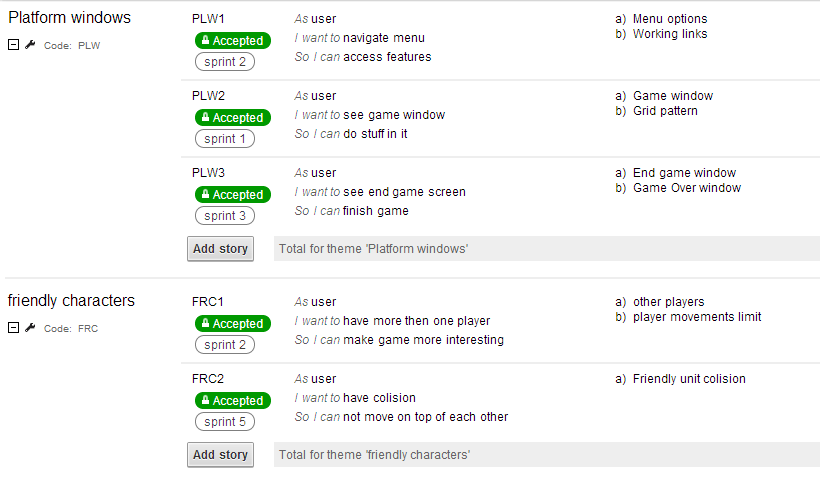
|  |
| --- |
| **KAUNO TECHNOLOGIJOS UNIVERSITETAS**  **INFORMATIKOS FAKULTETAS**    **Savarankiško darbo užduoties ataskaita**  *PLAT* |
| Darbą atliko: Paulius Ambroza IFE-1  Tomas Girčius IFE -1  Lukas Juškauskas IFE -1  Andrius Barčas IFK -1  Darbą tikrino: doc. Tomas Blažauskas |
| Kaunas, 2013 |

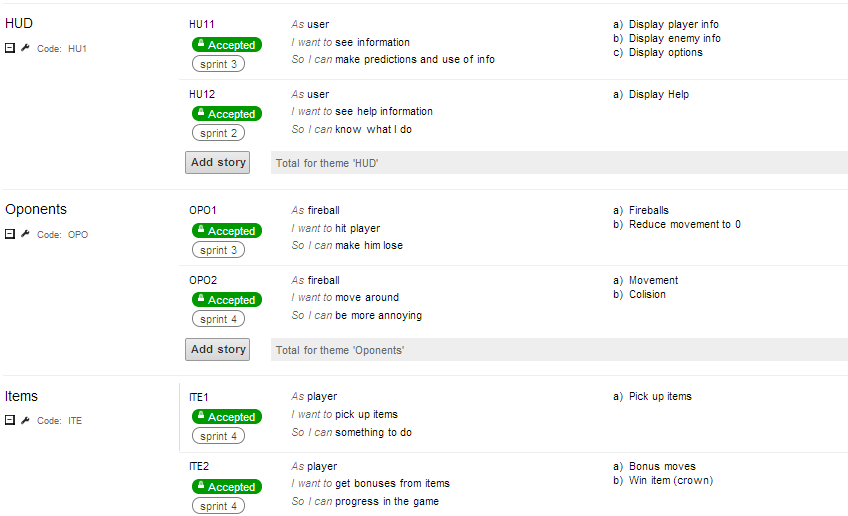
# Projekto užduotis

## Suprogramuoti žaidimą, kuriame valdysime tris veikėjus, kurie turi ribotą ėjimų skaičių. Ėjimų skaičius generuojamas automatiškai atsitiktine tvarka. Žaidimo esmė, rinkti žemelapyje išdėliotus daiktus, kadangi kiekvienas daiktas prideda daugiau ėjimu. Laimima tada, kai vienas is 3 veikėju pakelia nuo žemes daiktą su karūnos ikonėle. Kad nebūtų taip lengva visko surinkti, po žemėlapį „vaiksčioja“ ugnies kamuolys. Kai veikėjas paliečia ugnies kamuolį, traktuojama kad žaidimas pralaimėtas. Taipogi, kai visi trys žaidėjai nepasiekia karūnos žaidimas skelbiamas pralaimėtu.

# Projekto planas

## Projekto įgyvendinimui naudojama tokie įrankiai kaip “ easybacklog“ ir „GitHub“. Projektas buvo išanalizuotas komandos, atsižvelgiant į užduoties reikalavimus užduotis buvo išskirstyta į dalis ir kekvienai daliai buvo priskirtas už ją atsakingas asmuo.





Projekto darbo grafikas

# Programos paskirtis, analizė, detalus apibūdinimas

Šis programavimo inžinerijos darbas priklauso informatikos studijų programai.

## Darbo problematika ir aktualumas

Atliekant šį darbą, pagrindinė nagrinėjama problema yra loginio žaidimo tobulinimas, bei pritaikymas ivairių amžių grupių asmenims.

Įvairaus amžiaus grupių žmonių, laisvalaikio praleidimas, bei tuo pačiu loginio mąstymo vystimas žaidžiant, „turn base strategy“(ėjimų valdomą, logini) žaidimą.

## Darbo tikslas ir uždaviniai

Darbo tikslas – pritaikyti programinės kalbos (Java) specifikavimo, projektavimo, kurimo bei diegimo žinias, tobulinti grupinio darbo gebėjimus, darbo kokybes tobulinimas.

**Darbo uždaviniai**:

1. Atlikti išsamią užduoties analizę.
2. Surašyti tikslu į įrankį : easybacklog.
3. Sukurti projekto platformą.
4. Sukurti HUD.
5. Sukurti priešininkus.
6. Sukurti daiktus.
7. Parengti vartotojo vadovą.

## Dokumentacijos struktūra

Darbą sudaro 7 skyriai. Pirmajame pateikiame pradinė projekto darbo užduotis. Antrajame nurodoma darbo planas, pateikiami grupės easybacklog projekto pavyzdžiai. Kitame skyriuje pateikiama programos paskirtis, analizė bei detalus planas. Ketvirame aprašomi reikalavimai , funkciniai ir nefunkciniai reikalavimai. Penktame pateikiamos Panaudos bei veiklos diagramos, taipogi viso projekto pristatymas. Priešpaskutiniame pateikiama testavimai, rastos klaidos, bei klaidos kurios ištaisytos.Paskutiniąjąme vartotojo dokumentacija, aprašomas aktualus projekto valdymo funkcionalumas(vadovas).

## 4.Pirkėjai

Pirkėjas – projekta perkantis asmuo. Kadangi projektas yra pritaikytas rinkai jį gali įsigyti besidominčios kompanijos. Pirkėju gali tapti, bet kuri besidominti kompanija, kuri gilinasi žaidimo platformų, bei žaidimų sritimi.

Pirkėjai gali teikti reikalavimus projektui. Atsižvelgiant į aplinkybes gali būti įgyvendinti jų reikalavimai.

## 5.Vartotojas

Vartotojai – asmenys, kurie naudojasi programa.

Pagrindinė rinka yra mokiklinio ir universiteto studentai. Rinka buvo paringta atkreipiant dėmesį į šios amžaius grupės asmenų poreikius.

.

# Reikalavimo Aprašymas

Šiame skyriuje detaliai aprašomi visi funkciniai ir nefunkciniai reikalavimai bei sistemos apribojimai. Reikalavimų tipai:

* F - funkcinis reikalavimas
* NF - nefunkcinis reikalavimas

## 1.Funkciniai reikalavimai

Reikalavimas NR.:1

Reikalvimo tipas:

Aprašymas: Projektas turi turėti menių grafine sąsaja.  
Pagrindimas:Patogus varotojui naudoti menių.  
Šaltinis: Užsakovas  
Atitikimo kriterijus: Sistema turi veikiančia menių grafinę sąsaja.

Reikalavimas NR.:2

Reikalvimo tipas:

Aprašymas: Projektas turi turėti kelis lygius.  
Pagrindimas: Pasiekus objekta, turi būti persijungimas į sudetingesnį lygį  
Šaltinis: Užsakovas  
Atitikimo kriterijus: sudėtingumas, bei projekto testinumas.

Reikalavimas NR.:3

Reikalvimo tipas:

Aprašymas: Projektas turi turėti veikėjus.  
Pagrindimas: Vartotojas turi galėti valdyti veikėją.

Šaltinis:Užsakovas.

Atitikimo kriterijus:Vartotojas gali valdyti veikėjus.

Reikalavimas NR.:4

Reikalvimo tipas:

Aprašymas: Projektas turi turėti priešus.  
Pagrindimas: Dirbtinio intelekto valdomas oponentas.  
Šaltinis: Užsakovas  
Atitikimo kriterijus:Vartotojas gali matyti priešus, bei su jais susidūrti.

Reikalavimas NR.:5

Reikalvimo tipas:

Aprašymas:Projektas turi turėti išdėstytus žemėlapyje objektus(daiktus)  
Pagrindimas:Pasiekti, bei paimti daiktą  
Šaltinis: Užsakovas  
Atitikimo kriterijus: Projektas turi objektus.

## 2.Nefunkciniai reikalavimai

Reikalavimas NR.:1

Reikalvimo tipas:

Aprašymas:Projekte turi buti grafinis dizainas  
Pagrindimas:Vartotojas turi matyti veikėjus, priešus, daiktus.  
Šaltinis: Užsakovas  
Atitikimo kriterijus: Grafinis dizainas

Reikalavimas NR.:2

Reikalvimo tipas:

Aprašymas:Projekte turi buti paprastai valdomas  
Pagrindimas:Vartotojas gali valdyti veikėjus spausdamas aukštyn, žemyn, kairėn, desnėn.  
Šaltinis: Užsakovas  
Atitikimo kriterijus: Paprastai valdomi veikėjai

Reikalavimas NR.:3

Reikalvimo tipas:

Aprašymas:Projekte turi veikti be strigdžių  
Pagrindimas:Vartotojas turi tenkinti projekto veikimas.  
Šaltinis: Užsakovas  
Atitikimo kriterijus: Projektas maksimaliai ištestuotas

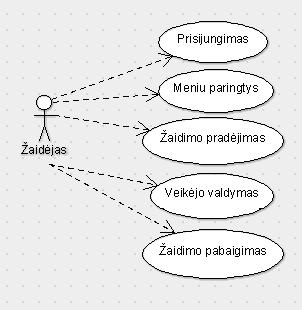
Reikalavimas NR.:4

Reikalvimo tipas:

Aprašymas:Projekte turi veikti neapkraunant opracines sistemos  
Pagrindimas:Vartotojas turi tenkinti projekto veikimas.  
Šaltinis: Užsakovas  
Atitikimo kriterijus: Projektas maksimaliai ištestuotas

# Projektavimas ir projekto dokumentacija

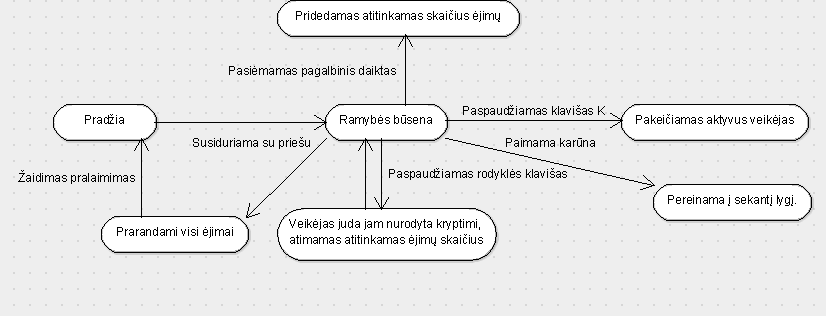
## Panaudos atvėjų diagrama



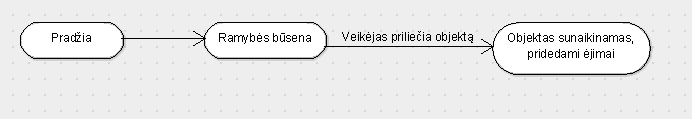
Panaudos atvejų diagrama nusako vartotojų roles

## Veiklos diagramos

**Veiklos diagramos**(angl., Activity Diagrams) vaizduoja pagrindinius sistemos naudotojų darbo scenarijus. Diagramoje esančios juostos vaizduoja aktorius (sistemos vartotojus), kurie vykdo atitinkamas veiklas.



Veikėjo veiklos diagrama.



Daiktu veiklos diagrama

## Projekto Pristatymas

### Meniu

Paleidus projektą išvystame menių langą



Pats programos kodas atrodo taip :

**public** **class** MenuState **extends** GameState {

**private** BufferedImage bg;

**private** BufferedImage marker;

**private** Font font;

**private** **int** currentOption = 0;

**private** String[] options = {

"START",

"QUIT"

};

**public** MenuState(GameStateManager gsm) {

**super**(gsm);

font = **new** Font(

"Century Gothic",

Font.*PLAIN*,

28);

}

**public** **void** init() {

bg = Content.*MENUBG*[0][0];

marker = Content.*MARKER*[0][0];

}

**public** **void** update() {

handleInput();

}

**public** **void** draw(Graphics2D g) {

g.drawImage(bg, 0, 0, **null**);

g.setFont(font);

**for**(**int** i = 0; i < options.length; i++) {

**if**(i == currentOption) {

g.setColor(Color.*WHITE*);

g.drawImage(marker, 340, 375 + i \* 40, **null**);

}

**else** {

g.setColor(Color.*RED*);

}

g.drawString(options[i], 375, 400 + i \* 40);

}

}

**public** **void** handleInput() {

**if**(Keys.*isPressed*(Keys.*DOWN*) && currentOption < options.length - 1) {

currentOption++;

}

**if**(Keys.*isPressed*(Keys.*UP*) && currentOption > 0) {

currentOption--;

}

**if**(Keys.*isPressed*(Keys.*ENTER*)) {

selectOption();

}

}

**private** **void** selectOption() {

**if**(currentOption == 0) {

gsm.setState(GameStateManager.*PLAY*);

}

**if**(currentOption == 1) {

System.*exit*(0);

}

}

}

### Spritai(paveikslai)

#### Veikėju spritai:

Ugnies kamuolys, Naudojamas kaip priešas



Daiktai, pakėlus viena is jų pridedami papildomi ėjimai, o paėmus karūna traktuojama, jog žaidimas laimėtas









Veikėjas. Veikėjas sukūrtas, kaip gif. Todel kiekvienai pozicijai reikalingi 2 paveikslai, jie imituoja judėjimą



#### Žemėlapio spirtai:

Žodės paveikslas



Akmenų paveikslas



Medžio paveikslas



### Veikėjas

Žaidime turime 3 veikėjus, spritų skiltyje parodyti veikėjo paveiksliukai.  
Veikėjas gali judėti į visas keturias puses, vaikšto tik 90 laipsniu kampu, ir skersai judėti negali  
Vienas žingsni, kainuoja 1 judėjimą



Pateikiame žaidimo koda:  
  
**public** **class** zaidejas **extends** Entity {

// sprites

**private** BufferedImage[] downSprites;

**private** BufferedImage[] leftSprites;

**private** BufferedImage[] rightSprites;

**private** BufferedImage[] upSprites;

// animation

**private** **final** **int** DOWN = 0;

**private** **final** **int** LEFT = 1;

**private** **final** **int** RIGHT = 2;

**private** **final** **int** UP = 3;

**private** **int** ejimai;

**public** zaidejas(TileMap tm) {

**super**(tm);

width = 30;

height = 30;

cwidth = 24;

cheight = 24;

moveSpeed = 4;

downSprites = Content.*zaidejas*[0];

leftSprites = Content.*zaidejas*[1];

rightSprites = Content.*zaidejas*[2];

upSprites = Content.*zaidejas*[3];

animation.setFrames(downSprites);

animation.setDelay(10);

}

**private** **void** setAnimation(**int** i, BufferedImage[] bi, **int** d) {

currentAnimation = i;

animation.setFrames(bi);

animation.setDelay(d);

}

**public** **int** getEjimai() { **return** ejimai; }

//Keyboard input. Moves the player.

**public** **void** setDown() {

**super**.setDown(ejimai);

ejimai = **super**.ejimai;

}

**public** **void** setLeft() {

**super**.setLeft(ejimai);

ejimai = **super**.ejimai;

}

**public** **void** setRight() {

**super**.setRight(ejimai);

ejimai = **super**.ejimai;

}

**public** **void** setUp() {

**super**.setUp(ejimai);

ejimai = **super**.ejimai;

}

**public** **void** setAction() {

**if**(currentAnimation == UP && tileMap.getIndex(rowTile - 1, colTile) == 21) {

tileMap.setTile(rowTile - 1, colTile, 1);

// JukeBox.play("tilechange");

}

**if**(currentAnimation == DOWN && tileMap.getIndex(rowTile + 1, colTile) == 21) {

tileMap.setTile(rowTile + 1, colTile, 1);

// JukeBox.play("tilechange");

}

**if**(currentAnimation == LEFT && tileMap.getIndex(rowTile, colTile - 1) == 21) {

tileMap.setTile(rowTile, colTile - 1, 1);

// JukeBox.play("tilechange");

}

**if**(currentAnimation == RIGHT && tileMap.getIndex(rowTile, colTile + 1) == 21) {

tileMap.setTile(rowTile, colTile + 1, 1);

// JukeBox.play("tilechange");

}

}

**public** **void** update() {

// set animation

**if**(down) {

setAnimation(DOWN, downSprites, 10);

}

**if**(left) {

setAnimation(LEFT, leftSprites, 10);

}

**if**(right) {

setAnimation(RIGHT, rightSprites, 10);

}

**if**(up) {

setAnimation(UP, upSprites, 10);

}

// update position

**super**.update();

}

**public** **void** draw(Graphics2D g) {

**super**.draw(g);

}

**public** **void** setEjimai(**int** nr){

**this**.ejimai = nr;

}

**public** **void** addEjimai(**int** sk){

**this**.ejimai += sk;

}

**private** **void** populatezaidejai(){

zaidejas Zaidejas;

Zaidejas = **new** zaidejas(tileMap);

Zaidejas.setTilePosition(3, 1);

zaidejai.add(Zaidejas);

Zaidejas = **new** zaidejas(tileMap);

Zaidejas.setTilePosition(2, 2);

zaidejai.add(Zaidejas);

Zaidejas = **new** zaidejas(tileMap);

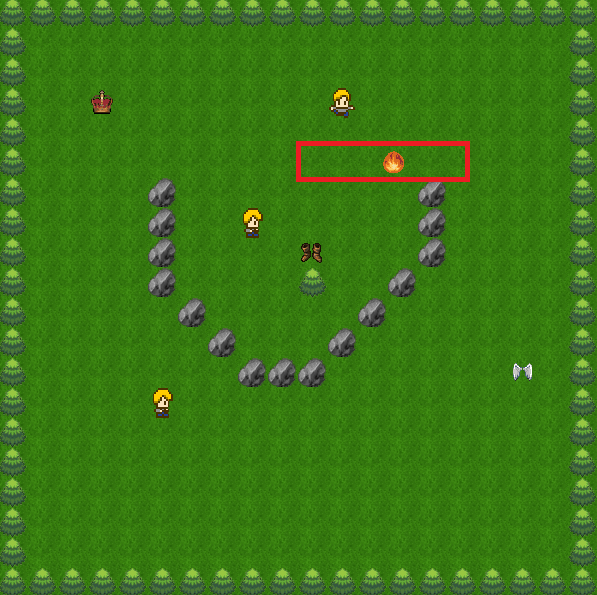
Zaidejas.setTilePosition(1, 3);

zaidejai.add(Zaidejas);

}

### Priešai

Žaidime turime, pirmyn atgal judanti ugnies kamuolį, kuri veikėjui palietus žaidimas buna pralaimėtas.



Kaip parodyta paveikslėlyje, ugnis juda pažymėtuose langeliuose

Štai pateikiame priešų kodą.

**public** **class** Enemy **extends** Entity {

**private** BufferedImage sprite;

**private** **int** type;

**public** **static** **final** **int** *WOLF* = 0;

**int** move = 10;

**public** Enemy(TileMap tm){

**super**(tm);

type = -1;

width = height = 30;

cwidth = cheight = 24;

moveSpeed = 4;

}

**public** **void** setType(**int** i) {

type = i;

**if**(type == *WOLF*) {

sprite = Content.*ENEMY*[0][0];

}

}

**public** **void** enemymove(){

wolfmove();

**if** (type == *WOLF*) wolfmove();

}

**private** **void** wolfmove(){

**if** (move < 6){

**super**.setLeft(move);

move= **super**.ejimai;

}

**if** (move > 5){

**super**.setRight(move);

move= **super**.ejimai;

}

**if** (move == 0){

move = 10;

}

}

**public** **void** draw(Graphics2D g) {

setMapPosition();

g.drawImage(sprite, x + xmap - width / 2, y + ymap - height / 2, **null**);

}

**public** **void** update() {

**super**.update();

}

}

**private** **void** populateEnemies(){

Enemy enemy;

enemy = **new** Enemy(tileMap);

enemy.setType(Enemy.*WOLF*);

//Initialize wolf

enemy.setTilePosition(5, 10);

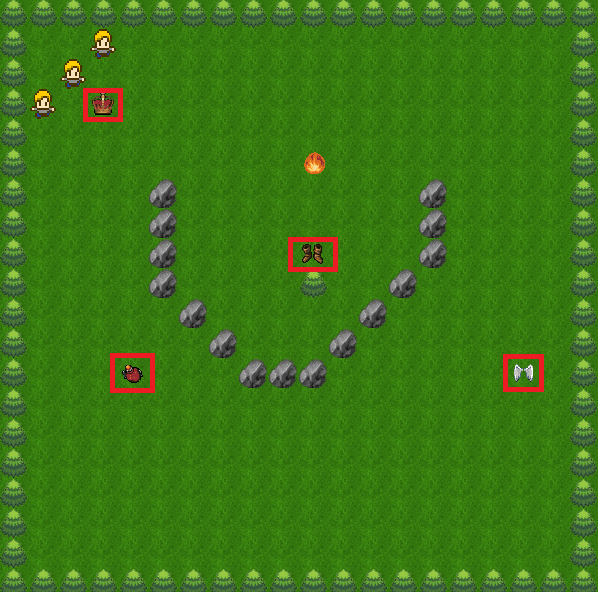
Priesai.add(enemy);

}

### Daiktai

Žemėlapyje išdelioti daiktai, kurie suteikia veikėjui papildomų ėjimų, tam kad veikėjas galėtu pasiekti savo tikslą – karūną. Kai daiktas buna pakeltas nuo žemes, jis žemėlapyje pradingsta ir iki žaidimo pabaigos nebeatsiranda.

Pakelta karūna reiškia laimėta žaidimą



**public** **class** Daiktai **extends** Entity{

**private** BufferedImage sprite;

**private** **int** type;

**public** **static** **final** **int** *BOOTS* = 0;

**public** **static** **final** **int** *POTION* = 1;

**public** **static** **final** **int** *WINGS* = 2;

**public** **static** **final** **int** *CROWN* = 3;

**public** Daiktai(TileMap tm) {

**super**(tm);

type = -1;

width = height = 30;

cwidth = cheight = 30;

}

**public** **void** setType(**int** i) {

type = i;

**if**(type == *POTION*) {

sprite = Content.*DAIKTAI*[0][1];

}

**if**(type == *BOOTS*) {

sprite = Content.*DAIKTAI*[0][0];

}

**if**(type == *WINGS*){

sprite = Content.*DAIKTAI*[1][0];

}

**if**(type == *CROWN*){

sprite = Content.*DAIKTAI*[1][1];

}

}

**public** **void** collected(GameStateManager gsm, zaidejas p) {

**if**(type == *BOOTS*){

p.addEjimai(10);

}

**if**(type == *POTION*){

p.addEjimai(5);

}

**if**(type == *WINGS*){

p.addEjimai(20);

}

**if**(type == *CROWN*){

gsm.setState(GameStateManager.*END*);

}

}

**public** **void** draw(Graphics2D g) {

setMapPosition();

g.drawImage(sprite, x + xmap - width / 2, y + ymap - height / 2, **null**);

}

}

**private** **void** populateItems() {

Daiktai daiktai;

daiktai = **new** Daiktai(tileMap);

daiktai.setType(Daiktai.*BOOTS*);

daiktai.setTilePosition(8, 10);

Irankiai.add(daiktai);

daiktai = **new** Daiktai(tileMap);

daiktai.setType(Daiktai.*POTION*);

daiktai.setTilePosition(12, 4);

Irankiai.add(daiktai);

daiktai = **new** Daiktai(tileMap);

daiktai.setType(Daiktai.*WINGS*);

daiktai.setTilePosition(12, 17);

Irankiai.add(daiktai);

daiktai = **new** Daiktai(tileMap);

daiktai.setType(Daiktai.*CROWN*);

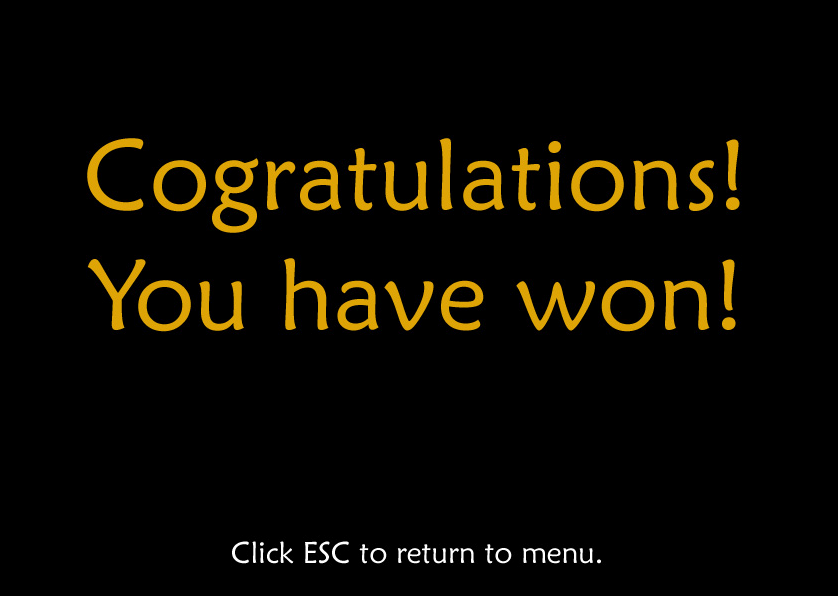
daiktai.setTilePosition(3, 3);

Irankiai.add(daiktai);

}

### Žaidimo pabaiga

#### Laimėjimas



Kaip jau minėjau laimėjimu traktuojama tada, kai dar nespėjus pasibaigti ėjimams buna pakelta karūna nuo žemės. Kai ji pakelta išsoka lentelė, kaip šiektiek aukščiau parodyta.  
Paspaudus ESC mygtuka yra lentelė išjungiama, taipogi sugrįžtama i pradinį menių langą.

**public** **class** EndState **extends** GameState {

**private** BufferedImage End;

**public** EndState(GameStateManager gsm) {

**super**(gsm);

}

**public** **void** init() {

End = Content.*END*[0][0];

}

**public** **void** update() {

handleInput();

}

**public** **void** draw(Graphics2D g) {

g.drawImage(End, 0, 0, **null**);

}

**public** **void** handleInput() {

**if**(Keys.*isPressed*(Keys.*ESCAPE*)) {

gsm.setState(GameStateManager.*MENU*);

}

}

}

#### Pralaimėjimas



Toki langą galite išvysti, jeigu per savo duotus ėjimus nesugebėsite pasiekti ir paimti karūnos arba jeigu prisiliesite prie ugnies.

**public** **class** GameOverState **extends** GameState {

**private** BufferedImage GAMEOVER;

**public** GameOverState(GameStateManager gsm) {

**super**(gsm);

}

**public** **void** init() {

GAMEOVER = Content.*GAMEOVER*[0][0];

}

**public** **void** update() {

handleInput();

}

**public** **void** draw(Graphics2D g) {

g.drawImage(GAMEOVER, 0, 0, **null**);

}

**public** **void** handleInput() {

**if**(Keys.*isPressed*(Keys.*ESCAPE*)) {

gsm.setState(GameStateManager.*MENU*);

}

}

}

# Žaidimo testavimas ir klaidų taisymas.

Žaidimas testuotas naudojant Java Eclipse programavimo aplinkoje.

Ištaisius rastas klaidas žaidimas buvo toliau testuojamas suteikiant žaidimo prieiga testuotojams ir registruojant jų rastus neatitikimus ar žaidimo klaidas.

## Testavimo planas

Testas atliktas norint sužinoti ar žaidimas atlieką jam reikalingas funkcijas.

### Testas NR.1 – tikrinamos menu paringtis.

Testo žingsniai:

1. Įjungti žaidimo aplikaciją.
2. Patikrinti, ar žaidimo paveikslai, teksto stiliu ir ekrano fonas yra toks koks yra reikalingas.
3. Patikrinti, ar spaudžiant aukštyn-žemyn menu ekrane keičiasi žymeklio pozicija.
4. Patikrinti, ar veikia „Start“ mygtukas ir ar pradedamas žaidimas jį paspaudus.
5. Patikrinti, ar veikia „Quit“ mygtukas ir ar programa baigia darbą jį paspaudus.

### Testas NR.2 – tikriname žaidimo variklį.

Testo žingsniai:

1. Paleisti žaidimą per menu sritį.
2. Patikrinti, ar sukuriami visi paveiksliukai, fonas ir šifras yra teisingi ir yra savo vietose.
3. Patikrinti, ar veikėjas gali judėti naudojant klaviatūros rodykles.
4. Patikrinti, ar veikėjas yra sustabdomas jei yra bandoma pereiti per nepereinama objektą.
5. Patikrinti, ar veikėjui žengus žingsnį yra atimamas teisingas kiekis ėjimų.
6. Patikrinti, ar veikėjas nebegali žengti jei jam nebelieka ėjimų.
7. Patikrinti, ar veikėjui priėjus daiktą daiktas tampa sunaikinamas.
8. Patikrinti, ar pridedamas teisingas kiekis ėjimų prie esamų veikėjo ėjimų kiekio.
9. Patikrinti, ar pridėti ėjimai pridedami prie atitinkamo veikėjo ėjimų kiekio.
10. Patikrinti, ar paėmus objektą „karūna“ žaidimas skelbiamas laimėtų.
11. Patikrinti, ar pasirodo teisingas langas laimėjus žaidimą, ar fonai ir paveikslas yra teisingi.
12. Patikrinti, ar paspaudus „esc“ mygtuką esant laimėjimo ekrane yra grįžtama į menu būseną.
13. Patikrinti, ar veikėjui prilietus priešą jo ėjimų kiekis yra sumažinamas iki nulio.
14. Patikrinti, ar visiems veikėjų ėjimams pasiekus nuli ir nepaėmus karūnas žaidimas skelbiamas pralaimėtu.
15. Patikrinti, ar pasirodo teisingas langas pralaimėjus žaidimą, ar fonai ir paveikslas yra teisingi.
16. Patikrinti, ar paspaudus „esc“ mygtuką esant pralaimėjimo ekrane yra grįžtama į menu būseną.

# Vartotojo dokumentacija ir pagalba

Dabar parodysime, kaip reikia naudotis žaidimu. Detaliai aprašysime kiekviena žingsnį, kad būtu lengva ir paprasta naudotis.

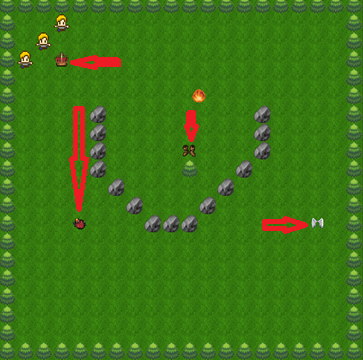
Paleidus programą atidaromas menių langs. Meniu lange galimi 2 pasirinkimai

„START“ ir „QUIT“  
Norint pradėti paspauskite „START“ punktą  
Norint išeiti paspauskite „QUIT“ punktą  
  


Paspaudus „START“ punktą, išvysime žaidimo langą :  
  

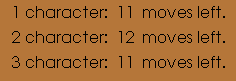
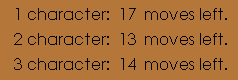

Žaidimo tikslas, pasiekti daiktus, kurie yra išdėstyti ant žemės  
pasiekus karūna traktuojama, kad žaidimas laimėtas.

Visi daiktai, išskyrus karūną suteikia papildomų ėjimų.

Papildomi ėjimai reikalingi, kad toliau galėtumete judėti  


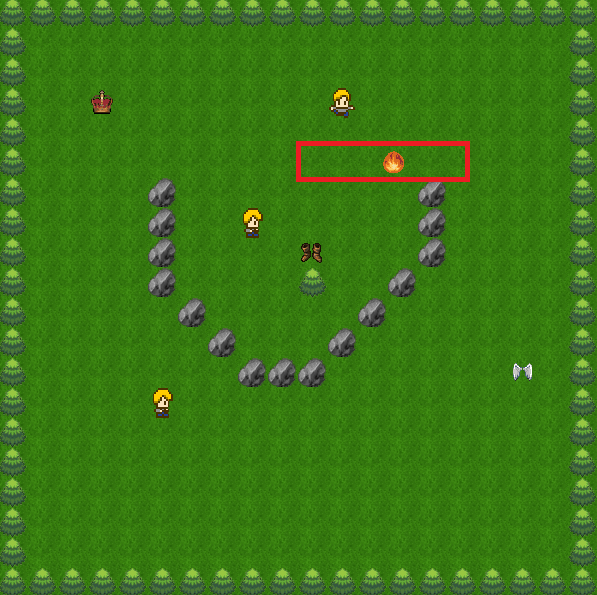
Kiekvienas veikėjas turi tam tikra judėjimu skaičių, pasibaigus visiems judėjimams – žaidimas pralaimėtas, kol nespėjo pasibaigti ėjimai ir būna pakelta nuo žemės karūna – žaidimas laimėtas

Kiekviena nauja partija, steikiamas asitiktinis ėjimų skaičius nuo 10 iki 17  
Pakėlus nuo žemės, kokį nors daiktą ėjimu yra pridedama.

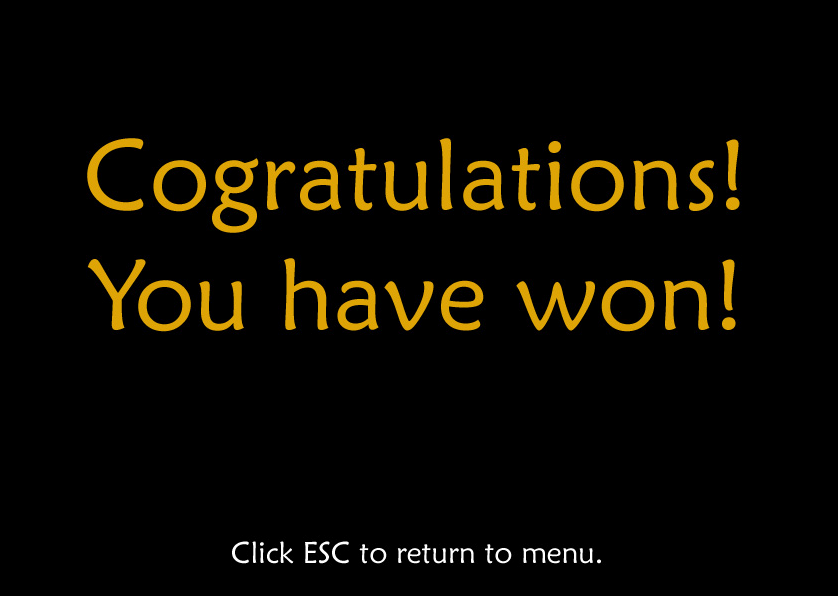


Šiose dvejuose lentelėse parodoma, kad žaidimo pradžioje suteikiamas skirtingas ėjimų skaičius.

Žaidime stengites vengti ugnies kamuolio, ji prilietus žaidimas bus iškarto pralaimėtas



Kai žaidime yra pasiekiama karūna išvystama štai tokį langą, kuris reiškia kad žaidimas laimėtas   
Paspaudus ESC mygtuką, sugrįšite i menių langą.



Jeigu per duotus ėjimus, nepavyko pasiekti karūnos iššoks štai toks langas  
Paspaudus ESC mygtuką sugrįšite į menių langą.

