KAUNO TECHNOLOGIJOS UNIVERSITETAS

INFORMATIKOS FAKULTETAS

2D Quake šaudyklės žaidimas

PROGRAMAVIMO TECHNOLOGIJŲ PRAKTIKA

Kursinio darbo ataskaita

Atliko: IF-8/7 gr. studentai
Tomas Uktveris
Remigijus Valys

Priėmė: dėst. G. Palubeckis

Kaunas, 2010

TURINYS

I. Teorinė dalis

1. Užduoties analizė

1.1. *Pavadinimas*. "2D Quake šaudyklės žaidimas"

1.2. *Užduotis*

Sukurti dvimatės grafikos žaidimą-šaudyklę, kuriame vartotojas galėtų žaisti prieš kelis autonomiškai valdomus žaidėjus-botus arba stebėti ir analizuoti automatinį botų žaidimą. Botų logikos būsena bei judėjimas būtų vaizduojamas grafiškai programos analizės tikslais.

Aprašymas

Quake šaudyklės tipo žaidime svarbiausia kuo ilgiau išlikti gyvam, rinkti išgyvenimui būtinus daiktus(amuniciją, vaistinėles ir pan.) bei pelnyti kuo daugiau taškų naikinant kitus žaidėjus. Dažniausiai *kiekvienas sunaikintas priešas = vienas taškas.*

Žaidimą patogu vykdyti raundais, kiekvieną raundą baigiant, kai bet kuris iš žaidėjų surenka *max* taškų skaičių(*mūsų atveju 15 taškų*). Žaidėjai taip pat lyginami pagal mirčių skaičių, todėl žaidėjas žuvęs mažiau kartų už kitą bus aukštesnėje geriausių žaidėjų lentelės vietoje.

Žaidimuose įprasta pagal surinktų taškų kiekį atitinkamai skirti ir nugalėtojų vietas, todėl šiame žaidime skiriami aukso, sidabro ir bronzos apdovanojimai pirmiems trims geriausiems žaidėjams.

Autonomiškai valdomi žaidėjai-botai dažnas tokių žaidimų funkcionalumas, kai nėra galimybės žaisti prieš tikrus žaidėjus internete ar LAN tinkle. Jie taip pat stengiasi imituoti žmogaus žaidimą, kuo ilgiau išgyvenanti, numatyti kitų žaidėjų veiksmus ir pozicijas, pirmiau pasiimti geresnius daiktus nei konkurentai.

1.3. *Sistemos paskirtis ir tikslai*

Ši programinė įranga, toliau PĮ, kuriama mokymosi tikslais ir yra programavimo technologijų praktikos kursinis darbas. Šiuo darbu siekiama įtvirtinti įvairių modulių(diskrečiosios matematikos, programavimo ir kt.) sukauptas žinias ir panaudoti jas praktiškai. Su sukurta PĮ bus galima stebėti ir analizuoti autonominių aktorių priimamus sprendimus, matyti jų navigaciją žaidimo erdvėje, prireikus toliau tobulinti.

1.4. *Užsakovas.* PĮ užsakovas yra KTU Programų inžinerijos katedra.

1.5. *Vartotojai.* PĮ leidžiama naudotis laisvai.

2. Duomenų analizė ir reikalavimai

2.1. *Apribojimai sistemai*

2.1.1. PĮ turi veikti *Linux* operacinėje sistemoje, *Intel x86* architektūros kompiuteriuose.

2.1.2. PĮ kuriama *Java kalba*

2.1.3. Grafinė vartotojo sąsaja programuojama naudojant *SWING* grafines bibliotekas.

2.2. *Funkciniai reikalavimai*

PĮ yra keliami šie funkciniai reikalavimai:

2.2.1. Galimybę žaisti vienam prieš botus arba stebėti autonominį botų-žaidėjų žaidimą.

2.2.2. Galimybė grafiškai matyti botų veiksmus, logiką ir judėjimo trajektorijas.

2.2.3. Galimybė keisti žaidimo žemėlapius-zonas, pridėti daugiau žaidėjų-botų. Žaidimo žemėlapiai turi būti įvedami iš duomenų failų, kurie aprašo žaidimo zoną ir joje esančius objektus.

2.2.4. Automatinė žemėlapio analizavimo sistema. Ji turi išanalizuoti pateiktą žemėlapį-zoną atsižvelgiant į kliūtis ir sukurti navigacinį grafą botams-žaidėjams judėti.

2.2.5. Leisti vartotojui atlikti botų žaidimo simuliaciją pažingsniui.

2.2.6. Žaidimo valdymui reguliuoti panaudoti baigtinių automatų principą pagal *State design pattern* šabloną.

2.3. *Nefunkciniai reikalavimai*

PĮ yra keliami šie nefunkciniai reikalavimai:

2.3.1. PĮ turi būti realizuota grafinė vartotojo sąsaja.
2.3.2. Vartotojo sąsaja anglų kalba.
2.3.3. Vartotojo sąsaja turi būti paprasta, intuityvi ir funkcionali.
2.3.4. Programos reakcijos laikas turi būti priimtinas vartotojui.
2.3.5. PĮ turi veikti stabiliai ir netrikdyti trečiųjų šalių PĮ ir operacinės sistemos darbo.
2.3.6. PĮ neturi sumažinti kompiuterio saugumo tinkle.

2.4. *Papildomi reikalavimai*

PĮ kūrimo procesas turi vykti etapais:

2.4.1. Užduoties analizė ir sprendimo metodo tyrimas.
2.4.2. UML diagramų sudarymas.
2.4.3. Algoritmo realizacija.
2.4.4. Grafinės vartotojo sąsajos realizacija.
2.4.5. Algoritmo ir grafinės dalies apjungimas į sistemą.
2.4.6. Projekto dokumentacijos rengimas.

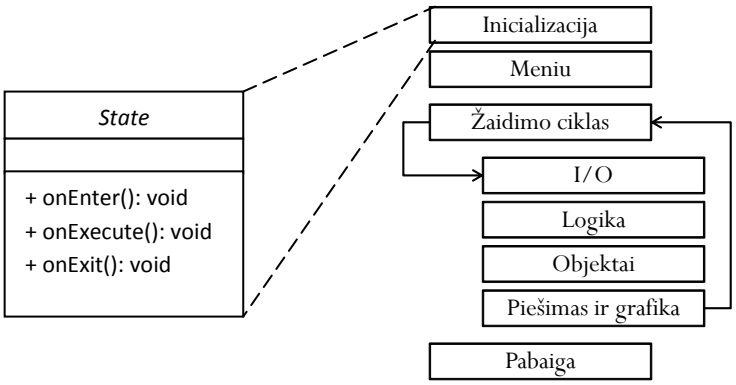
2.5. *Vartotojo dokumentacija.*

Sistemos kūrėjas turi pateikti tokią dokumentaciją:

2.5.1. Ataskaitą.
2.5.2. Vartotojo vadovą.
2.5.3. Diegimo instrukciją.
2.5.4. Diegimo kompaktinį diską.

3. Sprendimo metodas ir grafinė realizacija

Projekto įgyvendinimui pasirinkta programavimo kalba *Java* dėl savo universalumo ir daugiaterpiškumo. Žaidimą nutarta kurti su 2D *SWING* grafine biblioteka. Jo esminėms dalims – programos vykdymui bei botų veiksmų planavimui(*ataka, gynyba, pabėgimas ir pan*.) pasirinkta naudoti *State design pattern* būsenų šabloną. Tokiu būdu kodas tampa universalesnis, mažiau supainiotas ir neaiškus, lengvai keičiamas, nes kiekviena būsena tampa lyg atskirai funkcionuojančiu moduliu. Kiekvienai iš būsenų priskiriant tris pagrindinius metodus *onEnter(), onExecute(), onExit(),* kaip parodyta (1 pav.). Programos veikimo metu(*ir boto valdymo metu*) vykdoma tik viena iš būsenų.



1 pav. Pagrindinių žaidimo būsenų diagrama

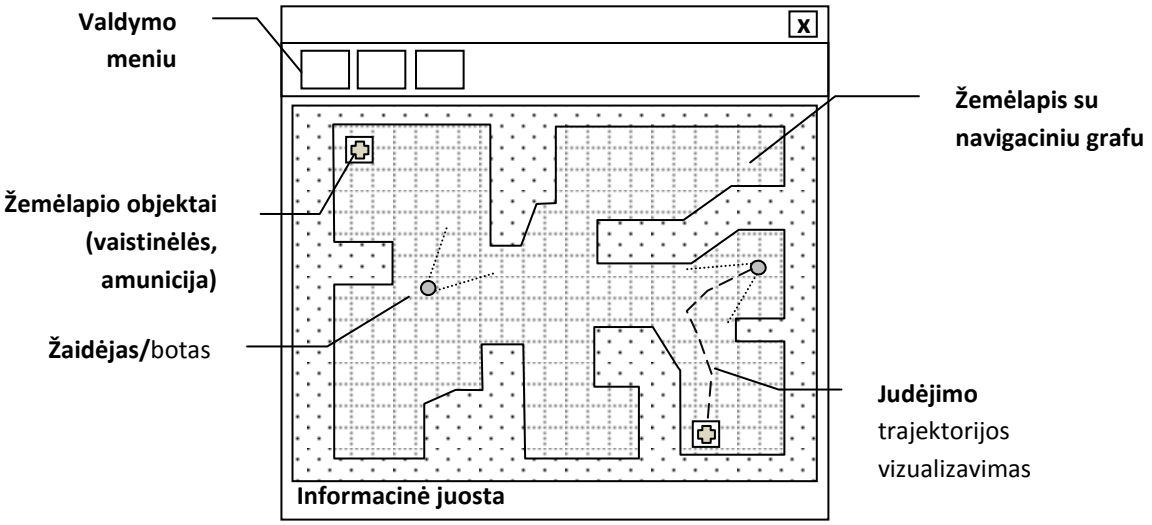
Žaidimas vizualiai įgyvendinamas pasitelkus 2D grafinius objektus iš SWING bibliotekos (2 pav.). Kiekvieną kadrą(~30 kadrų/sekundę) atnaujinant simuliaciją grafika piešiama lango ekrane panaudojant linijas, apskritimus ir kitus 2D primityvus. Siekiama realizuoti tokias pagrindines grafines sistemos funkcijas:

- Brėžti botų judėjimo trajektoriją ekrane per navigacinį grafo tinklą.
- Informuoti tekstu ekrane apie boto-žaidėjo esamą būseną bei siekiamą tikslą.
- Ekrano lange piešti žemėlapį ir jame esančius objektus.

II. Vartotojo sąsajos projektas

Siekiant programos valdymą padaryti kiek galima paprastesnį ir funkcionalesnį apsiribota pačiais svarbiausiais valdymo elementais – meniu juosta ir keli mygtukai pagrindinėms operacijoms atlikti: žemėlapio pasirinkimui, grafinės sąsajos ir informacijos pateikimo valdymui. Didžiausia vieta programos lange skiriama žaidimo ekranui, kuriame vyksta veiksmas. Taip pat žaidimo ekrane pateikiama daugiausiai informacijos – botų judėjimo trajektorijos, objektai, taškai ir kt.

1. ***Grafinės sąsajos prototipas***



2 pav. Grafinės programos sąsajos prototipas

Aprašymas

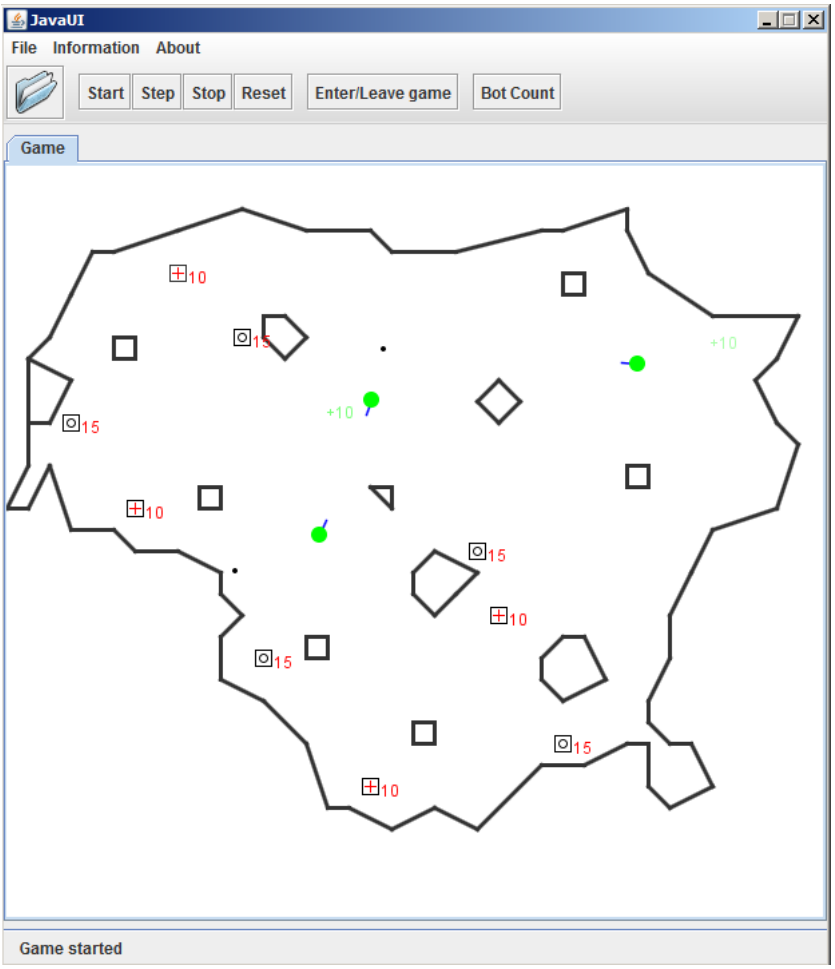
Žaidimo prototipe (2 pav.) buvo numatyti tokie grafinės vartotojo sąsajos objektai:

1. *Viršutinė meniu juosta su paspaudimu valdomais mygtukais*
   a. žemėlapio pasirinkimui
   b. žaidimo/programos užbaigimui
   c. žaidimo paleidimui/stabdymui, vykdymui pažingsniui
   d. žemėlapio ir skaičiavimų vizualizavimo nustatymui

2. *Centrinė žaidimo ekrano dalis*
   Ekrane piešiamas žemėlapis, kartu su navigaciniu grafu, žaidėjais/botais, žemėlapio objektais (vaistinėlėmis, amunicija). Vykstant žaidimui fiksuotu kadrų per sekundę skaičiumi perpiešiamas 2D žaidimo ekranas, taip atnaujinant žaidėjo/botų judesius bei animacijas.

3. *Apatinė lango dalis*
   Papildoma informacija apie atliktus veiksmus ir mygtukų paspaudimus yra rodoma informacinėje juostoje, esančioje ekrano apačioje.

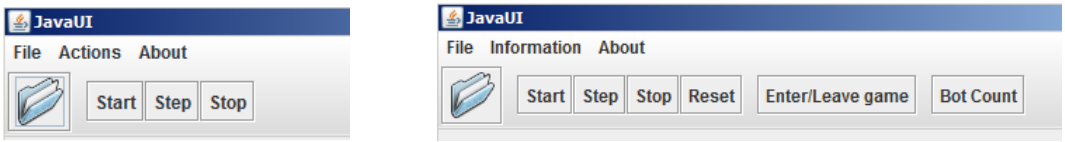
2. *Realizuota grafinė sąsaja*



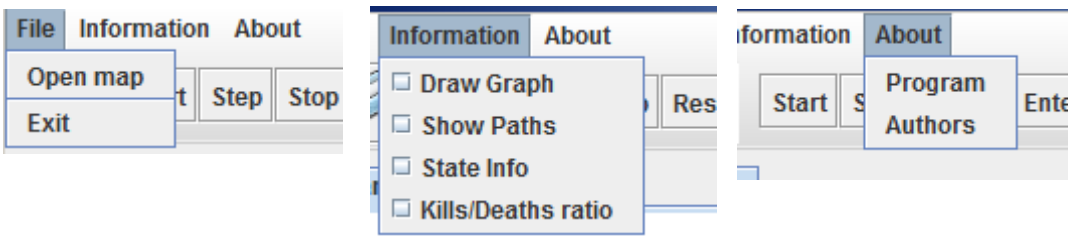
3 pav. Realizuotos grafinės sąsajos galutinis vaizdas

Galutinėje programoje grafinė sąsaja pilnai realizuota, atlikti pakeitimai aprašomi toliau esančiuose punktuose.

Atliktų pakeitimų aprašymas:



4a pav. Meniu išdėstymo pakeitimai



4b pav. Submeniu pakeitimai

1. *Viršutinės meniu juostos pakeitimai (4a-4b pav.)*

File→Open map: žemėlapio pasirinkimui.
File→Exit: žaidimo/programos užbaigimui.
Information→Draw Graph: įjungia/išjungia grafo piešimą ekrane.
Information→Show Paths: numatomos botų judėjimo trajektorijos rodymas
Information→State Info: informacija apie botų elgseną (puolimas, gynybą, ...).
Information→Kills/Deaths ratio: informacija apie botų surinktą taškų skaičių.
About→Program/Authors: informacijai apie programą ir autorius peržiūrėti.
Mygtukai: ***Start, Step, Stop, Reset*** – žaidimo eigos valdymui.
Enter/Leave Game – leidžia pačiam žaidėjui įsijungti į žaidimą
Bot Count – leidžia nustatyti žaidime veikiančių botų skaičių

2. *Centrinės žaidimo ekrano dalies pakeitimai*

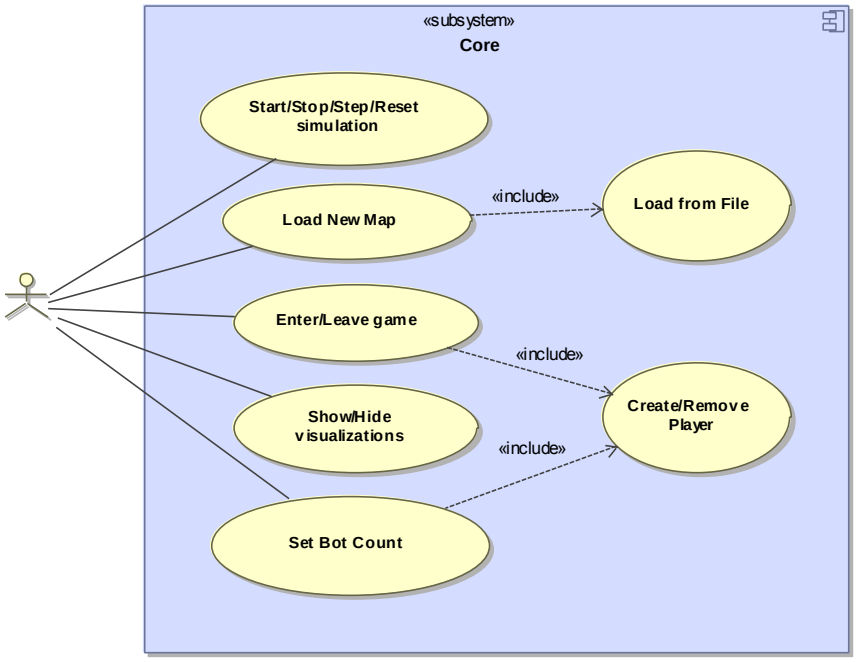
Patobulintas žaidimo lango piešimas (3 pav.) leidžia matyti iššautų šovinių judėjimo kryptį, botų matomumo ribas ir žiūrėjimo kryptį, judėjimo trajektorijas, judėjimo grafo tinklą. Viskas perpiešiama realiu laiku atnaujinus žaidimo simuliaciją. Apie taiklų šūvį praneša raudonos spalvos informacinis tekstas, nurodantis priešui padarytą žalą. Žalias informacinis tekstas pasirodo žaidėjui paėmus bet kokį daiktą – tekste nurodomas paimtas kiekis. Vartotojui valdant savo žaidėją nuo šiol atsiranda apvalus pelės judesius atkartojantis taikiklis, kad būtų lengviau nusitaikyti ir judėti pasirinkta kryptimi. Papildoma informacija(*surinktų taškų ir mirčių skaičius*) nuo šiol pateikiama ir šalia judančių botų-žaidėjų

3. *Apatinė lango dalis*

Informacinė juosta esanti ekrano apačioje praneša apie pasirinktą valdymo režimą ir vartotojo atliekamus veiksmus.

III. UML diagramos

1. Panaudos atvejų diagrama



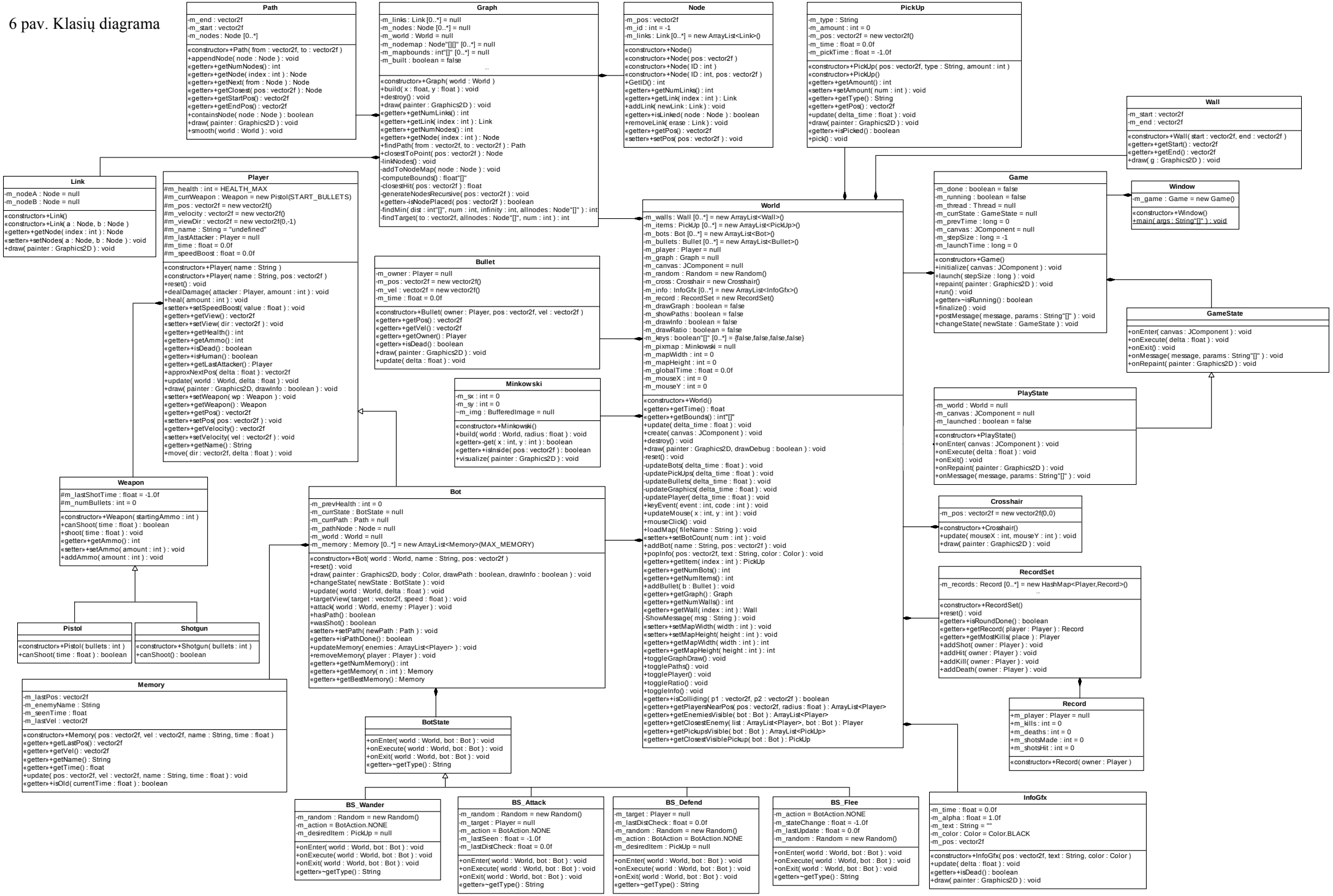
5 pav. Panaudos atvejų diagrama

1.1. Aprašymas

Pateikiama vartotojo sąveika su sistema. Iš diagramos (1 pav.) matome, kad vartotojas gali atlikti tokias funkcijas:

1. *Pradėti žaidimą iš naujo, jį sustabdyti, vykdyti pažingsniui, pradėti iš naujo*
2. *Pasirinkti naują žemėlapį, kuriame vyks žaidimo eiga*
    - pasirinkimas vyksta vartotojui nurodžius žaidimo žemėlapio failą, kurį sistema užkrauna ir paruošia vykdymui
3. *Pačiam dalyvauti-prisijungti prie žaidimo(valdyti savo žaidėją) arba stebėti autonominį botų žaidimą*
    - šis veiksmas atitinkamai sukuria arba pašalina vartotojo valdomą žaidėją iš simuliacijos
4. *Išjungti/įjungti skaičiavimų vizualizavimą: grafo tinklo piešimą, botų trajektorijos rodymą ir kt.*
5. *Pridėti į žaidimą daugiau botų ar pašalinti juos iš jo*
    - šis veiksmas atitinkamai sukuria arba pašalina kompiuterio valdomus žaidėjus iš simuliacijos

6 pav. Klasių diagrama



2.1. Aprašymas

Klasių diagramoje (6 pav.) pateikiamos sistemos klasės bei jas siejantys ryšiai. Šios diagramos pagrindinių klasių aprašymas:

Game – tai pagrindinė klasė kuruojanti žaidimo vykdymą, užtikrinanti veiksmų eiliškumą. Ja pradedamas žaidimo kūrimas.
World – klasė sauganti žaidimo žemėlapio ir vykdymo informaciją, joje yra pagrindiniai metodai darbui su žemėlapiu: galimybė atnaujinti jį, gauti informaciją apie esančius žaidėjus ir botus, statinius objektus: sienas, daiktus
Graph – tai klasė sukuriantį grafą, reikalingą žaidėjų navigacijai žemėlapyje.

- Konstruktorius: `public Graph(World world)`. World tipo objektas yra susiejamas su esamo žaidimo žemėlapiu.
- Pagrindiniai parametrai:
    1. `ArrayList` tipo objektuose saugo informaciją apie grafo viršūnes *Node*.
    2. `ArrayList` tipo objektuose saugo informaciją apie grafo ryšius *Link*.

Player – bazinė klasė informacijai apie žaidėją saugoti bei valdyti.

- Konstruktoriai*:* `public Player(String name)` leidžia sukurti naują žaidėją su vardu
  `public Player( String name, vector2f pos )` su vardu ir koordinatėmis žemėlapyje.
- Pagrindiniai parametrai:
    1. žaidėjo gyvybės taškai
    2. ginklas, pozicija
    3. vardas
    4. greitis

Bot – informacijai apie autonomiškai valdomą žaidėją-botą laikyti. Paveldi klasę ***Player***.

- Konstruktorius `public Bot( World world, String name, vector2f pos )` leidžia žemėlapyje sukurti botą su vardu ir pradine padėtimi.
- Pagrindiniai parametrai:
    1. būsena
    2. kelias iki norimo pasiekti objekto
    3. priešų atmintis

- ***BotState*** – bazinė klasė būsenų klasėms, leidžiančioms valdyti botų logiką. Botų logikos tipai:
    1) *BS_Flee* – būsena aprašanti veiksmus kai botui reikia pabėgti nuo stipresnio priešo
    2) *BS_Defend* – būsena aprašanti veiksmus kai botui reikia apsiginti nuo priešo
    3) *BS_Wander* – būsena aprašanti veiksmus tikroviškesniam botų judėjimui žemėlapyje imituoti
    4) *BS_Attack* – būsena aprašanti veiksmus kai botai naudoja puolamąją taktiką

Weapon – bazinė klasė informacijai apie ginklą laikyti bei manipuliuoti.

- Konstruktorius `public Weapon(int startingAmmo)` leidžia sukurti ginklą su pradiniu šovinių kiekiu.
- Pagrindiniai parametrai:
    1. laikas, kada buvu iššauta paskutinį kartą
    2. šovinių kiekis

PickUp – klasė skirta informacijai apie žemėlapyje esančius gyvybių, bei amunicijos išteklius saugoti.

- Konstruktorius leidžia `public PickUp(vector2f pos, String type, int amount)` sukurti objektą su pradine padėtimi žemėlapyje, tipu, kiekiu.
- Pagrindiniai parametrai:
    1. tipas
    2. kiekis
    3. pozicija.

Papildomų klasių aprašymas:

GameState – klasė atsakinga už dabartinę žaidimo būseną, keičiant būsenas galima valdyti patį žaidimą

Path – klasė sauganti grafo viršūnes(*Node*) pagal kurias skaičiuojama botų judėjimo trajektorija ir navigacija.

- Konstruktorius formuoja kelią nuo pradinės iki galutinės pozicijos.
- Pagrindiniai parametrai:
    1. kelio pradžia
    2. kelio galas
    3. `ArrayList` sąrašas saugantis informaciją per kurios node yra keliaujama

Node – tai informacija apie žemėlapyje, grafe esančias viršūnes.

- Konstruktoriai
    1. public `Node(vector2f pos)` sukurti naują node tik su pozicija
    2. public `Node(int ID)` tik su unikaliu ID
    3. public `Node( int ID, vector2f pos )` su abiem parametrais, pozicija ir ID.
- Pagrindiniai parametrai:
    1. Pozicija
    2. ID

Link – tai informacija apie sąryšį tarp grafo viršūnių, viena jungtis sieja tik dvi skirtingas viršūnes.

- Konstruktorius `public Link(Node a, Node b)` leidžia sukurti Link tipo objektą tarp dviejų node.
- Pagrindiniai parametrai:
    1. pradinis
    2. galinis node.

Bullet – informacija apie iššautas kulkas.

- Konstruktorius `public Bullet(Player owner, vector2f pos, vector2f vel)` sukuria objektą su iššovusiu žaidėju, pradine pozicija, greičiu.
- Pagrindiniai parametrai:
    1. greitis
    2. žala
    3. iššovęs žaidėjęs
    4. pozicija

Crosshair – klasė piešianti taikinuką, kai žaidime žaidžia žaidėjas.

- Pagrindiniai parametras:
    1. taikinuko dydis.

InfoGfx – klasė rodanti papildomus pranešimus, kai žaidėjes ar botas yra sužeidžiamas arba paima „PickUp“.

- Konstruktorius `public InfoGfx(vector2f pos, String text, Color color)` sukuria objektą su rodomo pranešimo pozicija, pačiu pranešimu ir spalva.
- Pagrindiniai parametrai:
    2. gyvavimo laikas
    3. kilimo aukštyn greitis
    4. pranešimo rodymo laikas
    5. pranešimo tekstas
    6. spalva

Memory – saugo informaciją apie botų sutiktus priešus.

- Konstruktorius `public Memory( vector2f pos, vector2f vel, String name, float time )` sukuria objektą su pozicija kurioje botas matė priešą, priešo greičiu, priešo vardu, matymo laiku.
- Pagrindiniai parametrai:
    1. pozicija kurioje botas matė priešą
    2. priešo greitis
    3. priešo vardas
    4. matymo laikas

Minkowski – klasė kuri skaičiuoja Minkowski‘o sumas naudojamas kulkų naikinimui atsitrenkus į sieną ir botų minimaliam atstumui nuo sienos išlaikyti.

- Pagrindinis parametras:
    1. saugo piešinį su spalvomis.

RecordSet – klasė naudojama taškų skaičiavimui ir kaupimui.

- Pagrindiniai parametrai:
    1. žaidėjas
    2. kiek kartų jis nušovė
    3. kiek kartų mirė
    4. kiek iššovė kulkų
    5. kiek kartų pataikė
    6. atitinkamos spalvos

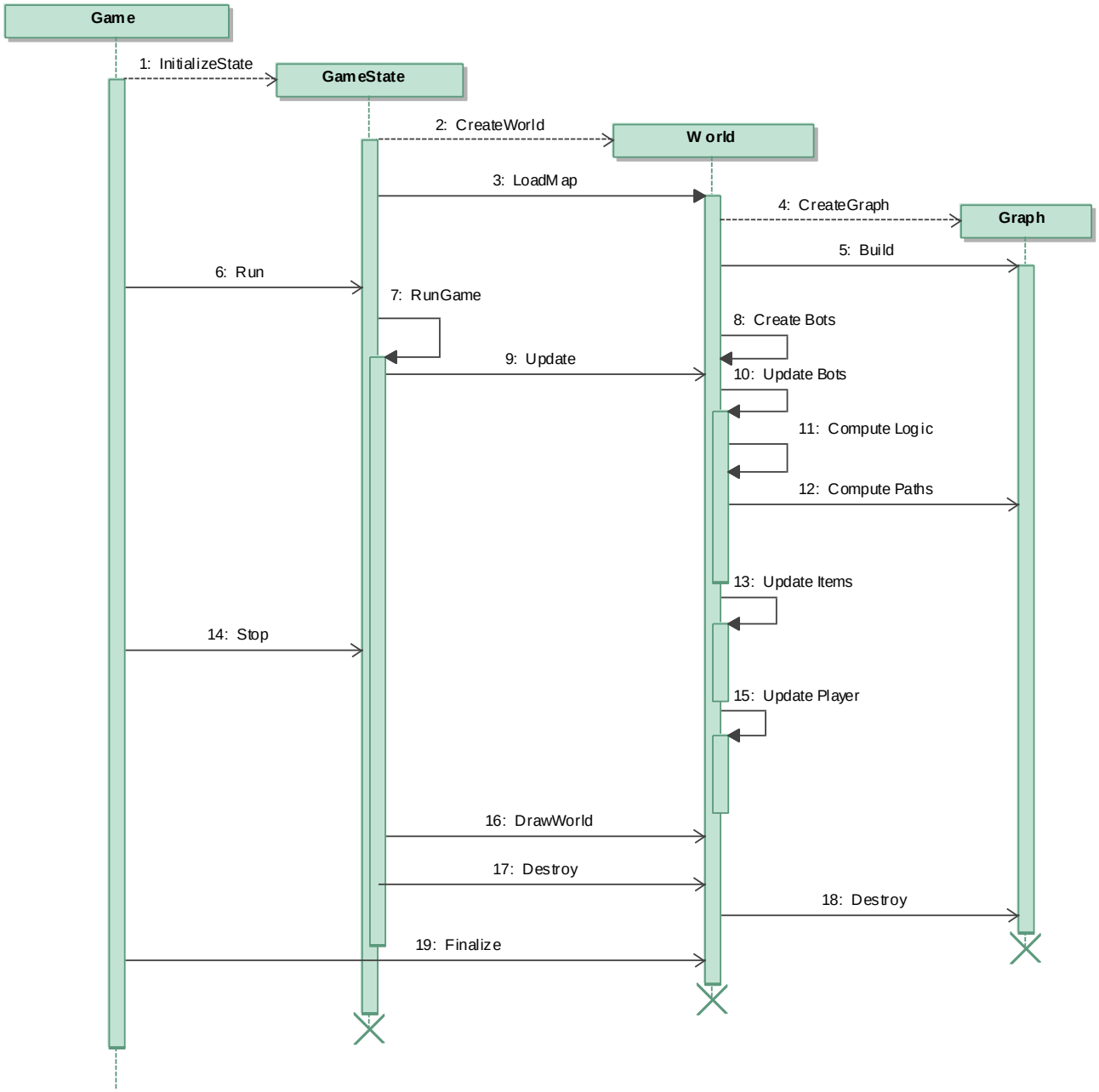
Wall - saugo informaciją apie seinas žemėlapyje.

- Konstruktoriuis `public Wall(vector2f start, vector2f end)` sukuria objektą su sienos pradžia ir galu.
- Pagrindiniai parametrai:
    1. sienos pradžia
    2. sienos pabaiga

Pistol – išvestinė ***Weapon*** lengvojo tipo ginklą(*pistoletą*) aprašanti klasė

- Konstruktorius `public Pistol(int bullets)` sukuria objektą su tam tikru kiekiu kulkų
- Pagrindinis parametras: užtaisymo laikas

2. Sekų diagrama



7 pav. Sekų diagrama

3.1. Aprašymas

Sekų diagramoje (7 pav.) pateikiama sistemos vykdymo schema, klasių sukūrimas chronologine seka.
Žaidimo seką galima suskirstyti į keletą etapų:

I. Pirmas etapas – sukūrimas

Šiame etape programoje sukuriama *Game* klasė, atsakinga už viso žaidimo valdymą. Lygiagrečiai jos kūrimosi etapui sukuriamos kitos pavaldžios klasės – *GameState*, *World*. Pasirinkus žaidimo žemėlapį sudaromas grafas naudojant *Graph* klasės metodą *build()*.

II. Antras etapas – vykdymas

Žaidimo eiga vykdoma iškvietus klasės *Game* metodą *run()*. Šiame metode, kol yra vykdoma žaidimo simuliacija, tol žaidimo procesas yra atnaujinamas kviečiant metodus *update()*. Pirmiausia kviečiama pagrindinėje klasėje *World*, o po to ir šiai klasei pavaldžiuose objektuose. Taip būsenos atnaujinimų grandinė pasiekia visus objektus, perskaičiuodama jų parametrus ir būsenas. Po atnaujinimų vykdomas grafinių objektų piešimas ekrane kviečiant klasių piešimo metodą *draw()*. Baigus piešimą etapo veiksmai kartojami iš naujo.

III. Trečias etapas – sistemos ir žaidimo užbaigimas

Šis etapas vykdomas kai gaunamas stabdymo signalas iš pagrindinės valdančiosios klasės *Game*. Atitinkamai šis signalas perduodamas ir kitoms pavaldžioms klasėms. Šio proceso metu stabdoma žaidimo simuliacija, atlaisvinami panaudoti resursai, baigiamas žaidimo vykdymas.

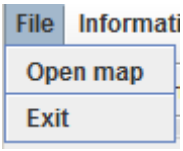
IV. Vartotojo vadovas

Šis Quake 2D šaudyklės žaidimas yra programa, skirta išbandyti įvairius programavimo metodus ir algoritmus. Ji sukurta mokymosi tikslais. Programa leidžia žaisti prieš kompiuterio valdomus žaidėjus analogiškai kaip ir 3D Quake žaidimo variantuose, tik pats žaidimo vaizdas perkeliamas į dvimatę aplinką. Be to programa suteikia galimybę ne tik žaisti žaidimą, bet ir stebėti bei analizuoti botų veiksmus, jų logiką, grafo paieškos algoritmus ir kt.

1. Sistemos funkcionalumo aprašymas

Sistema leidžia pasirinkti bet kokį žaidimo žemėlapį ir pilnai valdytį programos vykdymą, tuo pačiu ir žaidimo eigą. Sistemos valdymo elementai pateikiami toliau.

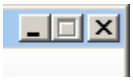
1. *Žemėlapio pasirinkimas*



8a pav. 8b pav.

Pasirinkus ***File->Load*** map iš meniu juostos (8a pav.) galima pakeisti/pasirinkti naują žaidimo žemėlapį. Tą patį galima atlikti ir iš įrankių juostos pasirinkus mygtuką su aplanku (8b pav.).

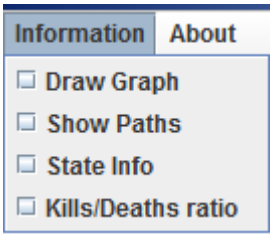
2. *Programos užbaigimas*



9 pav.

Programą galima užbaigti pasirinkus (8a pav.) iš meniu ***File->Exit*** arba paspaudus X mygtuką dešiniajame viršutiniame lango kampe (9 pav.).

3. *Informacijos pateikimo valdymas*

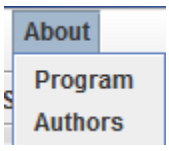


10 pav.

Žaidimo ekrane papildomai informacijai vaizduoti skirtas meniu punktas *Information*. Jame esantys submeniu mygtukai (10 pav.) įjungia ir išjungia pasirinkto elemento rodymą.

Draw graph – grafo tinklo peišimo vaizdavimas (14a pav.) linijomis ir taškais
Show Paths – numatomos botų trajektorijos vaizdavimas (14b pav.) laužtėmis
State Info – rodomas botų būsenos (14d pav.) jų apskritimų centruose
Kills/Deaths ratio – rodomas žaidėjų surinktų taškų ir mirčių skaičius prie žaidėjo apskritimo (14c pav.)

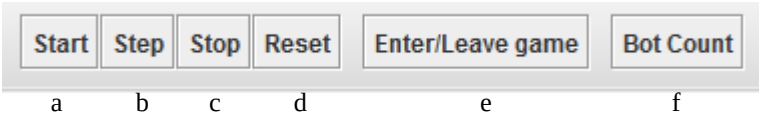
4. *Informacija apie programą*



11 pav.

Papildoma informacija (11 pav.) apie programą, jos valdymą ir autorius pateikiama meniu About skiltyje.

5. *Valdymo juostos funkcionalumas*

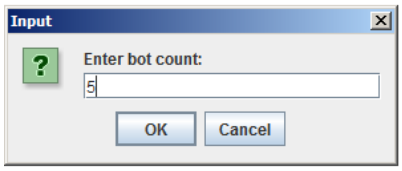


a b c d e f

12 pav.

Valdymo juosta(12 pav.) leidžia keisti svarbiausius žaidimo parametrus – pradėti/baigti žaidimą, paleisti iš naujo ir kt. Mygtukų funkcionalumo aprašymas pateikiamas žemiau:

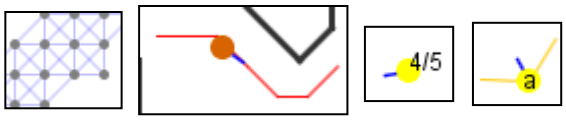
a) ***Start*** – žaidimo paleidimas. Užkrovus naują žemėlapį galima paleisti žaidimą, kuriame automatiškai žais 3 kompiuterio valdomi žaidėjai. Taip pat žaidimą galima paleisti, jei žaidimas buvo laikinai sustabdytas paspaudus Stop mygtuką.
b) ***Step*** – žaidimo vykdymas pažingsniui(*sulėtintai*) po 1 sekundę, praėjus šiam laikui vykdymas vėl sustabdomas žaidimo būsenos analizei. *Pastaba: Galima vykdyti pažingsniui tik kai žaidimas yra sustabdytas.*
c) ***Stop*** – žaidimo eigos sustabdymas, visi vykdomi metodai laikinai sustabdomi. Vėl paleisti galima paspaudus start mygtuką.
d) ***Reset*** – žaidimo atstatymas į pradinę būseną. Išvaloma taškų kaupimo sistema, botai išdėstomi atsitiktinai žemėlapyje, pašalinamos anksčiau iššautos kulkos, pradedamas žaidimo raundas iš naujo.
e) ***Enter/Leave game*** – leidžia vartotojui įvesti savo valdomą žaidėją į žaidimą arba pašalinti jį iš jo. Pridėti/išimti galima bet kuriuo žaidimo metu. Funkcionalumas matomas tik kai žaidimas yra paleistas. *Pastaba*: *Pridedant ar pašalinant žaidėją automatiškai sukaupti taškai anuliuojami ir pradedamas naujas žaidimo roundas.*
f) ***Bot Count*** – galimybė keisti žaidime dalyvaujančių kompiuterio valdomų žaidėjų skaičių. Paspaudus mygtuką iškviečiamas įvedimo langelis (13 pav.), kuriame prašoma įvesti norimą botų skaičių. Sistema priima didesnius nei 0 skaičius.



13 pav. Įvedimo dialogas

6. *Žaidimo ekrano funkcionalumas*

Žaidimo ekrane galima pateikiama daugiausiai svarbios informacijos. Siekiant suprasti informacijos reikšmę, vykdomus veiksmus ir naudojamus metodus būtina žinoti, ką reiškia kiekvienas programos lango grafinis komponentas.



a b c d

14 pav.

Grafinių objektų aprašymas pagal (14 pav.):

a) *Grafo tinklas* – tai tiesių(*sąryšių*) ir taškų aibė sudaranti galimybę botams judėti žemėlapyje.

b) *Botų trajektorijos* – tai laužtės, rodančios numatomą boto judėjimo kelią iki tam tikro taško ar objekto.

c) *Taškų skaičiavimas* – prie kiekvieno žaidėjo esantys skaičiai nurodo sunaikintų priešų kiekį/mirčių kiekį.

d) *Boto būsena ir matymo kampas* – logikos būsena apsprendžia kokią taktiką naudoja botas, ją rodo žaidėjų apskritimų centruose būsenos pirmoji raidė. Boto galimos būsenos:

- **Ataka**(„a“ - *attack*)

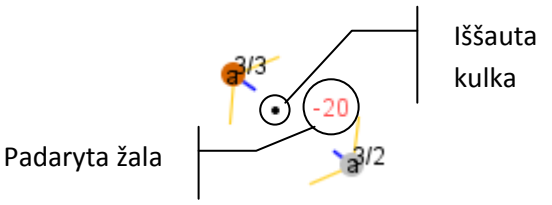
  Botas šioje būsenoje yra agresyvus, nerenka vaistinėlių ir amunicijos, o persekioja savo auką, kol auka nežūsta arba pats neišgyvena. Siekia išlaikyti minimalų atstumą iki priešo, kad būtų didesnė tikimybė pataikyti.
- **Gynyba**(„d“ - *defense*)

  Šioje būsenoje botas stengiasi rinkti visus matomus daiktus(amuniciją, vaistinėles). Stengiasi išvengti konfliktinių situacijų, išlaiko atstumą arba bėga nuo atakuojančio priešo, atsišaudo atgal.
- **Klajojantis**(„w“ - *wandering*)

  Būsena kai botas renka visus matomus daiktus, laisvai klajoja po žemėlapį, jei yra atakuojamas – ginasi, jei mato priešus ir yra stiprus – puola.
- **Pabėgimas**(„f“ - *flee*)

  Botas stengiasi pabėgti ir pasislėpti nuo bet kokio priešo, siekia kuo ilgiau išgyventi.

Boto matomumo kampas(*120 lapsnių*) rodomas dviem linijomis išeinančiomis iš apskritimo. Priešai ir objektai nepatenkantys į boto regėjimo sritį yra jam nematomi.



15 pav.

16 pav.


17 pav.

Kiti grafiniai elementai pagal (15-17 pav.)

- **Žalos, daikto paėmimo indikacija** – kulkai pataikius į priešą arba žaidėjui paėmus daiktą parodomas tekstinis pranešimas, nurodantis padarytos žalos dydį(raudonas tekstas) arba paimto kiekio dydį(žalias tekstas).
- **Iššautos kulkos** – vaizduojamos mažai juodais apskritimais, lekiančiais iššauta trajektorija.
- **Daiktai** – vaistinėlės ir amunicija vaizduojami kaip parodyta (16 pav.). Amunicija duoda žaidėjui nurodytą kiekį šovinių, o vaitinėlė – pagydo.
- Taikiklis – vartotojo žaidėjui įsijungus į žaidimą atsiranda apvalus raudonas taikiklis(17 pav.), leidžiantis tiksliau nusitaikyti ir manevruoti žaidėją.

7. *Tipiniai vartotojo scenarijai*

Keletas programos naudojimo scenarijų, kuriais parodomos pagrindinės programos galimybės:

Scenarijus Nr. 1

1) Vartotojas paleidžia programą
2) Išsirenka žemėlapį paspausdamas ant žemėlapio pasirinkimo mygtuko
3) Paspaudžia *Start* mygtuką žaidimui paleisti, stebi kaip žaidžia botai
4) Pasirenka papildomos informacijos vaizdavimą per *Information* meniu punktą, analizuoja pateiktą medžiagą
5) Uždaro programa per *File->Exit*

Scenarijus Nr. 2

1) Vartotojas paleidžia programą
2) Išsirenka žemėlapį paspausdamas ant *File->Open map*
3) Nustato norimą botų skaičių paspausdamas ant *Bot Count* mygtuko ir įvesdamas atitinkamą skaičių
4) Paleidžia žaidimo vykdymą paspausdamas ant *Start* mygtuko
5) Prisijungia į žaidimą su savo žaidėju paspausdamas mygtuką *Enter/Leave game*
6) *Sustabdo žaidimą spausdamas ant mygtuko Stop*
7) *Keletą kartų vykdo žaidimą pažingsniui spaudžiant mygtuką Step*
8) Spaudžia *Reset* ir *Start* ir toliau tęsia žaidimą
9) Uždaro programa per *File->Exit*

V. Instaliavimo vadovas

1. Reikalavimai kompiuterio programinei įrangai

Norint sėkmingai naudoti šią programinę įrangą, reikalinga *Java SE Runtime Environment 6 Update 16* arba kitos naujesnės versijos, kurią nemokamai galima parsisiųsti iš *Sun Developer Network* svetainės adresu: http://java.sun.com/javase/downloads/index.jsp

2. Reikalavimai kompiuterio techninei įrangai

1. Vaizduoklis
2. Klaviatūra
3. Pelė

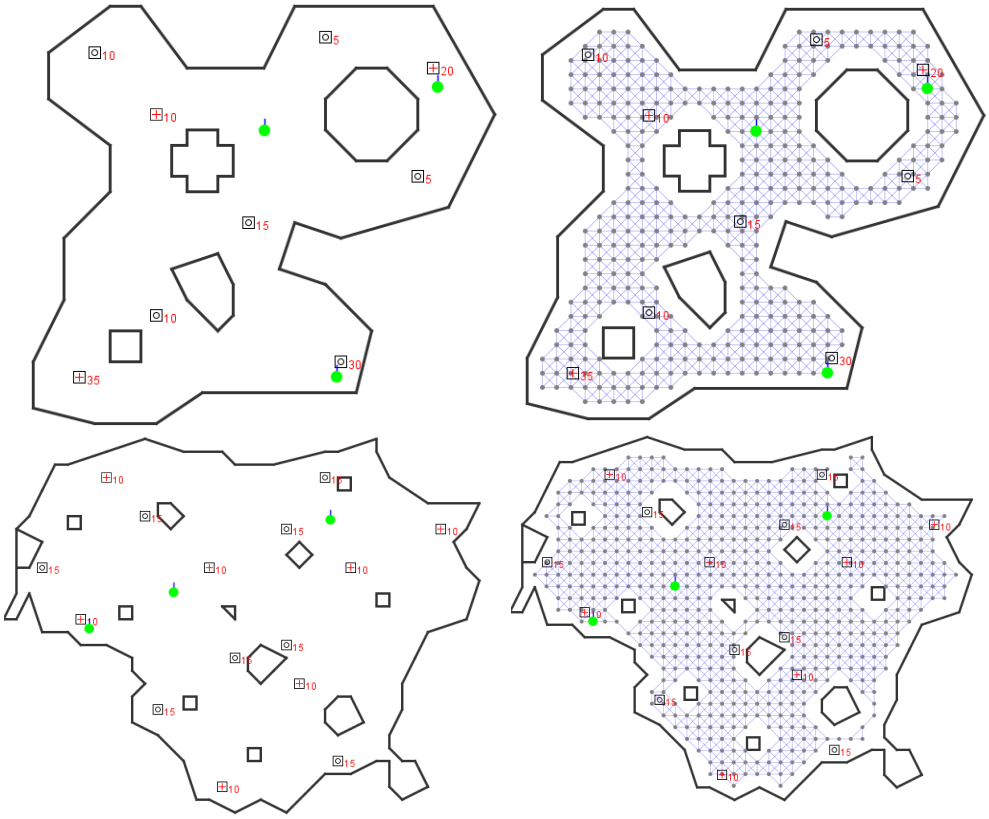
3. Instaliavimas iš pridedamo CD

1. Perkopijuokite failus iš pridedamo CD katalogo „\UktverisT_ValysR_D07\JAVA\dist“ į jums patinkantį katalogą savo kompiuteryje.
2. Norėdami pradėti žaidimą paleiskite „*run.bat*“ iš katalogo į kurį nukopijavote failus
3. Taip pat CD kataloge „\UktverisT_ValysR_D07\DUOMENYS\Map Editor\dist“ rasite programinę įrangą žemėlapių kūrimui bei taisymui.

4. Norėdami taisyti esamus ar kurti naujus žemėlapius perkopijuokite failus iš pridedamo CD katalogo „\UktverisT_ValysR_D07\DUOMENYS\Map Editor\dist“ į jums patinkantį katalogą savo kompiuteryje.
5. Norėdami paleisti žemėlapių redaktorių paleiskite „*run.bat*“ iš katalogo į kurį nukopijavote failus.

VI. Sistemos testavimo rezultatai

1. Testų pavyzdžiai ir siekiami tikslai

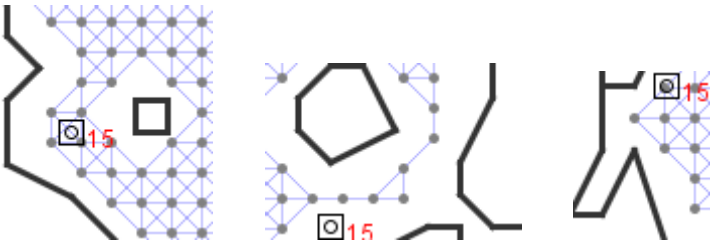


18 pav. Testo žemėlapiai ir jų grafo tinklas

Testavimui panaudoti žemėlapiai *test.map* ir *ltu.map* kaip parodyta (18 pav.). Testais siekta įvertinti tokius aspektus:

1. Grafo vientisumą, teisingumą
2. Botų judėjimo trajektorijas(ar randa trumpiausią kelią, ar jo laikosi judant)
3. Kliūčių vengimą
4. Tikslų siekimą (botų būsenos teisingas pasirinkimas, laikymasis)
5. Ar botai mato objektus (vaistinėles, šovinius) ir gali jus paimti
6. Ar botai gali sekti/atakuoti žaidėją
7. Ar nėra perėjimo kiaurai per sienas

2. Testų rezultatai



19 pav. Svarbios detalės

1. Padidintos žemėlapio *ltu.map* dalys (19 pav.)leidžia matyti, kad grafas yra generuojamas teisingai. Siaurose dalyse, kur botas telpa, yra ryšiai tarp grafo taškų, vadinamais *Node*. Vietose, kurios yra per siauros, kad botas tilptų ryšių nėra.
2. Sugeneruojamas grafas yra vientisas.
3. Testuojant žaidime pastebėta, jog botų trajektorijos yra generuojamos teisingai, t.y. randamas trumpiausias atstumas grafe, nėra ėjimo per sienas.
4. Botai analizuodami aplinką ir turimą amuniciją/gyvybes siekia skirtingų tikslų į skaičiavimus įtraukdami ir matomus objektus, bei priešus.
5. Kulkos pataikiusios į žaidėją arba susidūrusios su kliūtimi išnyksta.

Žaidimas testuotas sėkmingai, testavimo ir žaidimo eigoje klaidų nepastabėta.

VII. Išvados

1. Sistema suprojektuota bei ištestuota sėkmingai, įgyvendinti užsibrėžti tikslai ir metodai.
2. Sukurta programa puikiai pritaikoma edukaciniais tikslais analizuojant įvairius autonominių botų projektavimo ypatumus ir problemas.
3. Java programavimo kalbos pasirinkimas pasiteisino įgyvendinant šį projektą universalumo ir multiplatformiškumo atžvilgiu.
4. Programa suprojektuota ir realizuota taip, kad būtų lengva toliau ją plėsti ir tobulinti, pritaikyti kitiems poreikiams ir užduotims atlikti
5. Programos kūrimo metu naudota versijų kontrolės sistema, todėl pasiekta geresnė programos kodo kokybė bei pasiektas geresnis darbo našumas
6. Pradinis programos projektavimo etapas padėjo išvengti daugelio nesuderinamumo klaidų ir sąryšių supainiojimo
7. Praktikos projektas padėjo pagilinti praktinius programavimo ir projektavimo įgudžius, iki galo realizuoti sudėtingesnį projektą

VIII. Literatūra

[1] Mat Buckland, Programming Game AI by Example. Wordware Publishing, Texas, 2005.

[2] Kostas Plukas, Eugenijus Mačikėnas, Birutė Jarašiūnienė, Irena Mikuckienė. Taikomoji diskrečioji matematika: vadovėlis. Kaunas: Technologija, 2008.

[3] David Brackeen, Bret Barker, Laurence Vanhelsuwé, Developing Games in Java. New Riders Publishing, 2003.

IX. Priedai

Pateikiamų priedų sąrašas:

1. Įvertintos visų etapų ataskaitos
2. Kompaktinis diskas(CD) su programomis ir dokumentacija