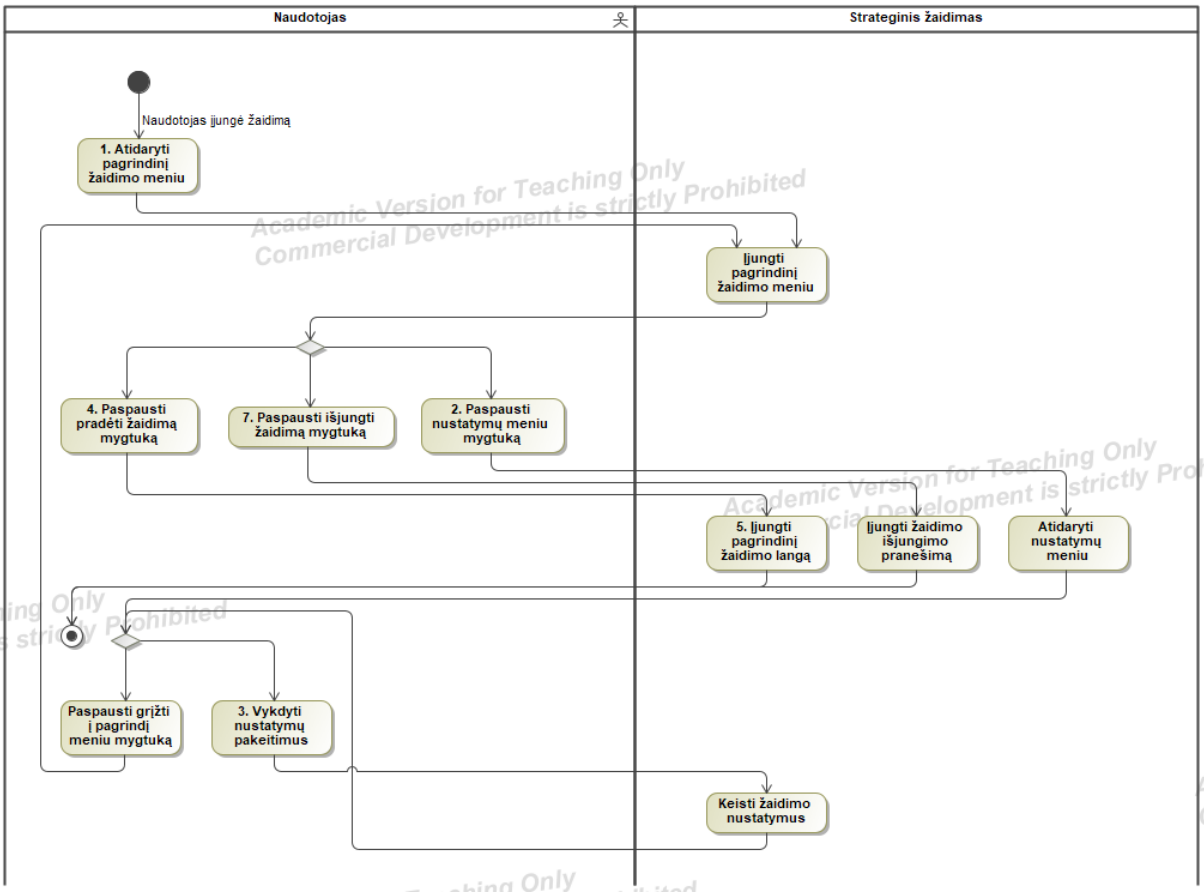

**11 pav. Paketo „Kelio radimo valdymas“ klasių diagrama**

## 5.6. Sistemos dinaminis vaizdas

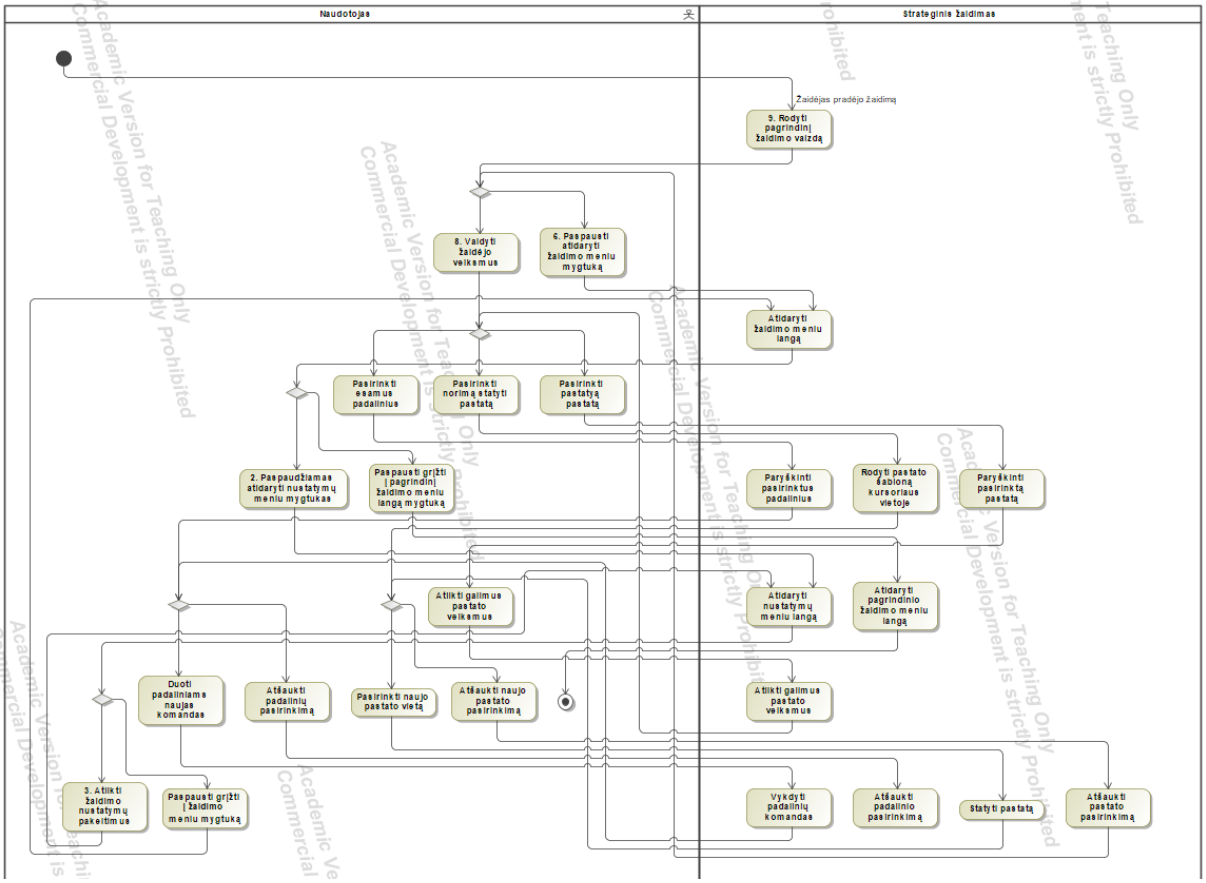
Šiame skyriuje pateikiamos veiklos, būsenų ir sekų diagramos. Diagramos palengvina bendro strateginio žaidimo veikimo proceso supratimą ir leidžia greičiau perprasti ryšį tarp skirtingų sistemos dalių.

### 5.6.1. Veiklos diagramos

Žemiau pateiktos veiklos diagramos pasirinktiems esminiams panaudojimo atvejams – žaidimo pagrindinio meniu navigacijai ir pagrindiniam žaidimo funkcionalumui, sutinkamam žaidimo metu. (**12 ir 13 pav.**).



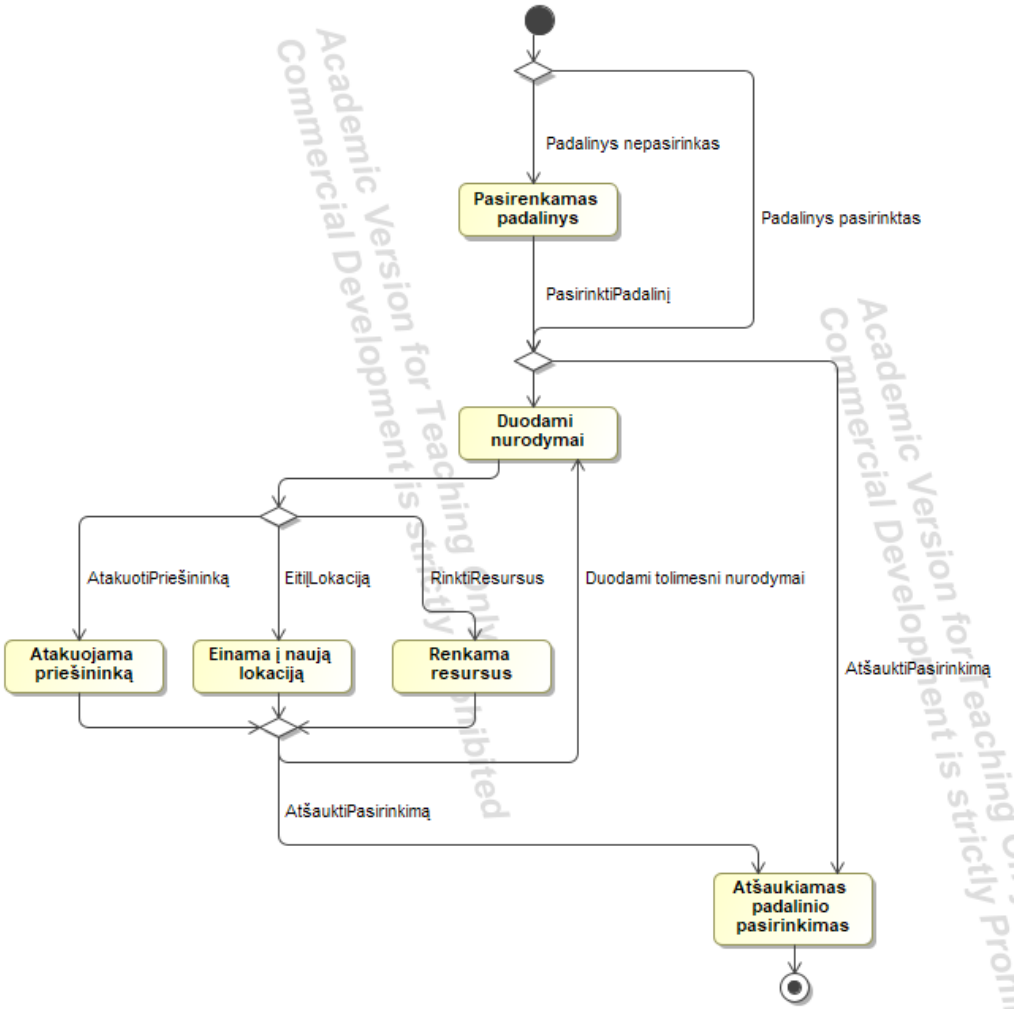

**12 pav. Žaidimo pagrindinio meniu navigavimo veiklos diagrama**




**13 pav. Žaidimo pagrindinio funkcionalumo, sutinkamo žaidimo metu, veiklos diagrama**

### 5.6.2. Būsenų diagramos

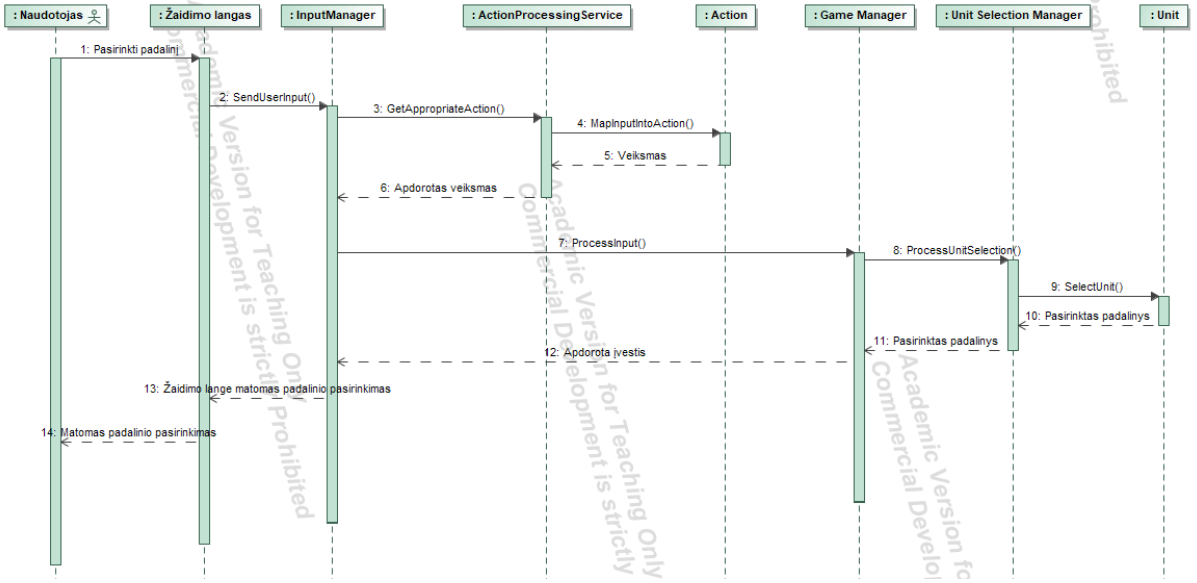
Žemiau pavaizduota būsenų diagrama (**14 pav.**) apibrėžia kiekvieno padalinio būsenas žaidimo metu. Padalinio būsenos kinta priklausomai nuo žaidėjo pasirinktų veiksmų su tuo padaliniu. Padalinio būsenos visada prasideda padalinio pasirinkimu, ir visada baigiasi padalinio pasirinkimo atšaukimu.



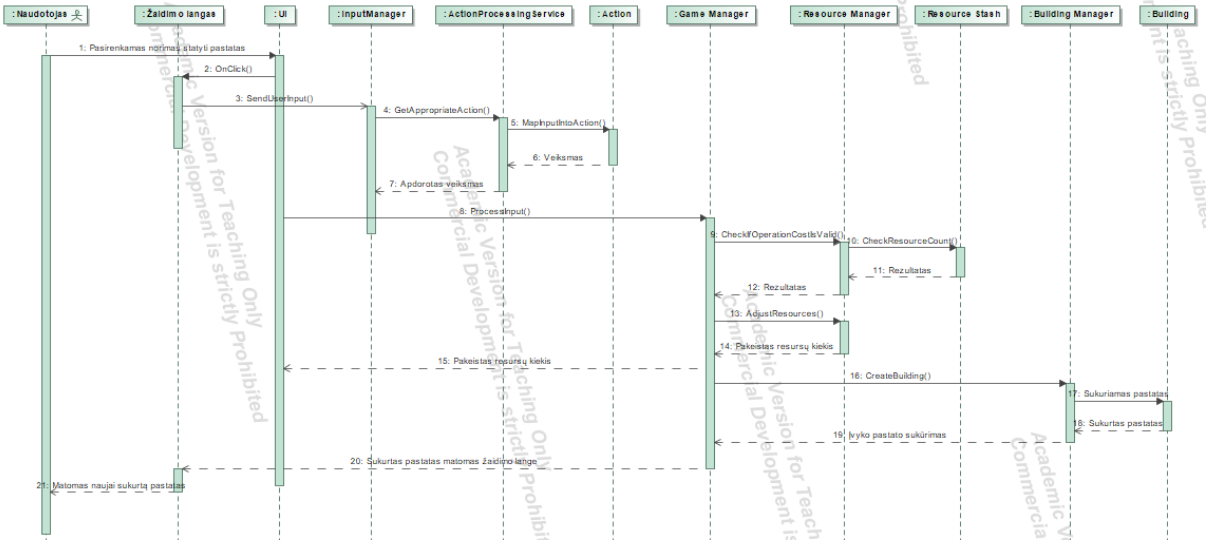

**14 pav. Padalinio būsenų diagrama**

### 5.6.3. Sekų diagramos

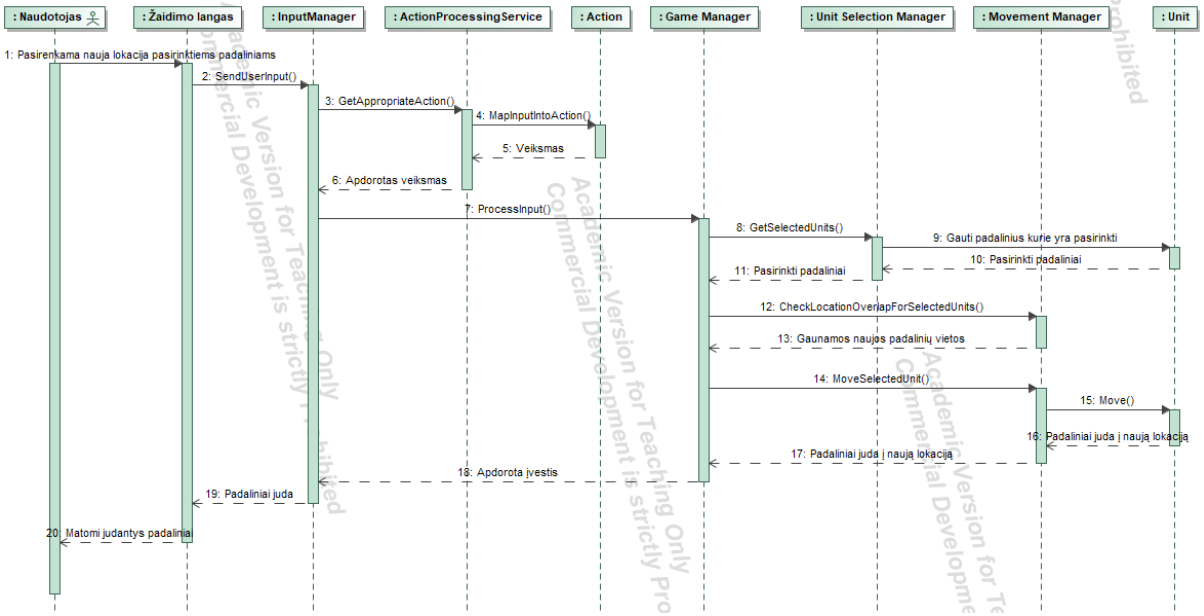
Žemiau esančiuose paveikslėliuose (**15-17 pav.**) pateikiamos sekų diagramos pasirinktam strateginio žaidimo funkcionalumui. Svarbu pastebėti, kad sekų diagramose neapibrėžiami elementarūs strateginio žaidimo panaudojimo atvejai, o apibrėžiamas sudėtingesnis funkcionalumas, iš kurio susideda panaudojimo atvejai „8. Valdyti žaidėjo veiksmus“ ir „5. Žaisti žaidimą“. Funkcionalumas, kuris turi beveik identiškas sekų diagramas (galbūt su tam tikrais nežymiais pakitimais), kaip ir žemiau apibrėžtos sekų diagramos, taip pat nėra apibrėžtas.




**15 pav. Padalinio pasirinkimo sekų diagrama**



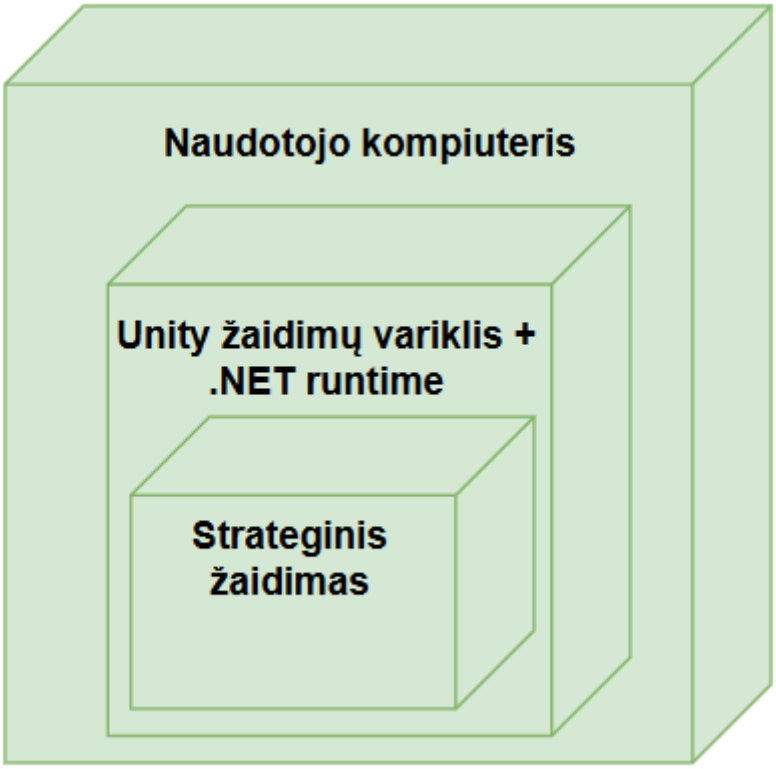

**16 pav. Pastatų statymo sekų diagrama**




**17 pav. Padalinių judėjimo į pasirinktą lokaciją sekų diagrama**

### 5.7. Išdėstymo (deployment) vaizdas

Projekto sistemos išdėstymo vaizdas pateikiamas įšdėstymo diagrama. Visos žaidimo dalys veikia naudotojo kompiuteryje – strateginis žaidimas veikia Unity žaidimų variklyje, kartu su visa žaidimo logika ir dirbtinio intelekto algoritmais.




**18 pav. Strateginio žaidimo išdėstymo diagrama**

### 5.8. Duomenų vaizdas

Projekto metu kuriama sistema – strateginis žaidimas nenaudos duomenų bazės. Žaidimo duomenys žaidimo metu bus saugomi darbinėje atmintyje (RAM) Unity žaidimo variklio pagalba automatiniu būdu. Pabaigus žaidimą, žaidimo duomenys bus išsaugomi atsikarame faile, užfiskuosiančiame paskutinę žaidimo būseną žmonėms neskaitomu formatu – sugeneruotu artefaktu.

### 5.9. Kokybė

Strateginis žaidimas su tam tikrais nedideliais sisteminio kodo pakeitimais, nekeičiant žaidimo architektūros, taip pat teoriškai gali būti padarytas žaisti ir ant kitų platformų, palaikančių Unity žaidimo variklių, tačiau šis žingsnis nėra planuojamas.

Strateginio žaidimo kodo kokybei užtrikrinti bus naudojami tiek statinės, tiek dinaminės kodo analizės įrankiai.

# 6. TESTAVIMO PLANAS

## 6.1. Įvadas

### 6.1.1. Testavimo tikslai ir objektai

Pagrindinis testavimo tikslas – užtikrinti kad kiekvienas kuriamos sistemos komponentas veikia tinkamai tiek vienas, tiek integruotas į komponentų visumą. Pagrindinis testavimo tikslas sudarytas iš šių dalių:

- Žaidimo logika ir mechanikos – užtikrinama kad pagrindiniai žaidimo elementai (žaidėjo judėjimas, resursų valdymas ir žaidime vykstančios sąveikos tarp žaidėjo ir žaidimo) veikia nuosekliai, žaidėjui atliekant skirtingus veiksmus.
- Dirbtinio intelekto elgsena – užtikrinama kad dirbtinio intelekto valdomi komponentai dinamiškai reaguoja į žaidėjo veiksmus, ir kad dirbtinio intelekto priimami sprendimai yra priimtinose normose.
- Žaidimo grafinės sąsajos elementų (UI) funkcionalumas – testuojami žaidimo UI elementai, tokie kaip meniu, meniu valdymo mygtukai ir žaidimo UI elementai rodantys tam tikrus žaidimo rodiklius, kad užtikrinti, jog šie atitinka žaidėjų patogaus naudojimo reikalavimus ir yra tiek greitai reaguojantys, tiek rodantys tikslią informaciją.
- Žaidimo našumo metrikos – užtikrinamas žaidimo stabilumas, greitas atsako laikas ir tinkamas kompiuterio resursų paskirstymas, ypač intesyvių žaidimo scenarijų metu.

### 6.1.2. Testavimo apimtis ir tipai

Naudojami testų tipai ir jų apimamos sritys:

- Funkciniai testai – šio testavimo tipo metu pagrindinis dėmesys sutelktas į kiekvieno žaidimo funkcionalumo atitikimo reikalavimams patikrinimą. Į šį testavimą įtraukiama: dirbtinio intelekto atliekamų veiksmų veikimo, žaidimo mechanikų veikimo, žaidimo reagavimo į žaidėjo veiksmus, korektiškumo užtikrinimas.
- Vienetų testai – dėmesys sutelkiamas į atskirus žaidimo komponentus, kuriuos galima ištestuoti po vieną. Šio testavimo metu siekiama patikrinti, kad kiekviena atskira dalis veikia teisingai, prieš integruojant jas į komponentų visumą.
- Sisteminiai testai – šie testai skirti patikrinti, ar visa integruota žaidimo sistema atitinka iškeltus reikalavimus. Jų metu pagrindinis dėmesys skiriamas patikrinti, ar visi atskiri komponentai funkcionuoja kaip vienas, stabilią ir malonią patirtį žaidėjams teikiantis žaidimas.
- Našumo testai – šių testų metu bus matuojami žaidimo reagavimo laikai ir stabilumas esant įvairioms apkrovoms. Į šį testavimą taip pat įtraukiami DI „stress“ testavimas pasitelkiant kompleksiškus žaidimo scenarijus, kad patikrinti ar šis veikia be strigimų ir trikdžų.

### 6.1.3. Pagrindiniai apribojimai

Nors testavimo plane siekiama apimti visus kritinius projekto aspektus, kai kuris žaidimo funkcionalumas yra neįtraukiamas dėl žemesnio prioriteto ar tam tikrų išorinių ribojimų. Neįtrauktas funkcionalumas:

1. Ne kritiniai UI elementai – žaidimo tematiką atspindintys ir kiti vizualiniai elementai, nesantys svarbūs pagrindinių žaidimų funkcijų veikimui.
2. Išplėstinis dirbtinio intelekto prisitaikymas, nepapuolantis į pradinę apimtį – nors pagrindinis dirbtinio intelekto algoritmų veikimas ir pritaikomumas bus ištestuotas pilnai, labai sudėtingo prisitaikymo elgesio testavimas nebus prioritetizuojamas.
3. Veikimas kitose platformose – testavimo planas susitelkia ties sistemomis, veikiančiomis Windows 10 ir aukštesnėse. Veikimo kitose, Unity žaidimų variklį palaikančiose sistemose, testavimas nepapuola į šiame testavimo plane aprašytą apimtį, tačiau gali būti įtrauktas į tolimesnes fazes, esančias už šio projekto ribų.

## 6.2. Testavimo planavimas

### 6.2.1. Testuojama programų sistema

Testuojama programinė įranga – strateginis žaidimas, naudojantis dirbtinio intelekto algoritmus ir metodikas bei grafinę vartotojo sąsają (UI). Šis testavimo planas padengia pagrindines žaidimo mechanikas, dirbtinio intelekto elgseną ir būtiną UI funkcionalumą. Išimtys – pagrindinėms žaidimo mechanikoms nebūtinas ir jose nenaudojamas UI funkcionalumas ir komponentai, tokie kaip vizualiniai elementai ir temos.

### 6.2.2. Testuojama vartotojo sąsaja

Į šią kategoriją įtraukiama pagrindiai meniu, interaktyvūs grafiniai žaidimo valdikliai, bei kritinę žaidimui informaciją perteikiantys skaitikliai.

### 6.2.3. Testavimo strategija

Šio projekto testavimo strategija susideda iš keletos testavimo lygių, iš kurių kiekvienas turi savo atskirus tikslus, ir kurių bendra paskirtis yra užtikrinti, kad kiekvienas žaidimo aspektas yra nuodugniai patikrintas. Testavimo lygiai:

- Funkcinis testavimas
- Vienetų testavimas
- Sisteminis testavimas
- Našumo testavimas

### 6.2.4. Funkcinis testavimas

Šio testavimo metu susitelkiama į užtikrinimą, kad kiekvienas žaidimo funkcionalumas veikia pagal reikalavimus. Funkcinis testavimas bus naudojamas tiek vienetų, tiek integracinių, tiek sisteminių testavimų metu.

### 6.2.5. Vienetų testavimas

Izoliuojamos specifinės funkcijos ir metodai, papildomas dėmesys sutelkiamas į su žaidimo ar DI logika susijusias funkcijas ir metodus. Komponentai testuojami atskirai vienas nuo kito. Pasitelkiami automatiniai testavimo įrankiai – Xunit ir Pytest, vienetų testų proceso greitumui ir rezultatų vienodumui užtikrinti.

### 6.2.6. Sisteminis testavimas

Testuojamas visas žaidimas, siekiama patikrinti ar visi komponentai dera ir veikia vienoje visumoje, ir ar žaidimas atitinka keliamus reikalavimus. Testavimo metu žiūrimas funkcionalumas, stabilumas ir tikslios žaidimo reakcijos į žaidėjo veiksmus.

### 6.2.7. Našumo testavimas

Siekiama įvertinti žaidimo kaip sistemos greitį, stabilumą ir reagavimo laiką, ypač didelės apkrovos metu. Šio testavimo tikslas rasti galimas kliūtis ir delsos problemas, šitaip užtikrinant, kad žaidimai po pataisymų veiks gerai, be trikdžių ir strigimų.

## 6.3. Testavimo ištekliai

Testavimui bus reikalingi šie ištekliai:

- Programinės įrangos ištekliai – Unity žaidimų variklis, Visual Studio 2022 Community, Github, git versijos kontrolės sistema.
- Aparatūrinės įrangos ištekliai – didelion našumo kompiuteris, turintis 32GB RAM, 1TB vietos turintį SSD, Ryzen 7 5600X procesorių ir Nvidia RTX 3080 Ti vaizdo plokštę.

## 6.4. Testavimo rezultatai

Testavimo rezultatai bus kaupiami toje pačioje git repozitorijoje kaip ir žaidimo kodas, ir su projektu susiję dokumentai.

## 6.5. Testavimo įrankiai ir aplinka

Automatizuoto testavimo įrankis naudojamas vienetų ir našumo testų metu - Unity testų karkasas. Testavimas vyks Windows 10 operacinės sistemos aplinkoje.

## 6.6. Testavimo tvarkaraštis

Testavimo tvarkaraštis:

- Funkcinis ir vienetų testavimas – 1-8 projekto mėnuo, taip pat pagal poreikį.
- Sisteminis testavimas – 6 - 8 projekto mėnuo.
- Našumo testavimas – 8 projekto mėnuo.

## 6.7. Testavimo procedūra (vykdymas)

### 6.7.1. Testuojama programų sistema

Testuojama programinė įranga – strateginis žaidimas, naudojantis dirbtinio intelekto algoritmus ir metodikas bei grafinę vartotojo sąsają (UI). Šis testavimo planas padengia pagrindines žaidimo mechanikas, dirbtinio intelekto elgseną ir būtiną UI funkcionalumą. Išimtys – pagrindinėms žaidimo mechanikoms nebūtinas ir jose nenaudojamas UI funkcionalumas ir komponentai, tokie kaip vizualiniai elementai ir temos.

### 6.7.2. Testavimo procedūros

Šiame skyriuje aprašomos tikslesnės procedūros kiekvienam testavimo tipui, nurodytam 2.2 skyriuje. Kiekvienoje sekcijoje nurodomi testavimo tikslai, specifiniai veiksmai, laukti rezultatai ir trumpi aprašymai.

### 6.7.3. Funkcinio testavimo vykdymas

1 lentelė. Funkcinio testavimo vykdymas.

| Testo nr. | Funkcija | Tikslas | Testuojantis veiksmas | Laukiamas rezultatas |
|---|---|---|---|---|
| 1 | Judėjimo sistema | Patikrinti veikėjų judėjimo funkcionalumą | Veikėjams duodamas nurodymas judėti | Veikėjai juda tiksliai pagal valdikliu duodamus nurodymus |
| 2 | Resursų valdymo sistema | Patikrinti resursų sistemos funkcionalumą | 1. Pastatomi resursus gaminantys pastatai.<br>2. Statyti pastatus.<br>3. Treniruoti naujus veikėjus. | Resursų kiekis kinta atitinkamai, pagal padarytus veiksmus. |
| 3 | Kovos sistema | Patikrinti kovos sistemos funkcionalumą | Veikėjams duodamas veiksmas pulti priešininkų valdomus veikėjus | Veikėjas puola priešininko valdomus veikėjus |
| 4 | Interakcija su dirbtinio intelekto algortimais | Patikrinti ar dirbtinio intelekto algoritmai veikia korektiškai | Žaidžiamas žaidimas | Dirbtinio intelekto algoritmai veikia pagal nustatytas normas |
| 5 | Dirbtinio intelekto kelio radimo algoritmas | Patikrinti dirbtinio intelekto valdomą kelio radimą | Žaidžiamas žaidimas | DI valdomų veikėjai teisingai randa kelią į reikiamas vietas |

### 6.7.4. Vienetų testavimo vykdymas

2 lentelė. Vienetų testavimo vykdymas.

| Testo nr. | Komponentas | Tikslas | Testuojantis veiksmas | Laukiamas rezultatas |
|---|---|---|---|---|
| 1 | Resursų valdymo sistemos skaičiavimai | Patikrinti resursų sistemos skaičiavimus | 1. Simuliuojamas veikėjams duodamas nurodymas rinkti resursus.<br>2. Simuliuojamas pastatų statymas.<br>3. Simuliuojamas naujų veikėjų treniravimas | Resursų kiekis kinta atitinkamai, pagal padarytus veiksmus. |

### 6.7.5. Sisteminio testavimo vykdymas

3 lentelė. Sisteminio testavimo vykdymas.

| Testo nr. | Funkcija | Tikslas | Testuojantis veiksmas | Laukiamas rezultatas |
|---|---|---|---|---|
| 1 | Žaidimo sesija | Patikrinti žaidimo stabilumą | Sužaidžiama pilna žaidimo sesija | Žaidimas veikia be klaidų. |

### 6.7.6. Našumo testavimo vykdymas

4 lentelė. Našumo testavimo vykdymas.

| Testo nr. | Aspektas | Tikslas | Testuojantis veiksmas | Laukiamas rezultatas |
|---|---|---|---|---|
| 1 | Atsako laikas | Užtikrinti greitą žaidimo atsako laiką | Matuojama įvairių žaidimo veiksmų atlikimo trukmė | Žaidimo atsako laikas atitinka reikalavimus (1 sekundė) |
| 2 | Didelė apkrova | Patikrinti žaidimo stabilumą esant didelei apkrovai | Simuliuojamas didelis kiekis žaidėjo ir DI atliekamų veiksmų su daug veikėjų | Žaidimas išlieka stabilus ir reaguojantis |
| 3 | Naudojamų sisteminių resursų kiekis | Išmatuoti CPU ir GPU apkrovą, bei sunaudojamą RAM kiekį žaidimo metu | Matuojami sisteminiai resursai ilgos sesijos metu | Sunaudojamų resursų kiekis išlyka numatytose ribose |

## 6.8. Testavimo išteklių paskirstymas

5 lentelė. Ištekliai, kurių reikės kiekvienam testavimo etapui.

| Testavimo tipas | Žmogiškieji ištekliai | Programinė įranga | Testavimo aplinka | Apskaičiuota trukmė |
|---|---|---|---|---|
| Funkcinis testavimas | 1 programuotojas | Unity editor, rankinio testavimo įrankiai | Programuotojo kompiuteris | 1 savaitė, nuolatinis |
| Vienetų testavimas | 1 programuotojas | XUnit, Visual Studio 2022 Community, Unity editor | Programuotojo kompiuteris | 1 savaitė, nuolatinis |
| Sisteminis testavimas | 1 programuotojas | Unity editor ir Unity žaidimų variklis | Programuotojo kompiuteris | 2-3 dienos |
| Našumo testavimas | 1 programuotojas | Unity editor ir Unity žaidimų variklis | Programuotojo kompiuteris | 1-2 dienos |

## 6.9. Testavimo rezultatų kaupimas

6 lentelė. Testavimo rezultatų kaupimas.

| Testavimo tipas | Renkami duomenys | Saugojimo vieta | Saugojimo periodas |
|---|---|---|---|

| | | | |
|---|---|---|---|
| Funkcinis testavimas | Testuojamų sąlygų rezultatai, ekrano iškarpos | Programuotojo kompiuteris, Git repozitorija | Projekto laikotarpis + 1 mėnesias |
| Vienetų testavimas | Testavimo logai, kodo padengimo ataskaitos | Programuotojo kompiuteris, Git repozitorija | Projekto laikotarpis |
| Sisteminis testavimas | Sisteminiai logai, našumo metrikos | Programuotojo kompiuteris, Git repozitorija | Projekto laikotarpis + 6 mėnesiai |
| Našumo testavimas | Našumo metrikos bei testinių duomenų rinkiniai | Programuotojo kompiuteris, Git repozitorija | Projekto laikotarpis + 1 metai |

# 7. TESTAVIMO REZULTATAI IR IŠVADOS

## 7.1. Testavimo rezultatai

Testavimas buvo vykdomas pagal 6 skyriuje nustatytas normas, vykdant funkcinius, vienetų, našumo ir sisteminius testavimus. Žemiau pateikiami kiekvieno testavimo tipo apibendrinti rezultatai.

### 7.1.1. Funkcinis testavimas

- Visi pagrindiniai žaidimo funkcionalumai, įskaitant resursų valdymas, dirbtinio intelekto veikimas ir UI veikimas veikia nustatytose normose.
- Dirbtinio intelekto algoritmai veikė sklandžiai, tačiau ribiniais atvejais kartais turėjo netikslumų.

### 7.1.2. Vienetų testavimas

- Padengtas resursų valdymo ir pažymėjimo valdymo funkcionalumas kiek įmanoma daugiau vienetų testų, testuojant tiek realius, tiek ribinius atvejus. Vienetų testai veikė sėkmingai.

### 7.1.3. Sisteminis testavimas

- Žaidimas atitiko iškeltus reikalavimus, veikė dauguma atvejų daug geriau, nei 60 kadrų per sekunde 1080p rezoliucijoje be jokių strigimų, trikdžių ar klaidų.
- Esant dideliai apkrovai, žaidimas be didesnių sunkumų susitvarkė su situacijomis.

### 7.1.4. Našumo testavimas

- Beveik viso testavimo metu, buvo užtikrintas optimalus pasiskirstymas tarp CPU ir GPU naudojamų resursų.
- Esant dideliai apkrovai, CPU sunaudojamų resursų kiekis augo šiek tiek daugiau nei GPU sunaudojamų resursų kiekis, tačiau šie rodikliai vistiek išliko numatytose normose.

### 7.2. Pastebėjimai, rasti testavimo metu

- Visų tipų testų testavimo metu, strateginio žaidimo sistema išliko stabili, bei nesukėlė didesnių sunkumų ar iššūkių.
- Dirbtinio intelekto kelio radimo logika buvo vienintelė vieta, kurioje buvo užfiksuoti didesni strigimai, ypač kai vienu metu buvo ieškomas kelias daugeliui padalinių.

### 7.3. Išvados

Testavimo rezultatai parodė, kad strateginio žaidimo sistema veikia tikslingai, sklandžiai, našiai ir užtikrina optimalų kompiuterio resursų išnaudojimą realių scenarijų metu. Testavimas taip pat atskleidė, jog dirbtinio intelekto kelio radimo algoritmą dar yra kur tobulinti.

## 8. NAUDOTOJO DOKUMENTACIJA

### 8.1. Sistemos funkcinis aprašymas

Strateginis žaidimas, kuriame žaidėjai stato savo miestą, jį plečia ir gina, tuo tarpu valdydami savo resursus, gamindami kareivius ir gindamiesi nuo priešų. Žaidimo tikslas yra sunaikinti visus priešus, neleidžiant jiems sunaikinti miesto centro pastato.

Pagrindinės žaidimo funkcijos:

- Pastatų statymas ir valdymas turint 12 skirtingų pastatų tipų.
- Resursų generavimas ir valdymas.
- 2 tipų kareivių gaminimas.
- Ginyba naudojantis 3 skirtingų tipų bokštus, sienas ir vartus.
- Puolimas naudojantis kareiviais.
- Laimėjimas sunaikinant visus priešus.
- Pralaimėjimas priešui sunaikinus miesto centrą.

### 8.2. Vartotojo atmintinė

Šiame skyriuje pateikiami baziniai vartotojo veiksmai, norint sklandžiai susikonfiguruoti ir žaisti žaidimą. Detalesnis sistemos aprašymas pateiktas 8.3 skyriuje.

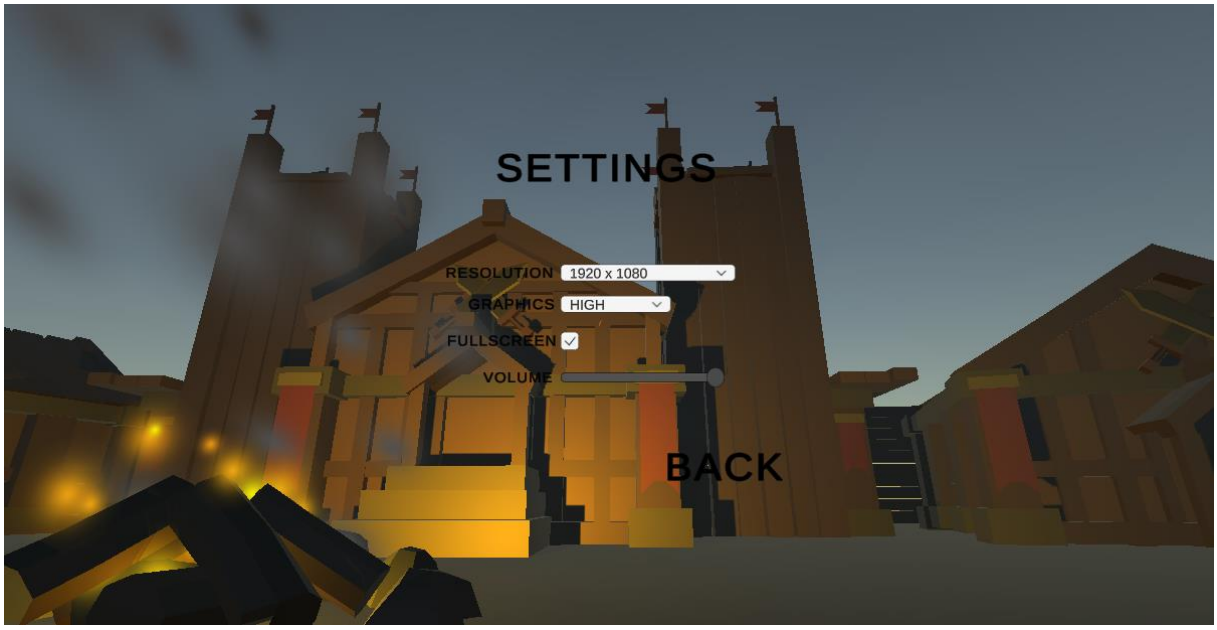
#### 8.2.1. Žaidimo pradėjimas

Įjungus žaidimą, žaidėjui matomas pagrindinio meniu langas, kuriame matomi trys mygtukai – Pradėti žaidimą (*Play*), nustatymai (*Settings*) ir išėjimo iš žaidimo mygtukas (*Quit*). Paspaudus *Play* mygtuką, pradedamas žaisti žaidimas.


**1 pav. Pagrindinio žaidimo meniu vaizdas.**

### 8.2.2. Žaidimo nustatymų keitimas

Žaidimo pagrindiniame meniu pasirinkus *Settings* mygtuką, žaidėjas perkeliamas į nustatymų koreagavimo langą.



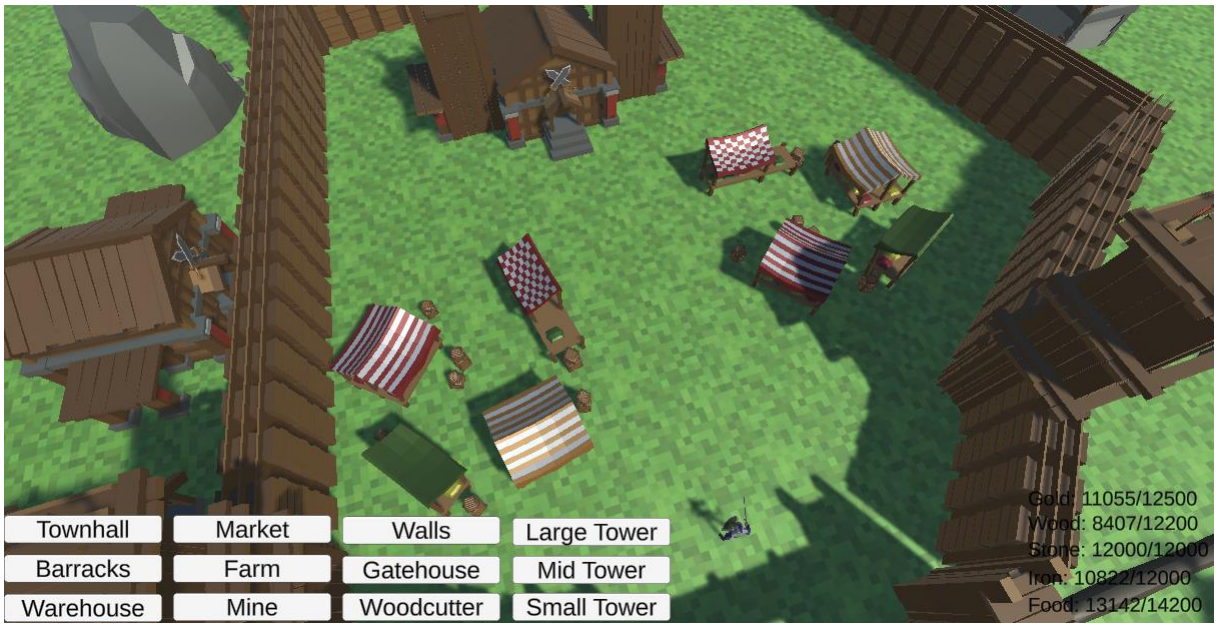

**2 pav. Žaidimo nustatymų langas.**

Nustatymų lange galima pasirinkti norimą ekrano rezoliuciją, grafikos nustatymus, pilno ekrano rėžimą bei koreguoti žaidimo garsumą naudojantis grafiniais elementais.

### 8.2.3. Kameros valdymas

Pradėjus žaisti žaidimą, galima valdyti kamerą (vaizdą). Kameros judinimas į bet kurią iš 4 krypčių vyksta WASD arba rodyklių klavišais klaviatūroje. Kameros priartinimas arba

atitolinimas vyksta pelės ratuko pagalba. Norint kamerą pasukti į šoną, laikomas „R“ klaviatūros klavišas kartu su dešiniu pelės mygtuku ir judinama pelė kad pasukti kamerą į norimą kryptį.




**3 pav. Kamera pasukta į apačią kad geriau matytųsi pastatyto miesto vaizdas.**

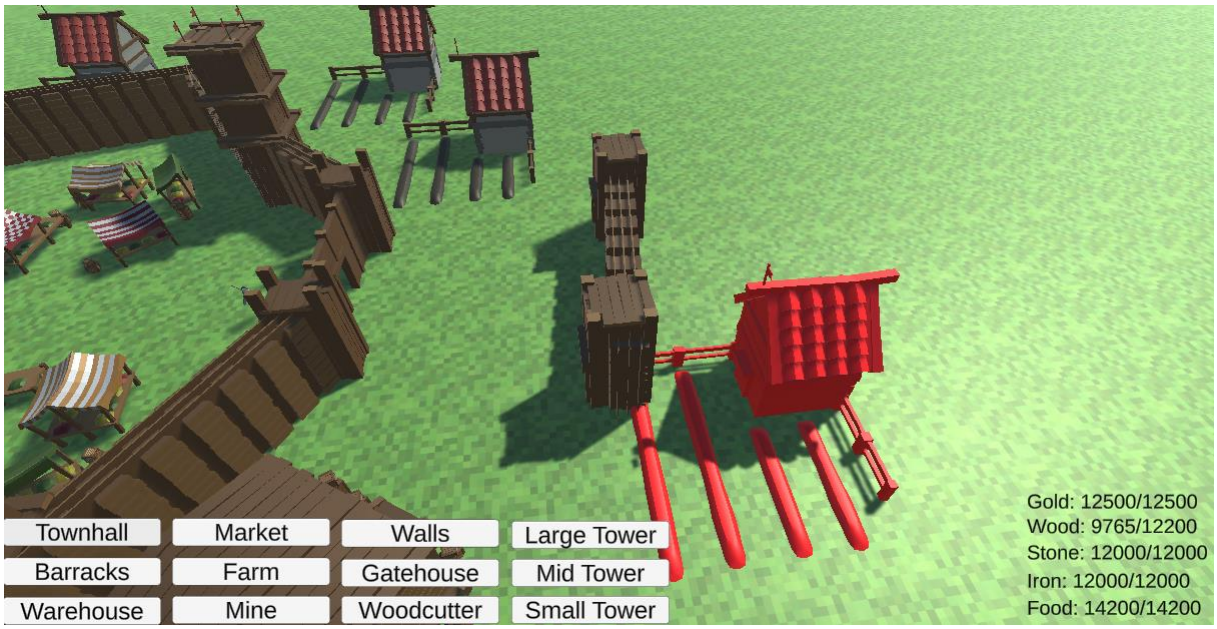
### 8.2.4. Pastatų statymas

Žaidimo metu, pastatų statymas vyksta kairiu pelės mygtuku pasirinkus norimo pastato mygtuką. Prieš paspaudžiant bet kurio pastato mygtuką, užvedus pelę ant bet kurio mygtuko, matoma pastato resursų kaina.

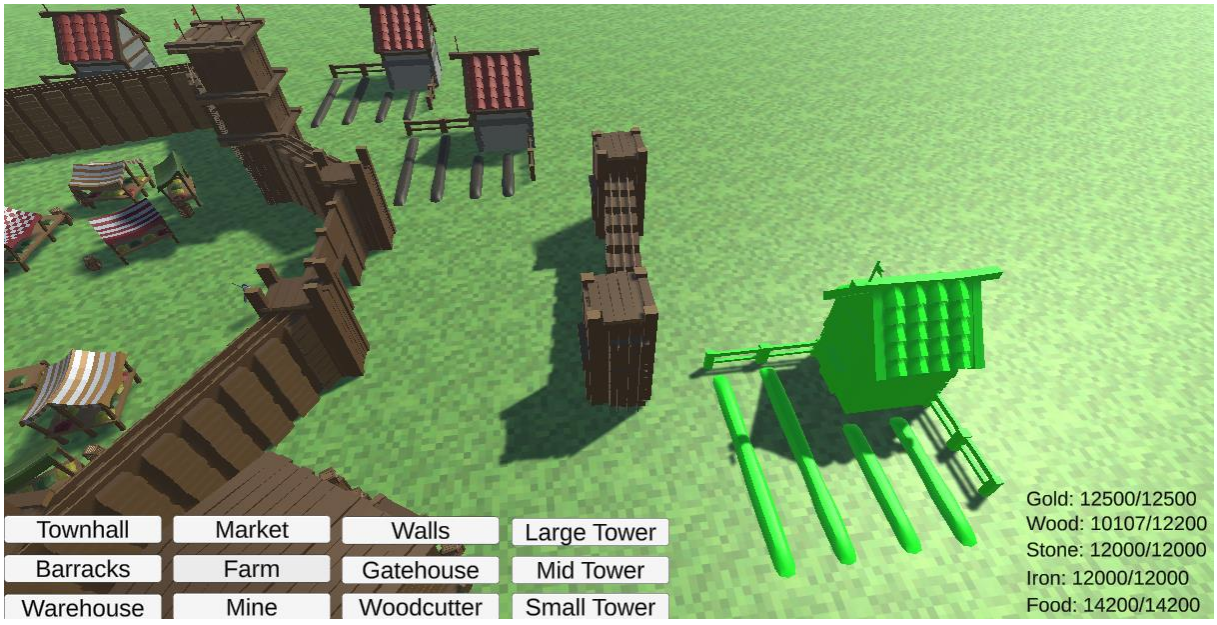

**4 pav. Matoma pastato kaina, užvedus pelę ant mygtuko *Market*.**

Paspaudus bet kurį iš pastatų mygtukų, pradedamas pastato statymas. Pastatas atsiranda pelės vietoje, tačiau iš pradžių yra nepastatytas. Norint pastatyti pastatą, reikia įsitikinti ar norimoje

vietoje nėra jokio kito pastato ar kareivio. Jeigu pastato vieta negalima, pastatas rodomas raudona spalva, o jeigu galima – pastatas rodomas žalia spalva.




**5 pav. Negalimas statyti pastatas, kuris kertasi su kitu, jau pastatytu pastatu.**




**6 pav. Galimas statyti pastatas.**

Suradus tinkamą vietą ir norint pastatyti pasirinktą pastatą, spaudžiamas kairys pelės klavišas ir pastatas yra pastatomas.

### 8.2.5. Resursai

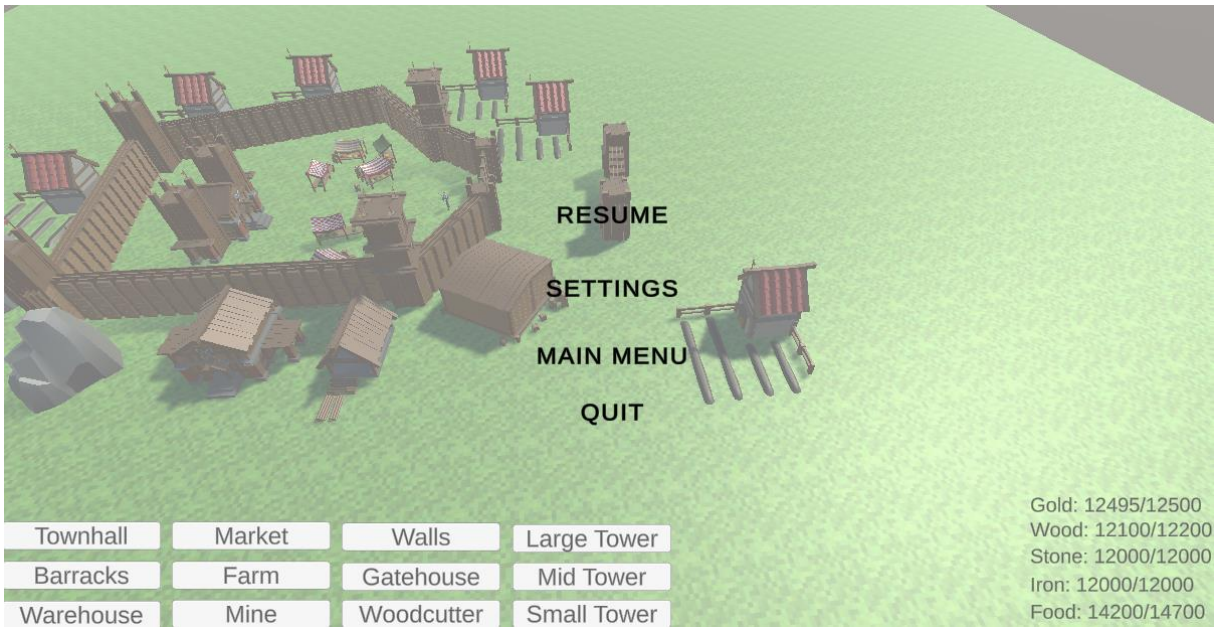
Resursų kiekiai ir talpos matomi apatiniame dešiniame ekrano kampe. Resursus periodiškai gaminantys pastatai – *Market* (gamina mažą kiekį visų resursų), *Farm* (gamina maisto resursą), *Mine* (gamina akmens ir geležies resursą), *Woodcutter* (gamina medžio resursą). Resursų talpą

praplečiantys pastatai – *Warehouse* (praplečia visų resursų talpą), *Farm* (praplečia maisto resurso talpą).


**7 pav. Resursus gaminantys ir resursų talpą praplečiantys pastatai. Resursų rodikliai.**

### 8.2.6. Meniu atidarymas žaidimo metu

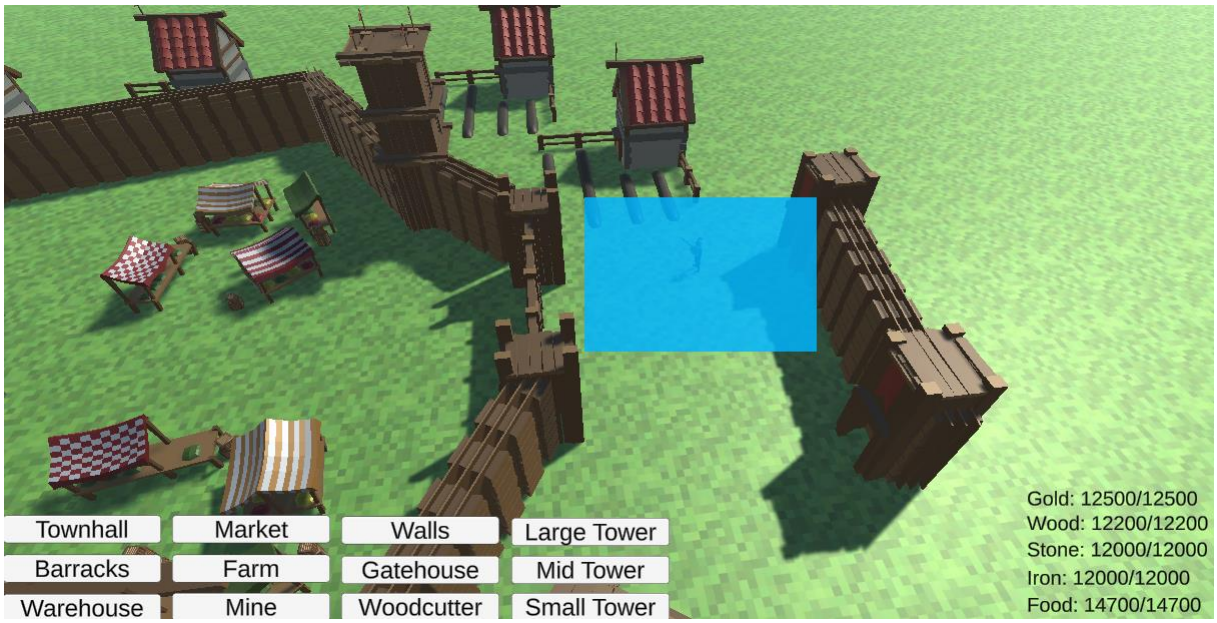
Žaidimo metu, žadėjas gali atidaryti meniu spausdamas „Esc“ arbar *Escape* mygtuką. Meniu navigacija analogiška kaip ir pagrindinio meniu (aprašyta 8.2.1. ir 8.2.2. poskyriuose). Paspaudus žaidimo meniu mygtuką, žaidime uždedama pauzė. Norint grįžti į žaidimą, spaudžiamas *Resume* mygtukas. Norint grįžti į pagrindinį meniu, spaudžiamas *Main Menu* mygtukas.




**8 pav. Žaidimo metu įjungtas meniu ir uždėta pauzė.**

### 8.2.7. Kareivių pažymėjimas ir valdymas

Kareivio pažymėjimui yra du būdai: paspausti kairį pelės klavišą tiesiai ant kareivio, arba laikyti paspaudus kairį pelės klavišą, ir tempti kol brėžiamas plotas apima visus norimus pasirinkti kareivius. Pasirinkus kareivį/-ius, jiems galima duoti nurodymus eiti į norimą vietą arba atakuoti norimą priešininką/pastatą dešiniu pelės klavišu paspaudus ant norimos vietos arba priešininko. Norint nustoti pažymėti kareivį, spaudžiamas kairys pelės klavišas.




**9 pav. Kareivių pasirinkimas laikant įspaudus kairį pelės klavišą.**

### 8.2.8. Kareivių gaminimas

Kareivius galima gaminti pastačius *Barracks* tipo pastatą. Pastačius šį pastatą, spaudžiamas kairys pelės klavišas ant šio pastato, ir atsiradusiame meniu lange paspaudžiamas norimas gaminti kareivis. Užvedus pelę ant norimo gaminti kareivio mygtuko, matomi kareivio gaminimo kaštai. Paspaudus kareivio gaminimo mygtuką, kareivis atsiranda šalia pasirinkto *Barracks* tipo pastato.


**10 pav. Pasirinktas *Barracks* tipo pastatas ir atsiradęs kareivių gaminimo pasirinkimas.**

## 8.3. Detalioji sistemos atmintinė

Pastatų tipai:

- *Townhall* – pagrindinis pastatas, kurį praradus, žaidimas pralaimimas
- *Barracks* – pastatas skirtas kareivių gaminimui.
- *Warehouse* – pastatas, skirtas išplėsti resursų limitus.
- *Market* – pastatas, periodiškai gaminantis visus resursus. Vienintelis pastatas aukso resurso gaminimui.
- *Farm* – pastatas, periodiškai gaminantis maisto resursą ir praplėčiantis šio resurso limitą.
- *Mine* – pastatas, periodiškai gaminantis akmens ir geležies resursus.
- *Walls* – apsauginis pastatas, skirtas aptverti miestą. Veikia kaip ginybinis pastatas, neleidžiantis priešams lengvai praeiti.
- *Gatehouse* – ginybinis pastatas, kurį galima statyti ant sienų.
- *Woodcutter* – pastatas, periodiškai gaminantis medžio resursą.
- *Large Tower* – pastatas, automatiškai šaudantis į artimiausius priešus. Turi didelį šaudymo nuotolį ir darantis didelę žalą, tačiau šaudantis lėtai.
- *Mid Tower* – pastatas, automatiškai šaudantis į artimiausius priešus. Turi vidutinį šaudymo nuotolį ir darantis vidutinę žalą, tačiau šaudantis vidutiniškai greitai.
- *Small Tower*– pastatas, automatiškai šaudantis į artimiausius priešus. Turi mažą šaudymo nuotolį ir darantis mažiausią žalą, tačiau šaudantis labai greitai.

Resursų tipai:

- *Gold* – auksas, skirtas pastatų statymui.
- *Wood* medis, skirtas daugelio pastatų statymui.
- *Stone* – akmuo, skirtas pastatų statymui.
- *Iron* – geležis, skirta kareivių gaminimui ir pastatų statymui.

– *Food* – maistas, skirtas kareivių gaminimui.

*Walls* tipo pastatas gali būti statomas kiaurai visų tipų bokštus ir vartus. Visų tipų bokštai ir vartai gali būti statomi tiesiai ant sienų.

### 8.4. Sistemos diegimas

Sistemos diegimas susideda iš šių etapų:

– Užtikrinama, kad kompiuteris, kuriame bandomas diegti žaidimas, turi Windows 10 arba naujesnę operacinę sistemą ir reikiamus CPU/GPU ir RAM kiekius.
– Vykdomas DayOfWrath.exe failas.

## 9. PAKEITIMŲ SĄRAŠAS

– Su užsakovu sutartas projekto pasukimas nuo vien tik priešininko elgsenos modeliavimo naudojantis dirbtinio intelekto algoritmais/metodais/modeliais, į bendrą dirbtinio intelekto panaudojimą strateginiuose kompiuteriniuose žaidimuose. Dėl šio punkto atlikti minimalūs pakeitimai daugumoje teksto vietų, įvykdyti perfrazavimai.
– Nenaudojama Python programavimo kalba kartu su TensorFlow ir Keras bibliotekomis, o liekama prie tik .NET C# programavimo kalba parašyto kodo. Šis sprendimas buvo priimtas atsižvelgiant į laiko trūkumą iki projekto įgyvendinimo dienos, sprendimo kompleksiškumą ir per mažos naudos gavimą palyginus su reikalingomis įdėti pastangomis. Šis pakeitimas galioja visuose dokumente esančiuose skyriuose, ir dėl šios priežasties buvo perbraižytos kai kurios dokumente esančios diagramos (4.6.1 ir 5.7)
– Perbraižytos beveik visos diagramos (išdėstymo, klasių), esančios 5 skyriuje, kadangi jos buvo braižytos prieš 2 metus ir buvo pasenę ir nebeatitiko užsakovo poreikių.
– Pasikeitė 4.3 vartotojų prioritetai, kadangi noras orientuotis yra labiau į kitus žaidimų kūrėjus, kurie galės naudoti įgyvendintus algoritmus, nei tiesiog į paprastus žaidėjus, kurie galės tik žaisti žaidimą, negaudami antrinės naudos.
– Testavimo plano dalyje palikti tik aktualūs testavimo tipai, išimti testavimo tipai, kurių vykdymo laiko ir naudos santykis yra nepriimtinas nei užsakovui, nei vykdytojui.
– Padaryta, kad kiekvieno dokumento skyriaus paveiksliukų ir lentelių pavadinimų numeracija prasidėtų nuo 1.
– Pataisyti dokumentų formatavimai.
– Ištrinti nereikalingi tarpai tarp teksto ir skyrių pavadinimų.
– Gramatinių ir loginių teksto klaidų pataisymai.

## 10. IŠVADOS

Strateginių žaidimų žaidėjams ir strateginių žaidimų kūrėjų komandoms ir įmonėms, kuriems įdomus ar komerciškai naudingas dirbtinio intelekto metodų naudojimas strateginiuose žaidimuose. Kurie nori linksmai praleisti laisvalaikį arba sukurti kompiuterinį strateginį žaidimą, kuris naudotų dirbtinio intelekto metodus, kurių implementacijos nėra viešai prieinamos, dėl žaidimo kūrimo kompanijų ir korporacijų, kurios nepadaro savo kodo atviro.

Mūsų siūlomas sprendimas: laisvai prieinamas atviro kodo strateginis žaidimas ir jame panaudotų dirbtinio intelekto metodų analizės rezultatas su alternatyvomis.

Suteikiamas įtraukiantis strateginis žaidimas, kuris žaidėjui leis jį peržaisti bent kelis kartus, dėl jame įgyvendinto dirbtinio intelekto. Taip pat suteikiama panaudotų dirbtinio intelekto metodų analizė, leisianti žaidimus kuriančioms komandoms ar įmonėms panaudoti šiuos sprendimus jų pačių kuriamuose žaidimuose.

Buvo atlikta įvairių literatūros šaltinių, susijusių su magistrinio darbo tema, analizė.

Išanalizavus dirbtinio intelekto naudojimą kompiuteriniuose žaidimuose pastebėtas ryški pažanga šioje srityje – per paskutinius 70 metų dirbtinis intelektas kompiuteriniuose žaidimuose patobulėjo nuo paprastų statinių metodų iki sudėtingų neuroninių tinklų ir tapo neatskiriama kompiuterinių žaidimų dalimi.

Pastebėta, kad dirbtinis intelektas kompiuteriniuose žaidimuose išskiriamas į dvi rūšis, o šios rūšys turi daug skirtingų porūšių, kurie tiek pavieniai, tiek kartu su kitais naudojami kompiuteriniuose žaidimuose.

Atliekant esamų sprendimų analizę, išsiaiškinta, jog tiek populiarūs, tiek nelabai populiarūs žaidimai naudoja dirbtinį intelektą vienokia ar kitokia forma ir padeda kompiuterio valdomiems žaidėjams imituoti tikrus žaidėjus ir tikriems žaidėjams užtikrinti įdomiai praleistą laiką.

Pastebėta tendencija, kad didžioji dauguma net ir nesenai išleistų kompiuterinių žaidimų vis dar naudoja kiek pasenusius, tačiau per laiką išdirbtus dirbtinio intelekto metodus, o tik maža dalis naujų kompiuterinių žaidimų bando įtraukti sudėtingesnius ir ne tiek išdirbtus dirbtinio intelekto metodus, dėl augančios žaidimo kainos ir testavimo laiko.

Testavimo rezultatai parodė, kad strateginio žaidimo sistema veikia tikslingai, sklandžiai, našiai ir užtikrina optimalų kompiuterio resursų išnaudojimą realių scenarijų metu. Testavimas taip pat atskleidė, jog dirbtinio intelekto kelio radimo algoritmą dar yra kur tobulinti.

## LITERATŪROS SĄRAŠAS

### Skyriaus Projekto paraiška literatūros sąrašas


[1] J. Dsouza, „*AI in Gaming*“ [Tinkle]. Nuoroda: https://www.engati.com/blog/ai-in-gaming [Kreiptasi 29 12 2022]
[2] C. Utting, „7 Video Games That Have Terrible, Terrible Artificial Intelligence“, 06 04 2014 [Tinkle]. Nuoroda: https://wegotthiscovered.com/gaming/7-video-games-terrible-terrible-a-i/ [Kreiptasi 29 12 2022]
[3] A. Anand, A. Kumar, „The Rise of Artificial Intelligence in Video Games“, IRJMETS, 07 2022 [Tinkle]. Nuoroda: https://www.irjmets.com/uploadedfiles/paper//issue_7_july_2022/27897/final/fin_irjmets1658458318.pdf [Kreiptasi 29 12 2022]
[4] J. Howarth, „How Many Gamers Are There?“, 07 10 2022 [Tinkle]. Nuoroda: https://explodingtopics.com/blog/number-of-gamers [Kreiptasi 29 12 2022]
[5] N. Gilbert „Number of Gamers Worldwide 2022/2023: Demographics, Statistics, and Predictions“, 06 11 2022 [Tinkle]. Nuoroda: https://financesonline.com/number-of-gamers-worldwide/ [Kreiptasi 29 12 2022]
[6] SteamCharts [Tinkle]. Nuoroda: https://steamcharts.com/ [Kreiptasi 29 12 2022].
[7] „Strategy Games - Worldwide“ [Tinkle]. Nuoroda: https://www.statista.com/outlook/dmo/app/games/strategy-games/worldwide [Kreiptasi 29 12 2022]
[8] S. Furqon „Deterministic AI pattern“ [Tinkle]. Nuoroda: https://www.researchgate.net/figure/Deterministic-AI-pattern_fig1_350476112 [Kreiptasi 29 12 2022]

[9] G. Seeman, D. M. Bourg „*Chapter 1. Introduction to Game AI*“, OREILLY [Tinkle]. Nuoroda: https://www.oreilly.com/library/view/ai-for-game/0596005555/ch01.html#:~:text=Game%20AI%20techniques%20generally%20come%20in%20two%20flavors%3A%20deterministic%20and%20nondeterministic.&text=Deterministic%20behavior%20or%20performance%20is,is%20a%20simple%20chasing%20algorithm. [Kreiptasi 29 12 2022]
[10] „Difference between Deterministic and Non-deterministic algorithms“, GeeksForGeeks 13 09 2022 [Tinkle]. Nuoroda: https://www.geeksforgeeks.org/difference-between-deterministic-and-non-deterministic-algorithms/ [Kreiptasi 29 12 2022]
[11] S. Pathak, „*Artificial Intelligence and Machine Learning in Game*“, Medium, 11 21 2021 [Tinkle]. Nuoroda: https://medium.com/geekculture/artificial-intelligence-and-machine-learning-in-game-33ae7310e30a [Kreiptasi 29 12 2022]
[12] Unity Documentation [Tinkle]. Nuoroda: https://docs.unity.cn/ru/2021.1/Manual/system-requirements.html [Kreiptasi 29 12 2022]
[13] S. Schaefsky, „Unity seems to break VS 2019 project files“, 06 05 2020 [Tinkle]. Nuoroda: https://forum.unity.com/threads/unity-seems-to-break-visual-studio-2019-project-files-can-not-attach-to-unity.884683/ [Kreiptasi 29 12 2022]

## Skyriaus projektavimo metodologijos ir technologijų analizė literatūros sąrašas

[1] J. Dsouza, „*AI in Gaming*“ [Tinkle]. Nuoroda: https://www.engati.com/blog/ai-in-gaming [kreiptasi 26 11 2022]
[2] „*History of AI Use in Video Game Design*“ [Tinkle]. Nuoroda: https://bigdataanalyticsnews.com/history-of-artificial-intelligence-in-video-games/ [kreiptasi 18 11 2022]
[3] A. Schulz, „*25 years ago: Deep Blue beats Kasparov*“ [Tinkle]. Nuoroda: https://en.chessbase.com/post/25-years-ago-deep-blue-beats-kasparov [kreiptasi 26 11 2022]
[4] S. Rizzo „*Imagined Thinking Machines: Artificial Intelligence in Video Games*“, Full Sail University, bal. 2, 2015 [Tinkle]. Nuoroda: https://hub.fullsail.edu/articles/imagined-thinking-machines-artificial-intelligence-in-video-games [Kreiptasi 26 11 2022]
[5] M. Trends, „*The Evolution of Artificial Intelligence in Video Games*“, Analytics Insight, 12 29 2020 [Tinkle]. Nuoroda: https://www.analyticsinsight.net/the-evolution-of-artificial-intelligence-in-video-games/ [Kreiptasi 26 11 2022]
[6] G. Seeman, D. M. Bourg „*Chapter 1. Introduction to Game AI*“, OREILLY [Tinkle]. Nuoroda: https://www.oreilly.com/library/view/ai-for-game/0596005555/ch01.html#:~:text=Game%20AI%20techniques%20generally%20come%20in%20two%20flavors%3A%20deterministic%20and%20nondeterministic.&text=Deterministic%20behavior%20or%20performance%20is,is%20a%20simple%20chasing%20algorithm. [Kreiptasi 26 11 2022]
[7] S. Pathak, „*Artificial Intelligence and Machine Learning in Game*“, Medium, 11 21 2021 [Tinkle]. Nuoroda: https://medium.com/geekculture/artificial-intelligence-and-machine-learning-in-game-33ae7310e30a [Kreiptasi 26 11 2022]
[8] N. Ahamed, „*AI Cheating in Games*“, Medium, 09 30 2022 [Tinkle]. Nuoroda: https://medium.com/mlearning-ai/ai-cheating-in-games-583e333677ce#:~:text=Cheating%20in%20the%20context%20of,have%20in%20the%20same%20circumstance. [Kreiptasi 26 11 2022]
[9] G. Valiulis, „Skysčio lygio valdymas taikant *fuzzy* logiką“, 09 02 2006 [Tinkle]. Nuoroda: http://www.elektronika.lt/straipsniai/elektronika/3168/skyscio-lygio-valdymas-taikant-fuzzy-logika/ [Kreiptasi 27 12 2022]
[10] K. Chaudhari, „*Implementing fuzzy logic to bring AI characters alive in Unity based 3D games*“, Packt hub, 01 06 2018 [Tinkle]. Nuoroda: https://hub.packtpub.com/fuzzy-logic-ai-characters-unity-3d-games/
[11] M. Pirovano, „*The use of Fuzzy Logic for Artificial Intelligence in Games*“, University of Milano, 12 07 2012 [Tinkle]. Nuoroda: http://www.michelepirovano.com/pdf/fuzzy_ai_in_games.pdf [Kreiptasi 27 12 2022]
[12] B. Antonette, „*Decision Tress in Video Games*“, Medium, 11 12 2019 [Tinkle]. Nuoroda: https://medium.com/@antoneb/decision-trees-in-video-games-3ea3f251f96e#:~:text=As%20mentioned%20earlier%20decision%20trees,not%20even%20be%20aware%20of. [Kreiptasi 28 12 2022]

[13] E. Eliaçık, „*AI in gaming: A complete guide*“, Data Conomy, 28 09 2022 [Tinkle]. Nuoroda: https://dataconomy.com/2022/04/artificial-intelligence-games/#Decision_trees [Kreiptasi 28 12 2022]
[14] R. Penton, „*Trees Part 1*“, Gamedev [Tinkle]. Nuoroda: http://archive.gamedev.net/archive/reference/programming/features/trees1/page4.html [Kreiptasi 28 12 2022]
[15] L. Cross „*What Neural Networks Playing Video Games Teach Us About Our Own Brains*“, CalTech, 07 01 2021 [Tinkle]. Nuoroda: https://www.caltech.edu/about/news/neural-networks-playing-video-games-teach-us-about-our-own-brains [Kreiptasi 28 12 2022]
[16] „*Convolutional Network for Game*“ [Tinkle]. Nuoroda: https://wiki.tum.de/display/lfdv/Convolutional+Neural+Network+for+Game [Kreiptasi 28 12 2022]
[17] „*The Greatest Game Of Civilization V No One Played*“ [Tinkle]. Nuoroda: https://civilization.com/civilization-5/news/entries/the-greatest-game-of-civilization-v-no-one-played/ [Kreiptasi 28 12 2022]
[18] Wikipedia – „*Civilization V*“ [Tinkle]. Nuoroda: https://en.wikipedia.org/wiki/Civilization_V [Kreiptasi 28 12 2022]
[19] Reddit – „*How exactly does Civ v's AI work?*“, 06 02 2016 [Tinkle]. Nuoroda: https://www.reddit.com/r/civ/comments/44g8id/how_exactly_does_civ_vs_ai_work/ [Kreiptasi 28 12 2022]
[20] Wikipedia – „*Blackboard system*“ [Tinkle]. Nuoroda: https://en.wikipedia.org/wiki/Blackboard_system#:~:text=A%20blackboard%20system%20is%20an,and%20ending%20with%20a%20solution. [Kreiptasi 28 12 2022]
[21] B. Poli, „*Creating Decoupled Features: The Blackboard System*“, Game Developer, 19 06 2018 [Tinkle]. Nuoroda: https://www.gamedeveloper.com/design/creating-decoupled-features-the-blackboard-system [Kreiptasi 28 12 2022]
[22] M. Rorie, „*Age of Empires III Walkthrough*“, GameSpot, 25 01 2006 [Tinkle]. Nuoroda: https://www.gamespot.com/articles/age-of-empires-iii-walkthrough/1100-6136278/ [Kreiptasi 28 12 2022]
[23] Wikipedia – „*Age of Empires III*“ [Tinkle]. Nuoroda: https://en.wikipedia.org/wiki/Age_of_Empires_III [Kreiptasi 28 12 2022]
[24] Age Of Empires Fandom Wikipedia - „*Artificial Inteligence*“ [Tinkle]. Nuoroda: https://ageofempires.fandom.com/wiki/Artificial_intelligence#Age_of_Empires_III [Kreiptasi 28 12 2022]
[25] J. Rasmussen, „*Are Behavior Trees a Thing of the Past?*“, Game Developer, 27 04 2016 [Tinkle]. Nuoroda: https://www.gamedeveloper.com/programming/are-behavior-trees-a-thing-of-the-past- [Kreiptasi 28 12 2022]