A Storm of Swords

Catana Ricardo-Marcelin

Grupa 1212B

1.Proiectarea contextului și descriere detaliată:

Într-o lume străveche, în care magia se împletea cu faptele eroice, exista un tărâm întunecat și plin de mister numit Gilead, un loc sumbru și mereu trist ce înca de la apariția primilor oameni a fost bântuit de monștrii. Primii vânatori au apărut în urmă cu multe secole, deoarece era clară necesitatea unor războinici specializați pentru a scăpa de amenințarea monștrilor. Legendele și scrierile înțelepților ne spun că acești vânatori au fost creați printr-un process extrem de dificilor si periculos ce reprezenta stări de suferință extremă si modificări genetice.

La început aceștia erau percepuți mai mult ca niște protectori ai umanității, însă, cu timpul, aceștia au devenit cunoscuți pentru faptul că făceau orice pentru suma corectă de bani.

Roland, unul dintre puținii vânători rămași in viață in aceste zile, s-a făcut remarcat si a ajuns o legendă. Cei o sută de ani de activitate l-au transformat într-un luptător genial si într-un asasin fără milă, acesta alegând sa ajute satele si oamenii neputincioși și să îi taxeze foarte scump pe lorzi si regi, astfel, reușind să câștige inimile oamenilor de rând, dar și ura celor bogați.

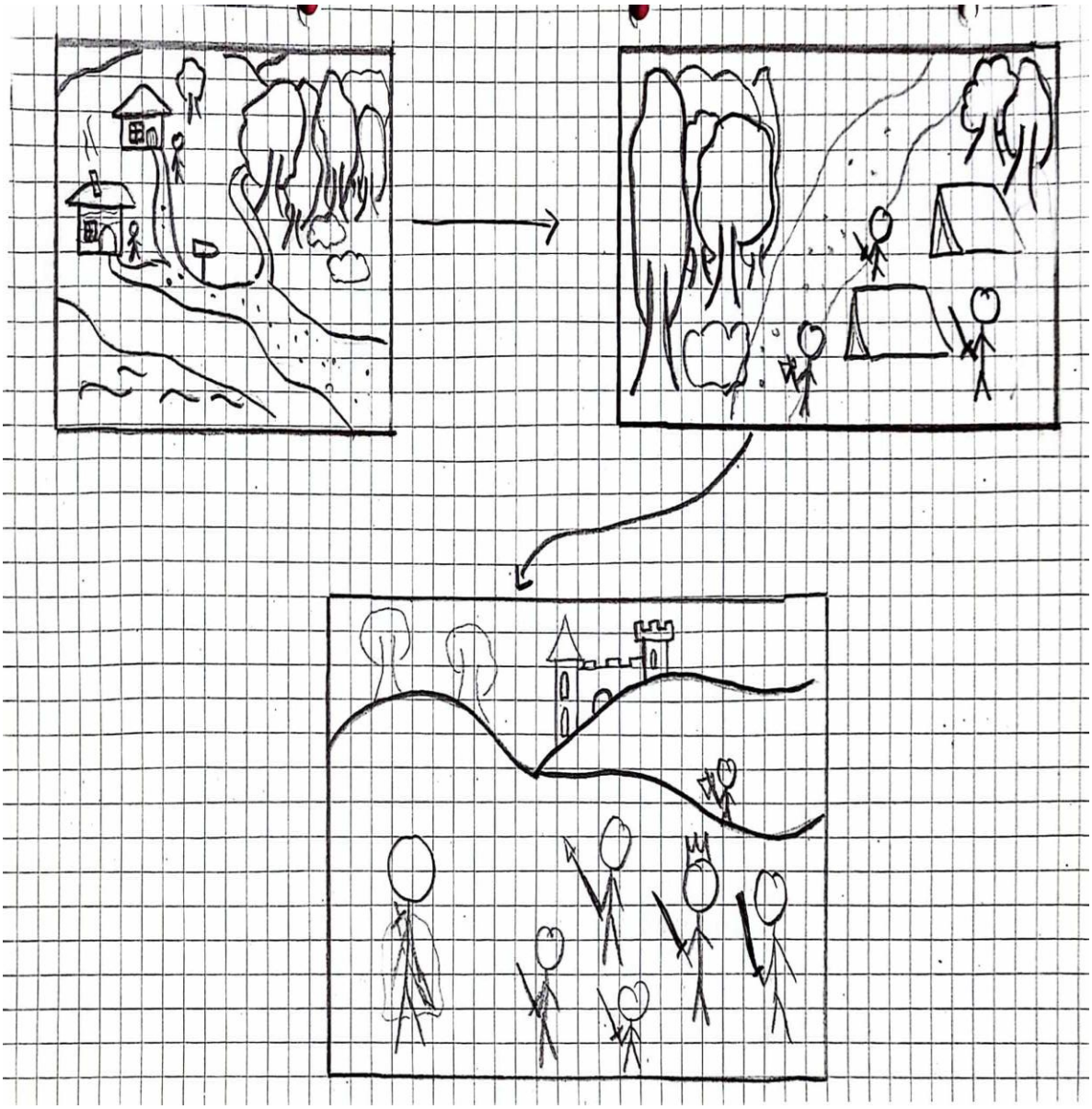
În decursul ultimului an, Roland, împreună cu colegii săi au descoperit ceva straniu: monștrii, încetul cu încetul au început să dispară, până nu a mai rămas niciunul în întreg ținutul.

Așa au ajuns ultimii 6 vânatori să se despartă în încercarea de a afla cauza dispariției monștrilor, fiind toți de acord să se întâlnească în urmatoarea vară în Pădurea Zeiilor Vechi pentru a vedea rodul călatoriei lor.

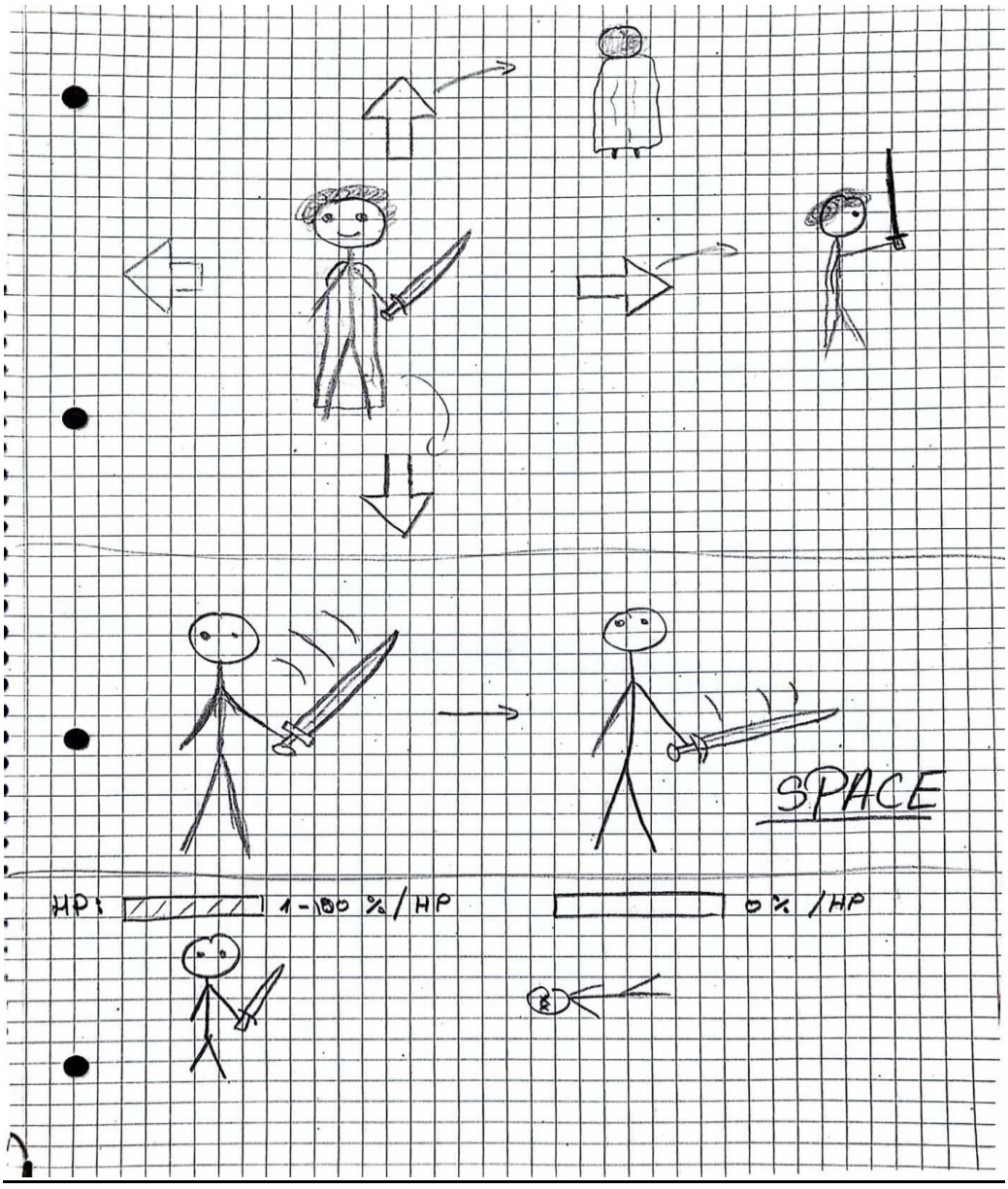
Pe parcursul călatoriei, Roland rămâne singur, după ce tovarășii săi au pierit în urma atacului unei bande de răufăcători. Ajuns în Pădurea Zeiilor, acesta poposește câteva săptămâni în așteptarea celorlalți. Sătul să tot aștepte, pleacă pe urmele lor, și parcurge sute de kilometrii până ajunge într-un sat aflat în apropiere de Rivia. Aici află că vechii lui tovarăși de arme au fost capturați și executați de un lord din apropiere și se decide să îi răzbune .

Imaginea atașată mai sus va fi folosită pentru interfața de pornire a jocului ce va conține un buton “START” pentru a începe jocul, și unul pentru ieșirea din joc.

Genul jocului reprezintă o combinație dintre Aventură și Acțiune, iar scopul este răzbunarea vechilor camarazi de arme.



2. Proiectarea sistemului și descriere detaliată:



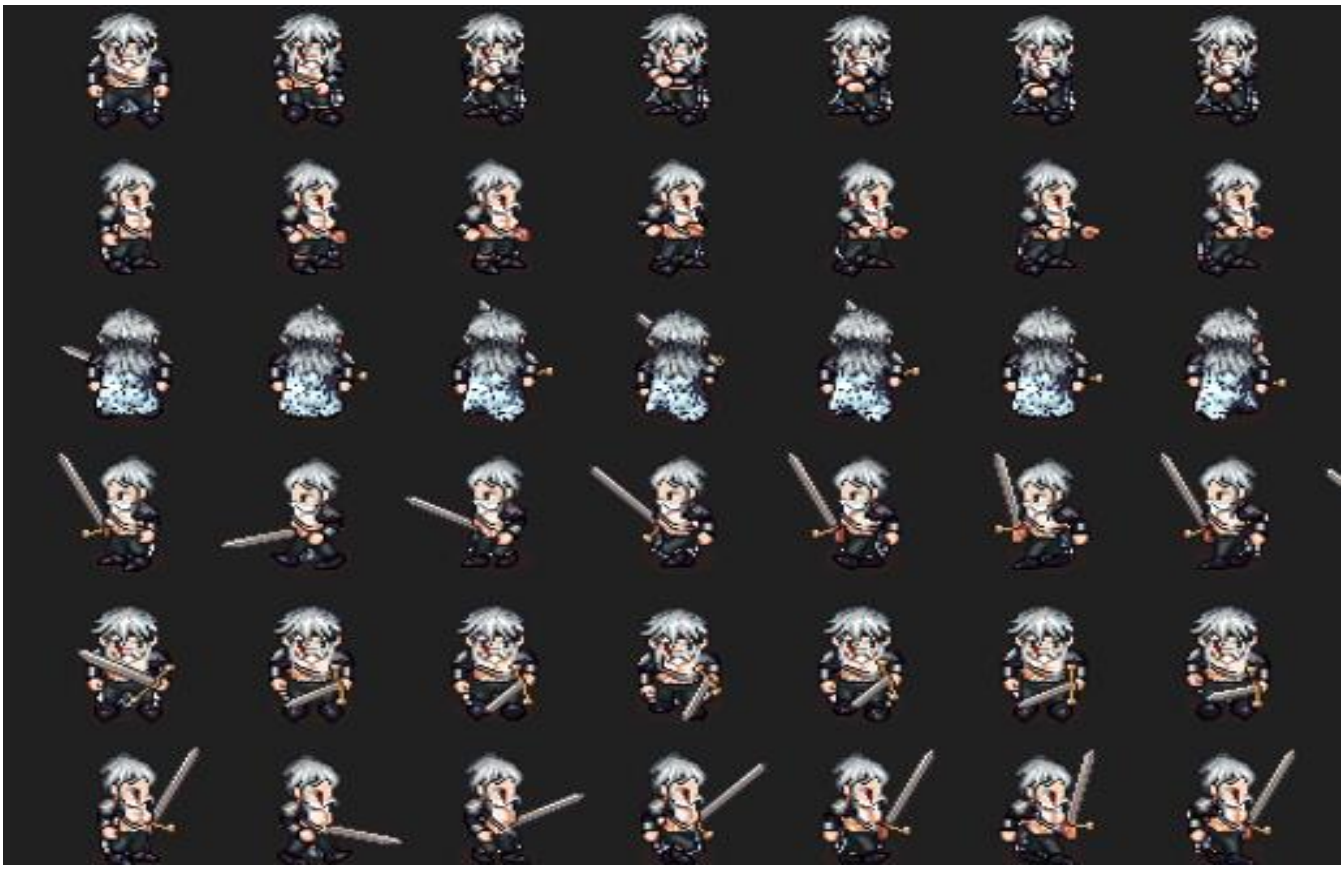
Jocul are la baza sa 3 nivele, în care protagonistul parcurge drumul din sat spre castelul lordului ce i-a executat pe ceilalți vânători. Pe parcursul nivelelor, jucatorul se va deplasa cu ajutorul săgeților, iar în momentul în care acesta întâmpină inamici, se va folosi de tasta SPACE pentru a ataca și va putea bloca atacurile inamicilor cu ajutorul unei taste, spre exemplu tasta Q.

Jucătorul va avea poziționat în colțul din stânga o bara ce va reprezenta HP-ul caracterului. În momentul în care acesta este atacat, dacă nu va reuși să se ferească în timp util sau să blocheze atacul, atunci va pierde un anumit număr de puncte de viață, în funcție de

nivelul în care se află. Dacă jucătorul moare în urma unui asemenea nivel, atunci el va relua de la început jocul. Scopul final este de a trece toate nivelele pentru a-l putea confrunta pe lord.

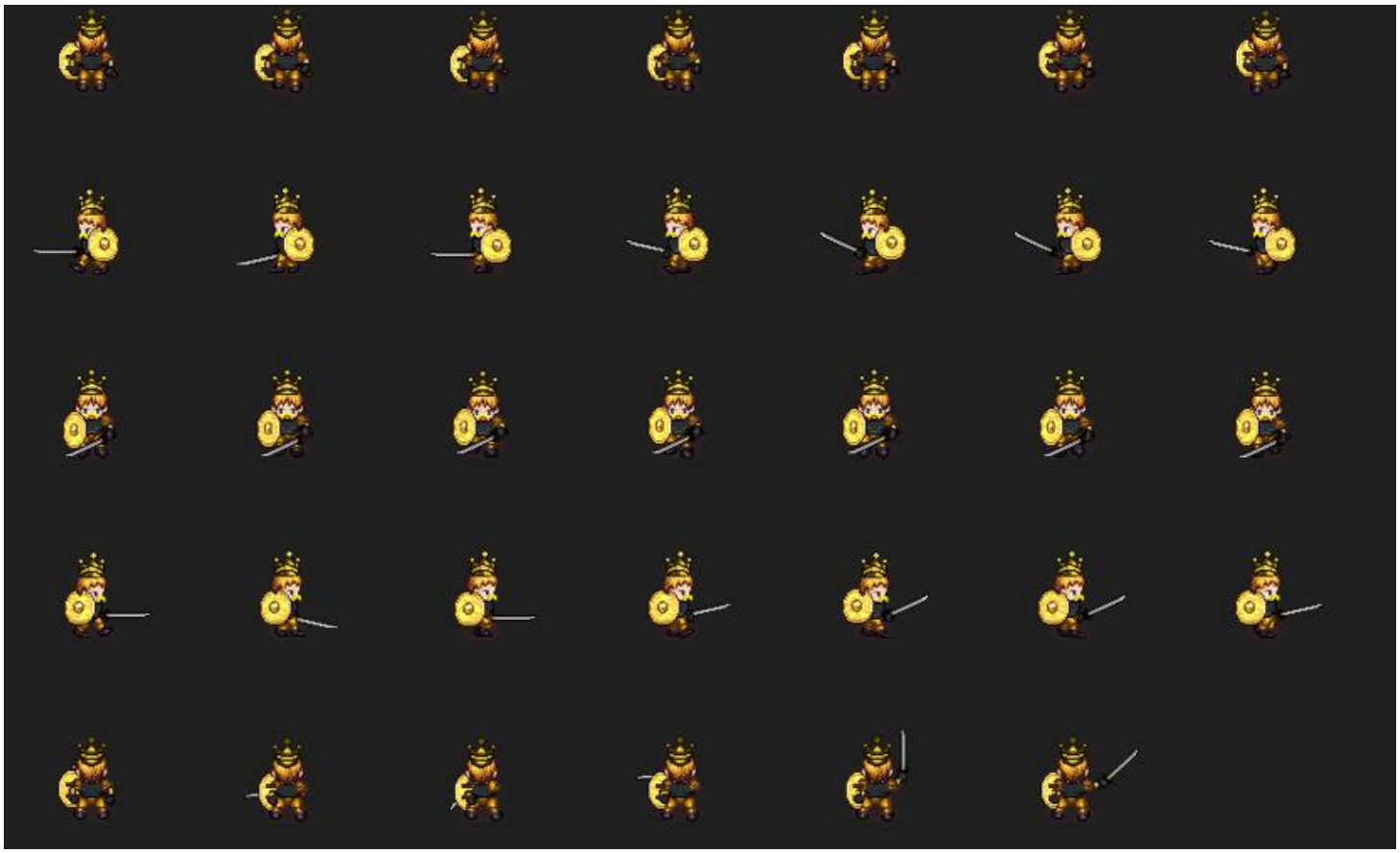
3.Proiectarea conținutului și descriere detaliată:

1) Roland: un fost vânător de monștrii cu păr alb si fața plină de cicatrice dobândite in cei o suta de ani de activitate. Antrenamentele îndelungate, împreună cu stilul greu de viață l-au transformat într-un luptător genial si într-un asasin fără milă. Acesta nu era un ucigaș ca oricare altul, țintele lui au fost monștrii, până nu a mai rămas niciunul. Însă acum are o nouă țintă. Un lord ranchinos ce i-a executat pe aproape toți vânători aflați în viață, cu excepția lui.



2) Lord Beric Vane (Lordul din Rivia): În zilele sale de glorie, nobila casă Vane era una dintre cele mai nobile din întreaga lume cunoscută. Lordul Beric a crescut înconjurat de ura

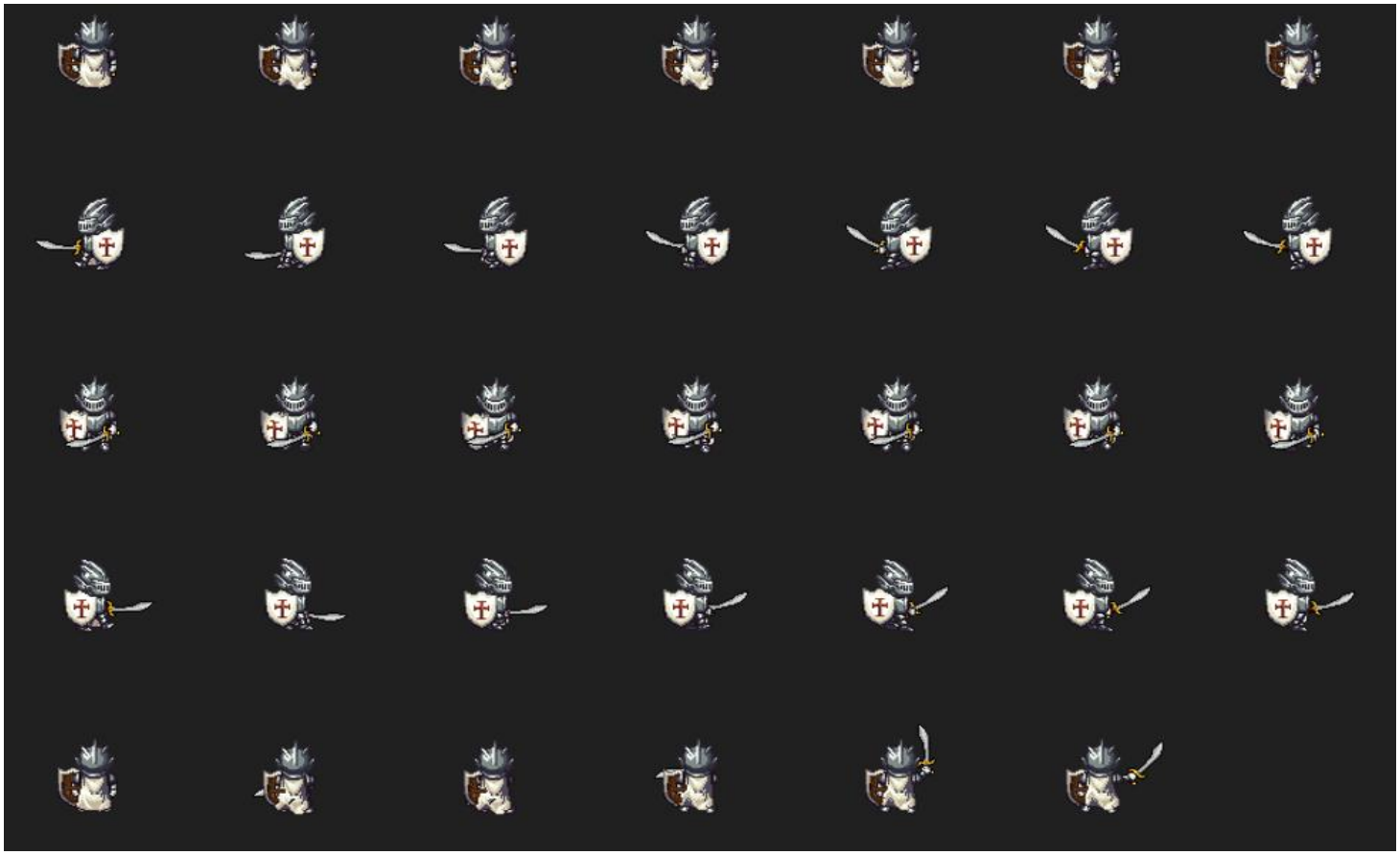
tatălui său pentru vântători, deoarece, după părerea sa, orice lucru rău care se abătea asupra casei sale, era cauzat de acești ciudați, după cum îi numea el.



3) Soldați/Cavaleri: razboinici aflați în slujba lordului Beric. Aceștia sunt protejați de armuri și au vaste cunoștințe despre mânuirea săbiilor.

4)Garda personală a Lordului: Aceasta este formată dintr-un numar mic de cavaleri căliți prin nenumărate războaie, fiind considerați cei mai buni din întreaga zonă. Datorită mantiilor albe, aceștia mai sunt numiți de către săteni și “White Cloaks” sau “White Swords”. Ei depun un jurământ sacru în fața zeilor vechi, cât si în fața celor noi și sunt pregatiți sa își piardă viața pentru lordul în a căreia slujba se află. (În acest caz, lordul Beric)

Aceștia au o armură mai puternica decât cavalerii de rând și reprezinta un pericol mult mai mare pentru jucător.



4. Proiectarea nivelurilor și descriere detaliată:

Jucătorul trebuie să parcurga 3 nivele pentru a-și putea razbuna prietenii.

În cadrul primului nivel acesta află ce s-a întâmplat cu ceilalți vânători, si va fi abordat de un numar mic de soldați ce au aflat de sosirea sa în zona. După înfrângerea lor, jucătorul o va lua pe drum, prin pădure unde va ajunge la nivelul 2.

În al doilea nivel, acesta va interacționa cu o tabără de inamici care se afla pe drumul său, jucătorul încercând să afle mai multe detalii legate de acțiunile lordului.

În al treila nivel, acesta îl va confrunta pe lordul vinovat de uciderea celorlalți vânatori si pe cavalerii ce au jurat să îl protejeze pe acesta.

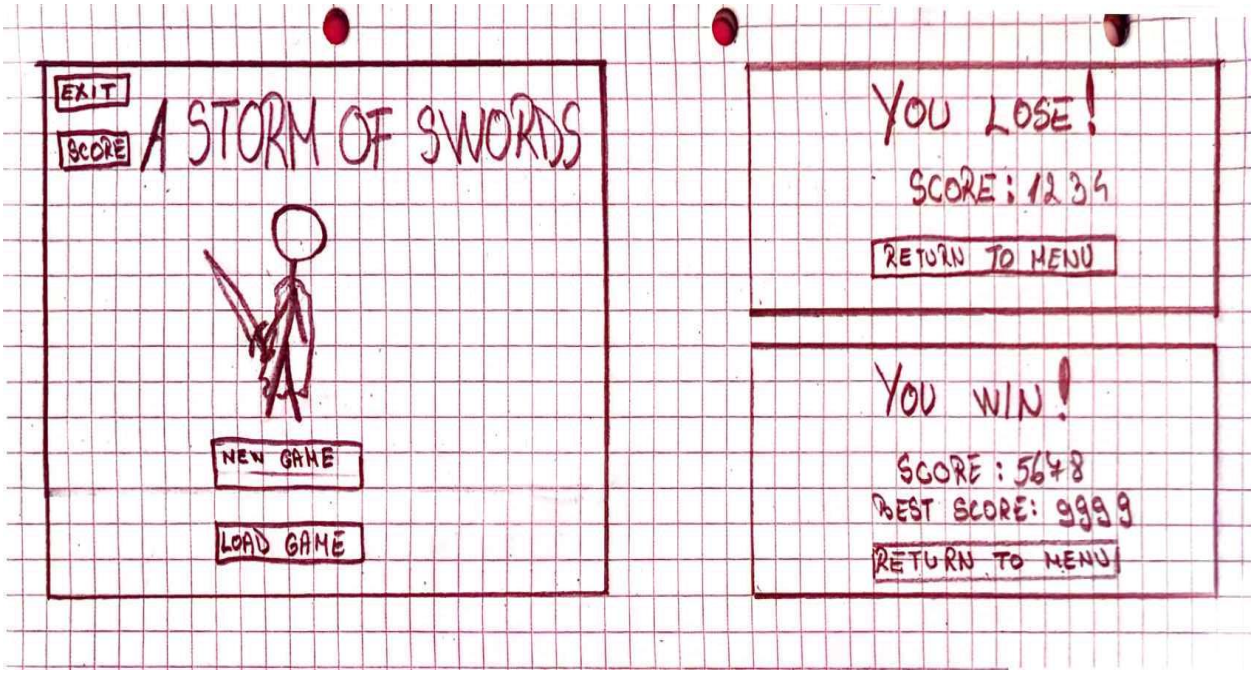
EXEMPLU HARTA NIVEL 1

În cadrul fiecărui nivel, hărțile vor fi construite din dale (tile), iar caracterul principal va fi poziționat în mijlocul ecranului, astfel, implementându-se concepul de cameră ca mod de afișare și control asupra perspectivei jocului. Fiecare hartă va fi limitată la un moment dat de anumite obstacole, spre exemplu: râuri, munți, copaci, etc.

5.Proiectarea interfeței cu utilizatorul și descriere detaliată:

Interfața jocului este una simplistă. După pornirea jocului, se va deschide o fereastră ce va conține imaginea din al doilea slide. Meniul principal va avea un buton pentru a incepe un nou joc, unul pentru a relua ultima salvare si un buton pentru a inchide jocul.

Refresh Rate-ul va fi de aproximativ 30 de cadre pe secunda pentru o experiență cât mai buna.

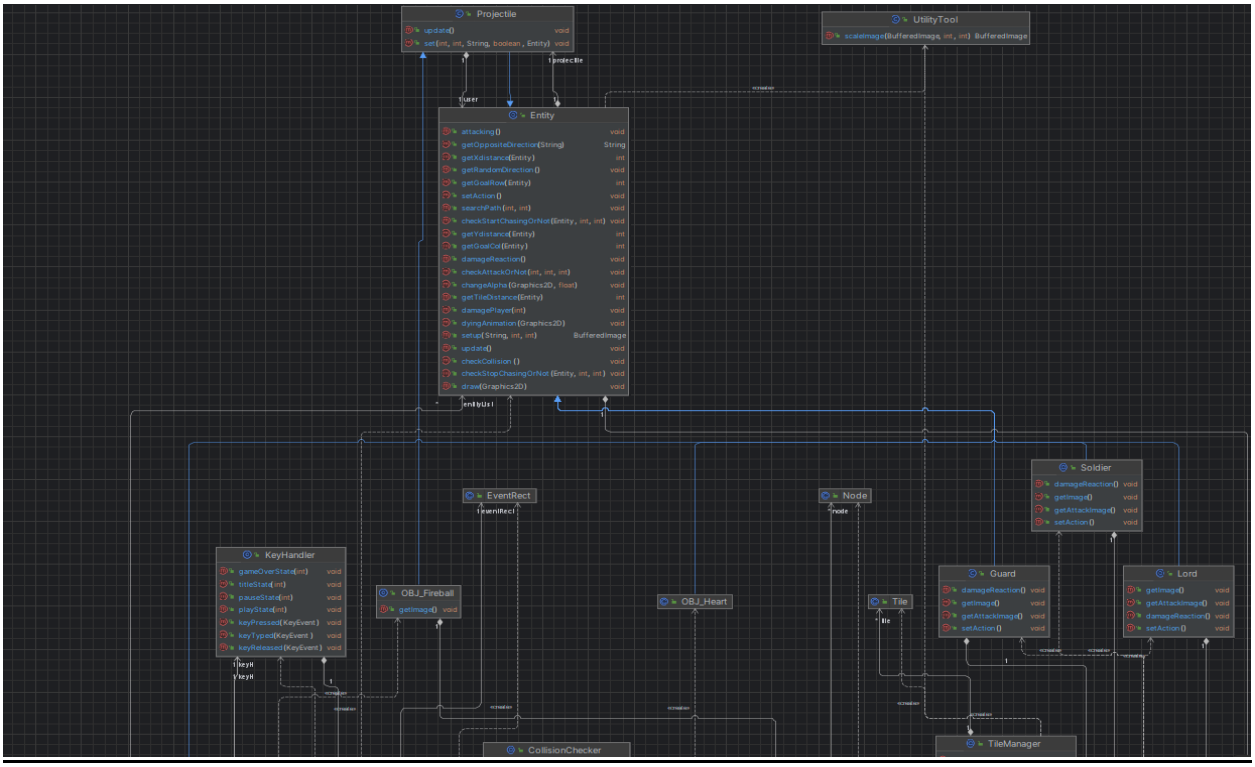


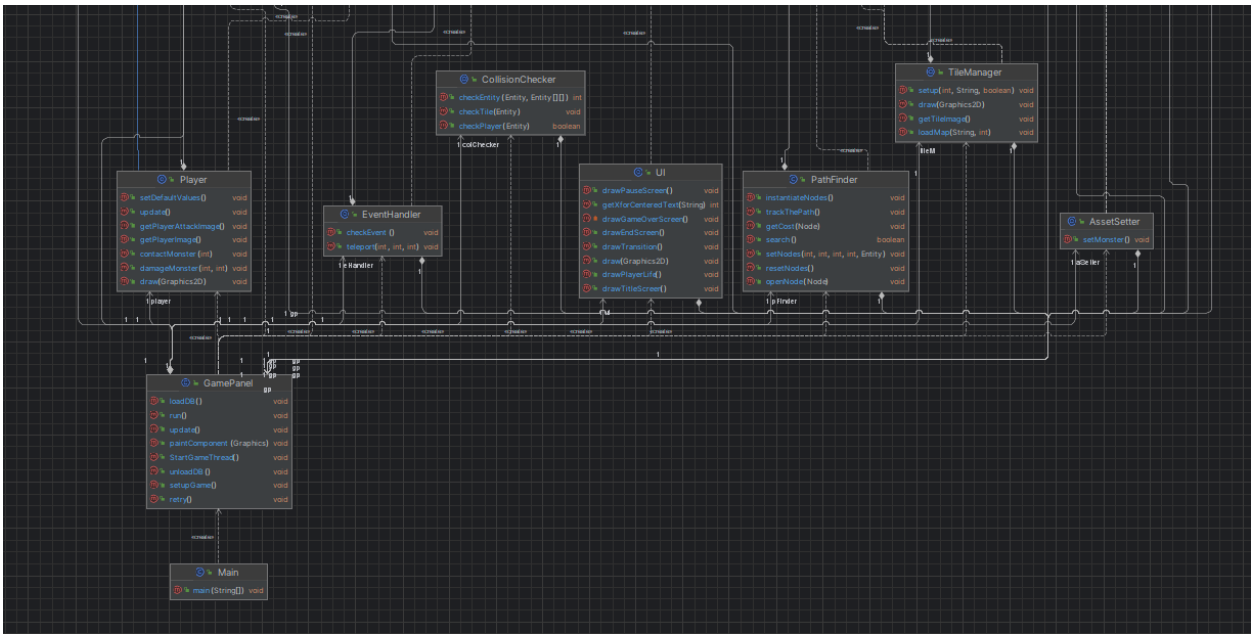
În cazul în care jocul va conține NPC-uri, acesta va putea interacționa cu ele, aparând pe ecran un dialog pentru a ajuta jucatorul în întelegerea poveștii.

O altă metodă poate fi inserarea unui buton în meniul jocului ce va deschide o fereastră unde se va găsi povestea jocului, sau o varianta mai scurtă a acesteia.

6.Diagramă proiect

În figura de mai jos se poate observa diagrama de clase UML (sunt vizibile doar metodele claselor pentru a nu aglomera diagrama). Fiecare clasă are rolul ei în cadrul proiectului și la rândul său instanțiază/moștenesc alte clase.


Projectile
update()
set(int, int, String, boolean, Entity)
UtilityTool
scaleImage(BufferedImage, int, int)
Entity
attacking()
getOppositeDirection(String)
getXdistance(Entity)
getRandomDirection()
getGoalRow(Entity)
setAction()
searchPath(int, int)
checkStartChasingOrNot(Entity, int, int)
getYdistance(Entity)
getGoalCol(Entity)
damageReaction()
checkAttackOrNot(int, int, int)
changeAlpha(Graphics2D, float)
getTileDistance(Entity)
damagePlayer(int)
dyingAnimation(Graphics2D)
setup(String, int, int)
update()
checkCollision()
checkStopChasingOrNot(Entity, int, int)
draw(Graphics2D)
EventRect
Node
Soldier
damageReaction()
getImage()
getAttackImage()
setAction()
KeyHandler
gameOverState(int)
titleState(int)
pauseState(int)
playState(int)
keyPressed(KeyEvent)
keyTyped(KeyEvent)
keyReleased(KeyEvent)
OBJ_Fireball
getImage()
OBJ_Heart
Tile
Guard
damageReaction()
getImage()
getAttackImage()
setAction()
Lord
getImage()
getAttackImage()
damageReaction()
setAction()
CollisionChecker
TileManager


CollisionChecker
checkEntity(Entity, Entity[])
checkTile(Entity)
checkPlayer(Entity)
TileManager
setup(int, String, boolean)
draw(Graphics2D)
getTileImage()
loadMap(String, int)
Player
setDefaultValues()
update()
getPlayerAttackImage()
getPlayerImage()
contactMonster(int)
damageMonster(int, int)
draw(Graphics2D)
EventHandler
checkEvent()
teleport(int, int, int)
UI
drawPauseScreen()
getXforCenteredText(String)
drawGameOverScreen()
drawEndScreen()
drawTransition()
draw(Graphics2D)
drawPlayerLife()
drawTitleScreen()
PathFinder
instantiateNodes()
trackThePath()
getCost(Node)
search()
setNodes(int, int, int, int, Entity)
resetNodes()
openNode(Node)
AssetSetter
setMonster()
GamePanel
loadDB()
run()
update()
paintComponent(Graphics)
StartGameThread()
unloadDB()
setupGame()
retry()
Main
main(String[])

7.Descriere clase proiect:

Main: se ocupă cu configurarea și inițializarea interfeței grafie. Este punctul de intrare în aplicație

UI: responsabilă pentru desenarea interfeței grafice a utilizatorului în cadrul jocului. Aceasta include elemente precum ecranul de meniu, cel de pauză și cel de finalizare a jocului.

draw() – metodă responsabilă cu desenarea interfeței jocului și a ecranului corespunzător stării curente a jocului;

drawGameOverScreen() – metodă responsabilă cu desenarea ecranului de Game over

drawPlayerLife() – metodă ce desenează HP-ul jucătorului

drawTitleScreen() – metodă ce desenează meniul jocului

drawTransition() – metodă ce desenează efectul de tranziție dintre nivele

drawPauseScreen() – metodă ce desenează ecranul de pauză

drawEndScreen() – metodă ce desenează ecranul de final

getXforCenteredText() – metodă ce returnează coordonata X pentru a centra un text

UtilityTool: conține metoda scaleImage() – utilizată pentru a redimensiona imaginile, pastrând proprietățile imaginii

KeyHandler: implementează KeyListener pentru a gestiona evenimentele legate de taste

keyPressed() – responsabilă pentru gestionarea acțiunilor declanșate atunci când o tastă este apăsată în funcție de starea de joc în care ne aflăm

keyReleased() – responsabilă pentru gestionarea acțiunilor declanșate atunci când o tastă este eliberată

titleState() – folosită pentru a putea parcurge meniul

playState() – permite jucătorului să controleze personajul

pauseState() – permite punerea jocului pe pauză, cât și revenirea acestuia în playState

gameOverState() – folosită pentru a parcurge ecranul de GameOver

GamePanel: componenta principală a jocului responsabilă pentru gestionarea și actualizarea tuturor entităților, precum și desenarea lor pe ecran.

Implementează interfața Runnable pentru a permite rularea într-un fir de execuție

LoadDB() și unloadDB() – metode ce gestionează salvarea și încărcarea datelor dintr-o bază de date. Vor fi salvate nivelul, timpul jucat și viața jucătorului.

setupGame() – setează inamicii pe hartă și starea jocului la titleState

retry() – resetează caracteristicile jucătorului, inamicii, timpul de joc și harta curentă

StartGameThread() – creează și pornește un nou fir de execuție

run() – metoda principală de rulare a jocului. Actualizează starea jocului și re-desenarea la fiecare ciclu, respectând un interval de timp specific pentru a menține un FPS constant

update() – actualizează starea entităților din joc, daca acesta se află în playState

paintComponent() – desenarea elementelor grafice pe ecran, precum tile-uri, jucător, inamici, proiectile

EventHandler: responsabilă pentru gestionarea evenimentelor, precum schimbarea hărții.

checkEvent() – verificăm dacă jucătorul se află pe o anumită poziție și are o anumită direcție pentru a putea declansa evenimentul, precum schimbarea hărții

teleport() – poziția jucătorului va fi actualizată pe următoarea hartă și va actualiza viața acestuia înapoi la 100%

CollisionChecker: verifică diferite tipuri de coliziuni, cum ar fi cel cu tile-urile, cu jucătorul sau cu inamicii

checkTile() – verificarea coliziunii entităților cu tile-urile

checkEntity() – verificarea coliziunii dintre entități

checkPlayer() – verificarea coliziunii entității date cu jucătorul

AssetSetter: conține metoda setMonster() – pentru fiecare tip de inamic, se crează o instanță noua a acestuia și se setează coordonatele acestora

Entity:

setAction() și damageReaction() – destinate a fi suprascrise de către clasele derivate

checkCollision() – verifică coliziunile cu tile-urile, cu jucătorul și alte entități

update() – actualizează starea entităților și gestionează mișcarea, coliziunile, atacurile și animațiile

attacking() – responsabilă pentru gestionarea logicii de atac a unei entități din joc

checkAttackOrNot() – verifică dacă entitatea ar trebui să atace jucătorul, verificând dacă se află în raza de acțiune a acestuia

damagePlayer() – se ocupă cu aplicarea daunelor către jucător în funcție de tipul de inamic care îl atacă și dacă jucătorul blochează atacul sau nu

getOppositeDirection() – returnează direcția opusă jucătorului

draw() – selectează sprite-urile corecte și le desenează în funcție de starea entității și direcția de deplasare, desenarea barei de viață pentru inamici, efectul de invincibilitate, animația de deces

dyingAnimation() – crează un effect de blink înainte ca entitatea să dispară de pe ecran

changeAlpha() – modifică transparența elementelor desenate

setup() – încarcă o imagine din fișierul de resurse și o redimensionează

searchPath() – calculează și actualizează calea către un obiectiv

checkStopChasingOrNot() – verifică daca entitatea ar trebui să oprească urmărirea jucătorului

checkStartChasingOrNot() – verifică dacă entitatea ar trebui sa pornească în urmărirea jucătorului

getRandomDirection() – setează o direcție random pentru următoarele 120 cadre

Player: constructorul acestei clase inițializează jucătorul cu dimensiuni, zone solide și de atac, setările implicite și imaginile pentru mișcare și atac

setDefaultValues() – setează valorile implicite pentru jucător

getPlayerImage() – încarcă și setează imaginile jucătorului pentru diferite direcții

getPlayerAttackImage() – încarcă și setează imaginile jucătorului pentru atac

update() – actualizează starea jucătorului

contactMonster() – dacă jucătorul se lovește de un inamic, acesta primește damage

damageMonster() – se ocupă cu aplicarea de damage inamicilor

draw() – desenarea sprite-urilor jucătorului pe ecran, în funcție de starea și direcția acestuia

Projectile: constructorul acestei clase inițializează caracteristicile unui proiectil

update() – dacă proiectilul este tras de jucător se verifică coliziunea cu monștrii, iar dacă acesta ajunge la inamic, se aplică daune iar proiectilul dispare

OBJ_Fireball: constructorul clasei inițializează caracteristicile proiectilului de acest tip.

getImage() – încarcă și seteazp imaginile proiectilului pentru fiecare direcție

OBJ_Heart: încarcă imaginile pentru a putea afișa HP-ul jucătorului

Guard, Soldier, Lord: constructorul acestor clase inițializează variabilele de instanță ale clasei.

getImage() – încarcă și setează imaginile associate cu mișcarea inamicului în diferite direcții

getAttackImage() - încarcă și setează imaginile associate cu atacul inamicului în diferite direcții

setAction() – metodă responsabilă pentru definirea acțiunilor inamicului

damageReaction() – în momentul în care primește daune, inamicul va începe să urmărească jucătorul

PathFinder: implementează logica algoritmului A* pentru a găsi calea optima de la un punct de start la un punct de destinație

instantiateNodes() – inițializează matricea de noduri

resetNodes() – resetează starea nodurilor și a altor setări

setNodes() – setează nodurile de start și destinație și matricea nodurilor solide

getCost() – calculează costurile G, H și F

search() – utilizată pentru căutarea căii folosind algoritmul A*. Aceasta explorează nodurile vecine pentru a găsi cel mai bun nod până când nodul destinație este atins sau este atins un anumit număr de pași

openNode() – metod- pentru deschiderea unui nod

trackThePath() - metodă pentru urmărirea căii de la nodul destinație înapoi la nodul de start și construirea listei de noduri care formează calea

TileManager: inițializare pentru array-ul tile și mapTileNum. Încărcăm tile-urile pentru toate cele 3 hărți.

getTileImage() – încărcăm fiecare imagine pentru toate tipurile de tile-uri existente și le setăm coliziuniea tile-urilor ce necesită acest lucru

setup() – încărcăm imaginile și le redimensionăm

loadMap() – se încearcă deschiderea și citirea fișierului ce conține matrice care conține ID-urile pe care le au tile-urile

draw() – desenează toate tile-urile vizibile pe ecran

8.Șabloane de proiectare:

1. Singleton Pattern

Utilizat în clasa GamePanel. Acesta asigură că există o singură instanță a GamePanel în cadrul aplicației și oferă un punct global de acces la acesta.

2. State Pattern:

Implementat în clasa UI pentru a gestiona diferitele stări ale jocului (titileState, playState, pauseState, gameOverState, transitionState, gameFinished). Fiecare stare are propriul său comportament specific pentru desenarea interfeței utilizatorului.

3. Factory Method:

Utilizat în UI pentru crearea instanțelor de inimi (heart_full, heart_half, heart_blank). Clasa AssetSetter folosește acest șablon pentru a crea și inițializa diferite tipuri de inamici.

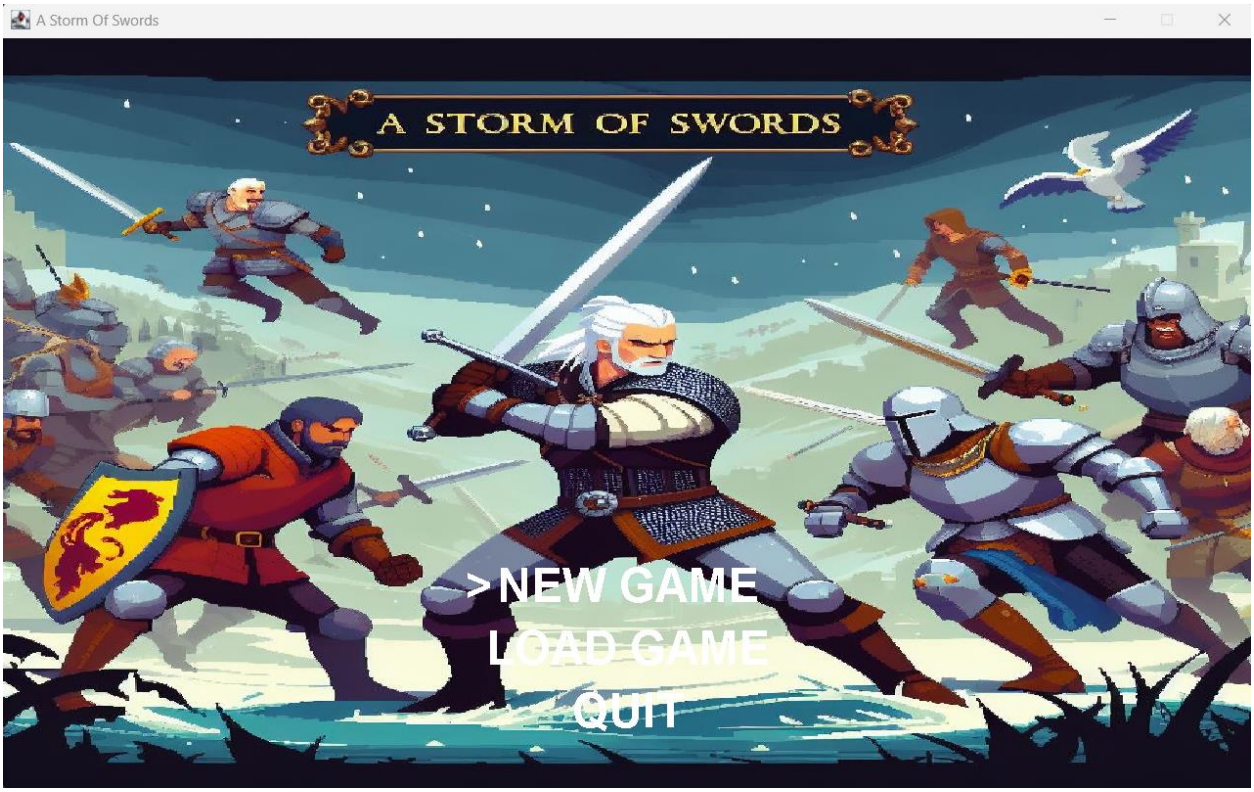
4. Observer:

Vizibil în relația dintre UI și GamePanel. În UI, metoda draw() este folosită pentru a desena diferitele ecrane ale jocului, precum meniul, ecranul de joc, de pauză. Această metodă este apelată din metoda paintComponent() a clasei GamePanel care este responsabila pentru desenarea componenteloro jocului pe ecran. Astfel, UI este notificat și actualizat automat de fiecare data când GamePanel trebuie sa fie redesenat.

9.Diferite ipostaze din joc:

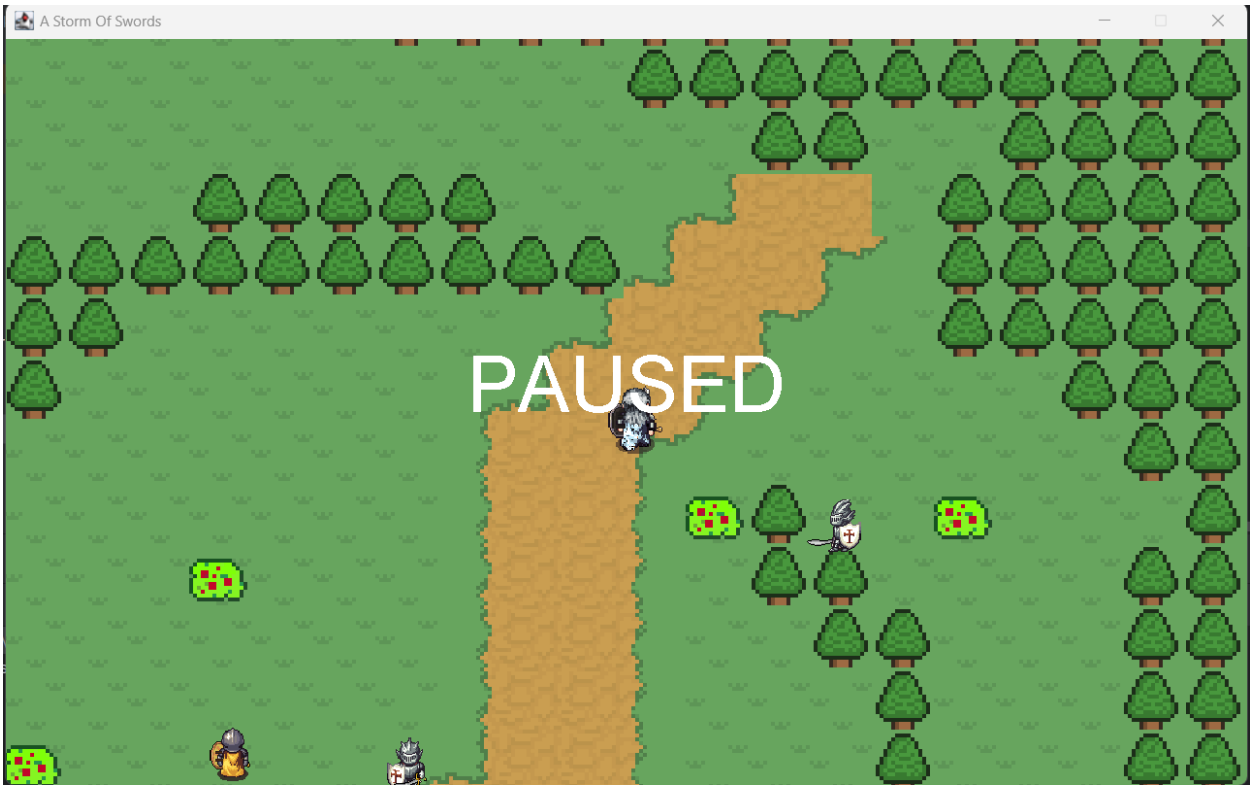
1. TitleState:

Reprezintă meniul jocului, ce poate fi parcurs cu ajutorul săgeților.



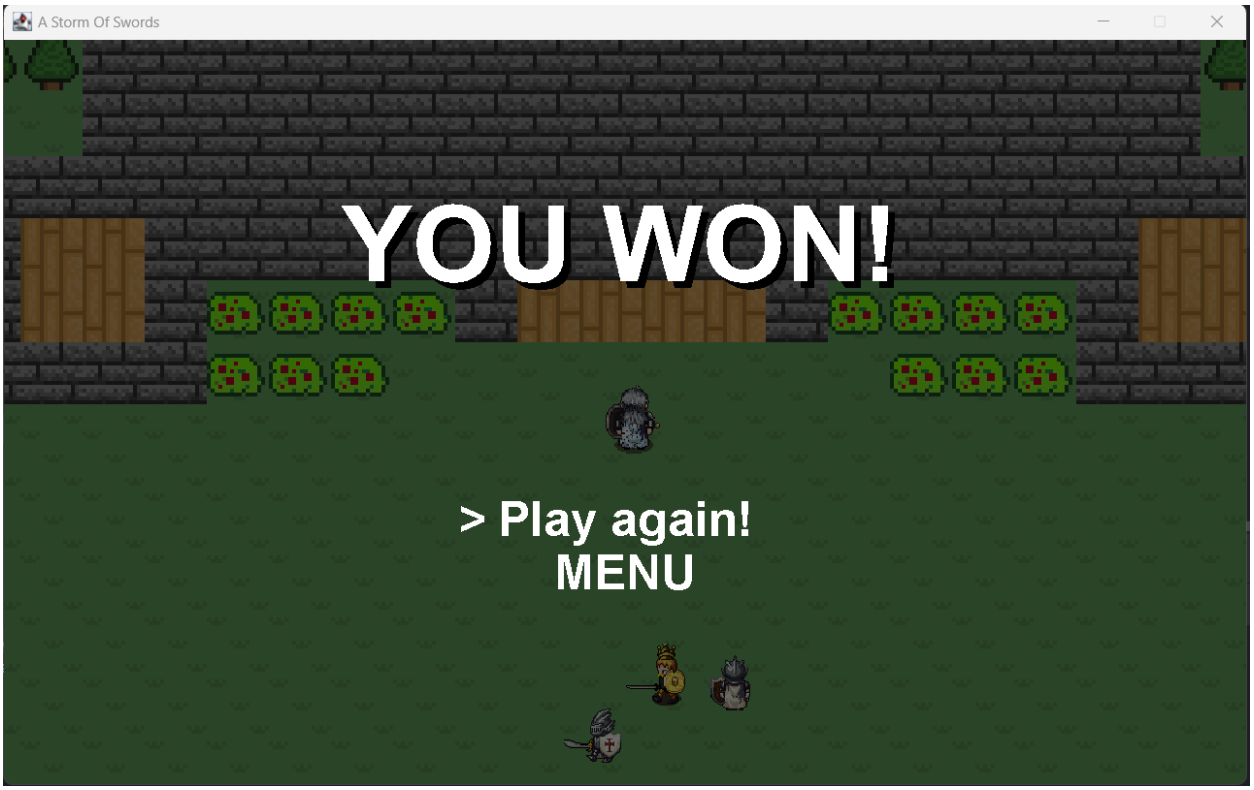
2. PlayState: jucătorul se poate deplasa pe hartă și interacționa cu inamicii

3. PauseState: jucătorul poate reveni în playState apăsând tasta P.



4. gameOverState: jucătorul se poate întoarce în meniu, sau poate reveni la nivelul 1.

5. gameFinished: jucătorul se poate întoarce în meniu, sau la începutul jocului.



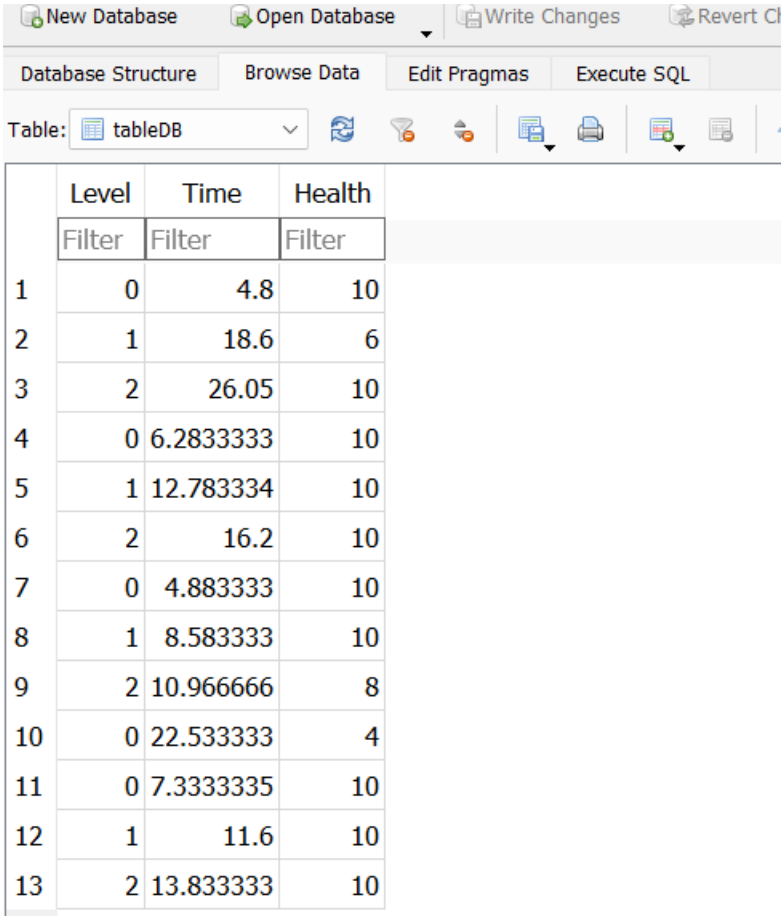
10.Baza de date:

În acest joc, baza de date este folosită pentru a înregistra informațiile despre toate câștigurile jucătorului. Pentru fiecare instanță sunt stocate:

-nivelul câștigat

-viața cu care a terminat jucătorul nivelul

-momentul de timp în care jucătorul a terminat nivelul



	Level	Time	Health
	Filter	Filter	Filter
1	0	4.8	10
2	1	18.6	6
3	2	26.05	10
4	0	6.2833333	10
5	1	12.783334	10
6	2	16.2	10
7	0	4.883333	10
8	1	8.583333	10
9	2	10.966666	8
10	0	22.533333	4
11	0	7.3333335	10
12	1	11.6	10
13	2	13.833333	10

11.Bibliografie:

Image Generator:

https://www.bing.com/images/create?FORM=GENILP

Sprites:

https://sanderfrenken.github.io/Universal-LPC-Spritesheet-Character-Generator/#?body=Body_color_light&head=Human_male_light