Fate of the Human Kingdom

Georgiță Adrian

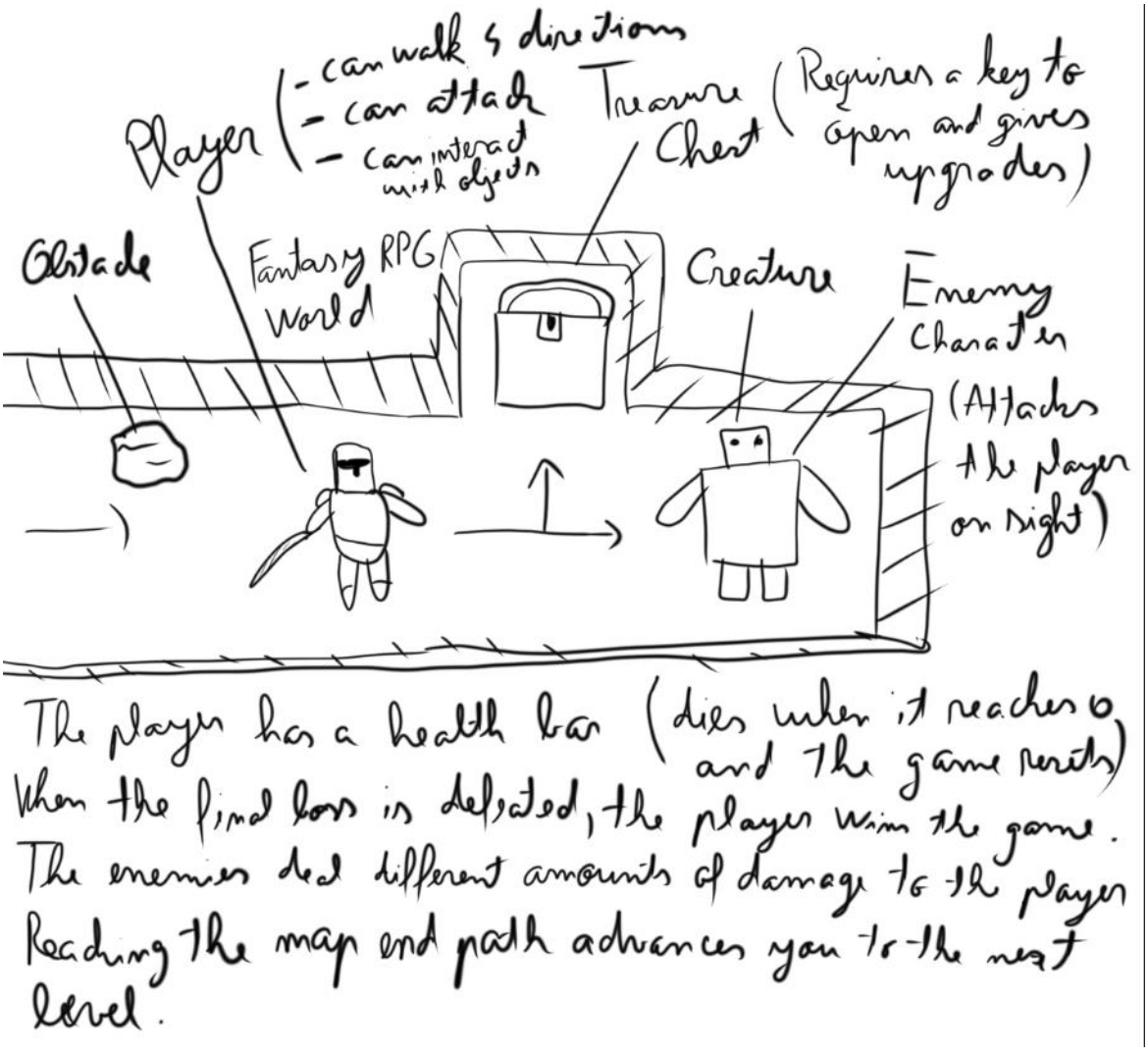
1211A

Contents

Proiectarea contextului

Poveste joc

Regatul oamenilor trăia liniștit și a prosperat într-o pace care a durat sute de ani. Cavalerii Ordinului Phoenix au protejat și menținut ordinea și pacea în regat încă de la înființarea ordinului. Intr-o misiune în care aceștia au fost trimiși să-l captureze pe fostul cavaler Lodrax care era suspectat de trădare și rebeliune, au fost nimiciți de către supușii lui Lodrax, noul rege proclamat al creaturilor, permițându-i acestuia să invadeze și să ardă din temelii regatul oamenilor. Marele maestru al Ordinului Phoenix se întorcea dintr-o misiune diplomatică atunci când a vazut ruinele regatului. Din dorința de a se răzbuna și de a pune capăt tiraniei lui Lodrax, acesta a pornit într-o misiune pentru a-l ucide și pentru a salva soarta omenirii.



Proiectarea sistemului

Controale

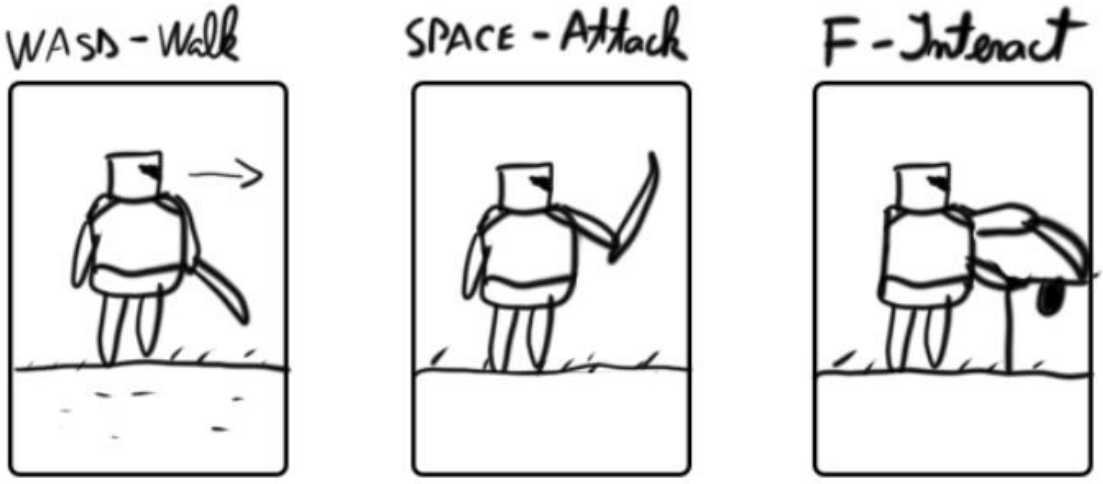
Jucătorul se poate deplasa pe hartă cu ajutorul tastelor W,A,S,D sau cu săgețiile de la tastatură (atunci când setarea este activa). Atunci când pe direcția dorită se află un obstacol, jucătorul nu se deplasează și rămâne pe poziția curentă până se allege altă direcție posibilă.

Camera este actualizată automat odată cu deplasarea caracterului, fiind focalizată pe acesta.

Prin apăsarea tastei Space, jucătorul poate inița acțiunea de atac, prin care cauzează daune inamicilor din apropriere, dacă aceștia sunt prezenți.

Prin apăsarea tastei F, jucătorul interacționează cu diferite obiecte cu care poate interacționa, precum cufere și chei.

Fiecare acțiune a caracterului este reprezentată printr-o serie de animații.



Gameplay

Jucătorul trebuie să se deplaseze pe hartă prin locurile permise precum cărări din pământ, iarbă sau din piatră. Acesta trebuie sa ocolească obstacole dacă acestea se află în calea lui.

Pentru a se putea deplasa la finalul nivelului și a trece la nivelul următor, jucătorul trebuie să atace inamicii și să îi ucidă pentru a putea trece de aceștia.

Atunci când jucătorul este atacat de către un inamic, nivelul acestuia de viață scade în funcție de dauna pe care o provoacă tipul de inamic respectiv.

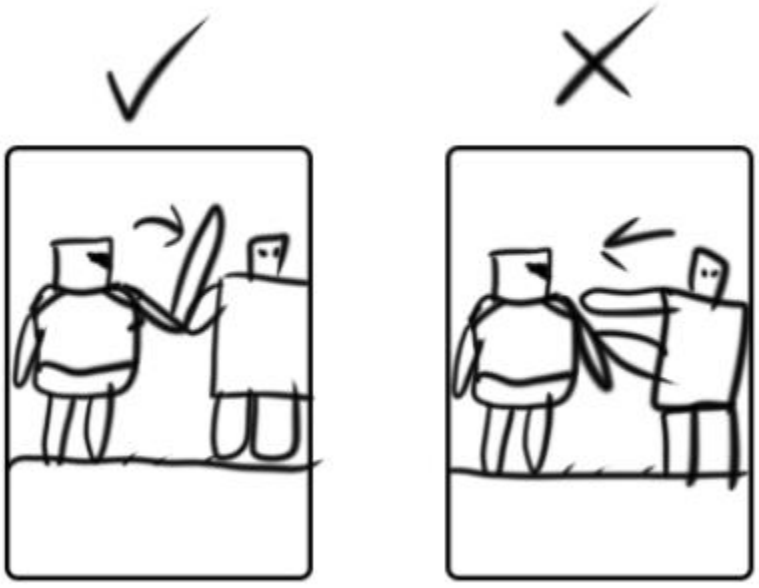
Jucătorul poate explora harta și să găsească diferite cufere ce îi oferă diferite îmbunătățiri atunci când sunt deschise. Aceste îmbunătățiri constau în daună provocată mai mare precum și un nivel mai mare de viață sau resetarea acestuia. Pentru a deschide un cufăr, este necesar să se găsească înainte o cheie aflată în același nivel.

Jucătorul trece la nivelul următor atunci când ajunge la finalul hărții respective, indiferent dacă acesta învinge inamicii sau nu învinge inamicii sau nu.

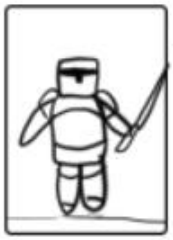
Pentru a câștiga jocul, jucătorul trebuie să învingă caracterul negativ principal.

Pentru fiecare inamic învins, acesta primește un anumit număr de puncte ce contribuie la scorul primit odată cu terminarea jocului.

Dacă jucătorul este învins de către inamici, acesta pierde jocul și revine la nivelul inițial.

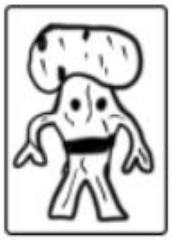


Proiectarea conținutului



Player - Grandmaster of the Phoenix Order
Purpose: Defeat the creatures and their king
Abilities: Mastered swordsmanship
Animations: Walk (4 dir), Attack (1), Idle, Interactions

Boss - Lodrax, King of the creatures
Purpose - Destroy the remaining humans
Abilities - Mage/Necromancer



Enemy - Ent / Golem / Wizard
Purpose: Help the creatures king and destroy the hero
Abilities: Melee Attack / Ranged Attad (Wizard)

Chest - Grants upgrade(s) upon interaction
Key - Required to open a chest

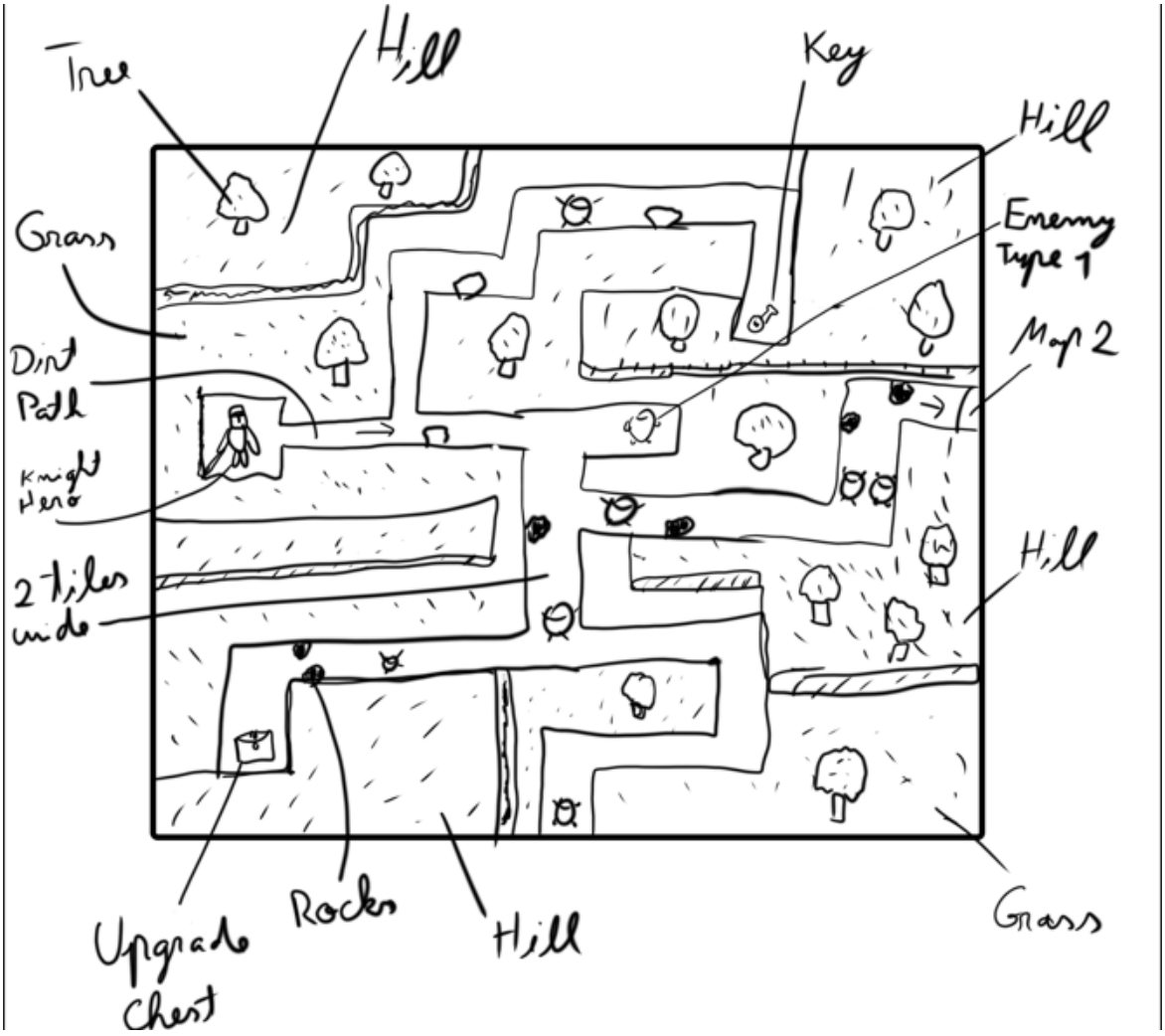
Rock - Obstacle that prevents the player from advancing
Comes in different variations (Walls, Altars, etc)

Proiectarea nivelurilor

Jocul este constituit din 3 niveluri, fiecare cu o hartă diferită și cu un nivel de dificultate mai greu odată cu progresarea acestora. Pentru a avansa la nivelul următor, jucătorul trebuie să ajungă la punctul de trecere spre zona următoare. Primele două niveluri au loc într-o pădure, iar cel de-al treilea are loc în catacombe. În fiecare nivel, apare un nou tip de inamic, care este mai puternic decât cei din nivelurile anterioare, precum și diferite obstacole pe care jucătorul trebuie să le ocolească pentru a avansa.

Nivelul 1

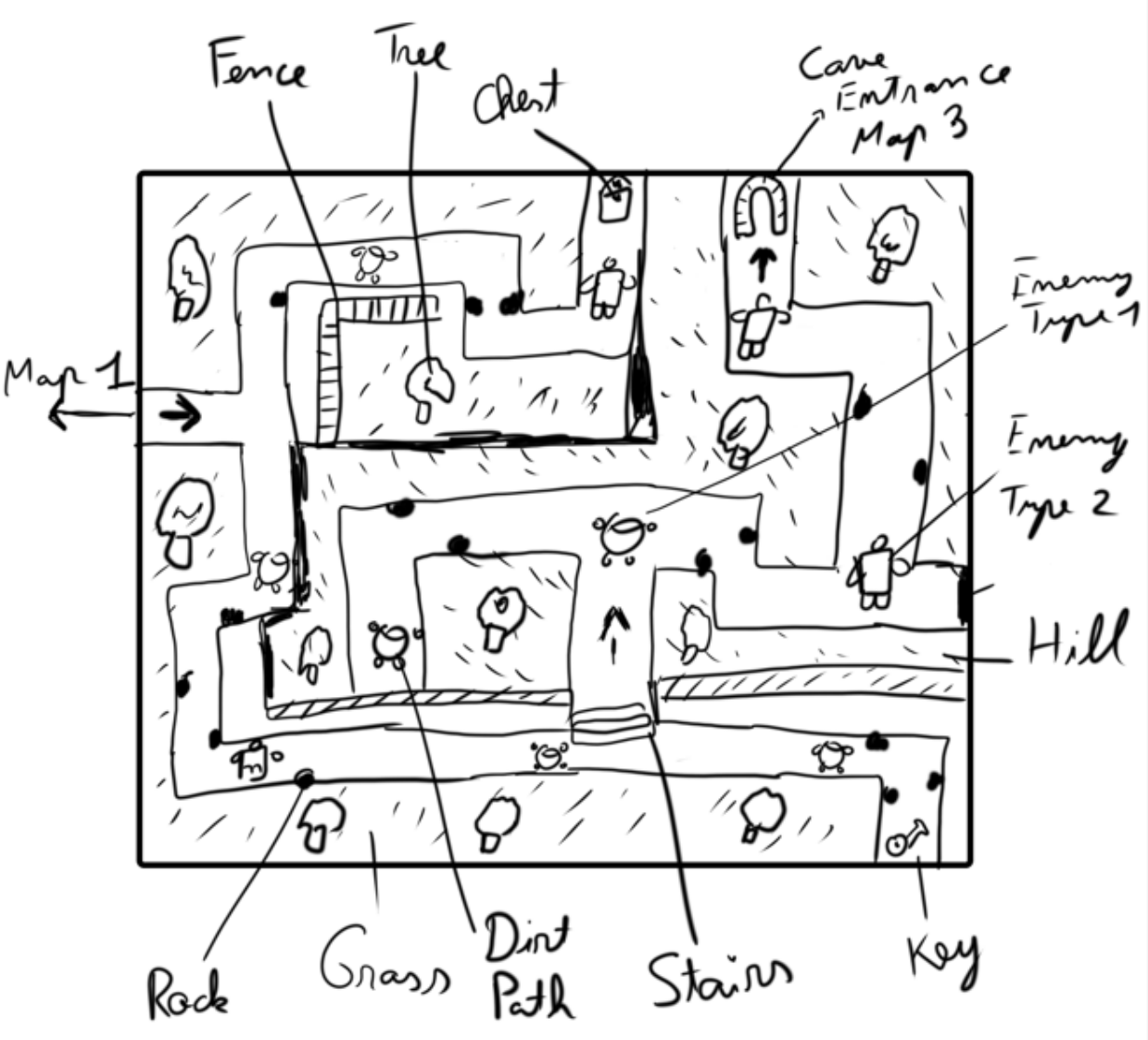
Poziția caracterelor și obiectelor în scena nivelului este următoarea:



În acest nivel, există doar un tip de inamic care este relativ ușor de învins.

Nivelul 2

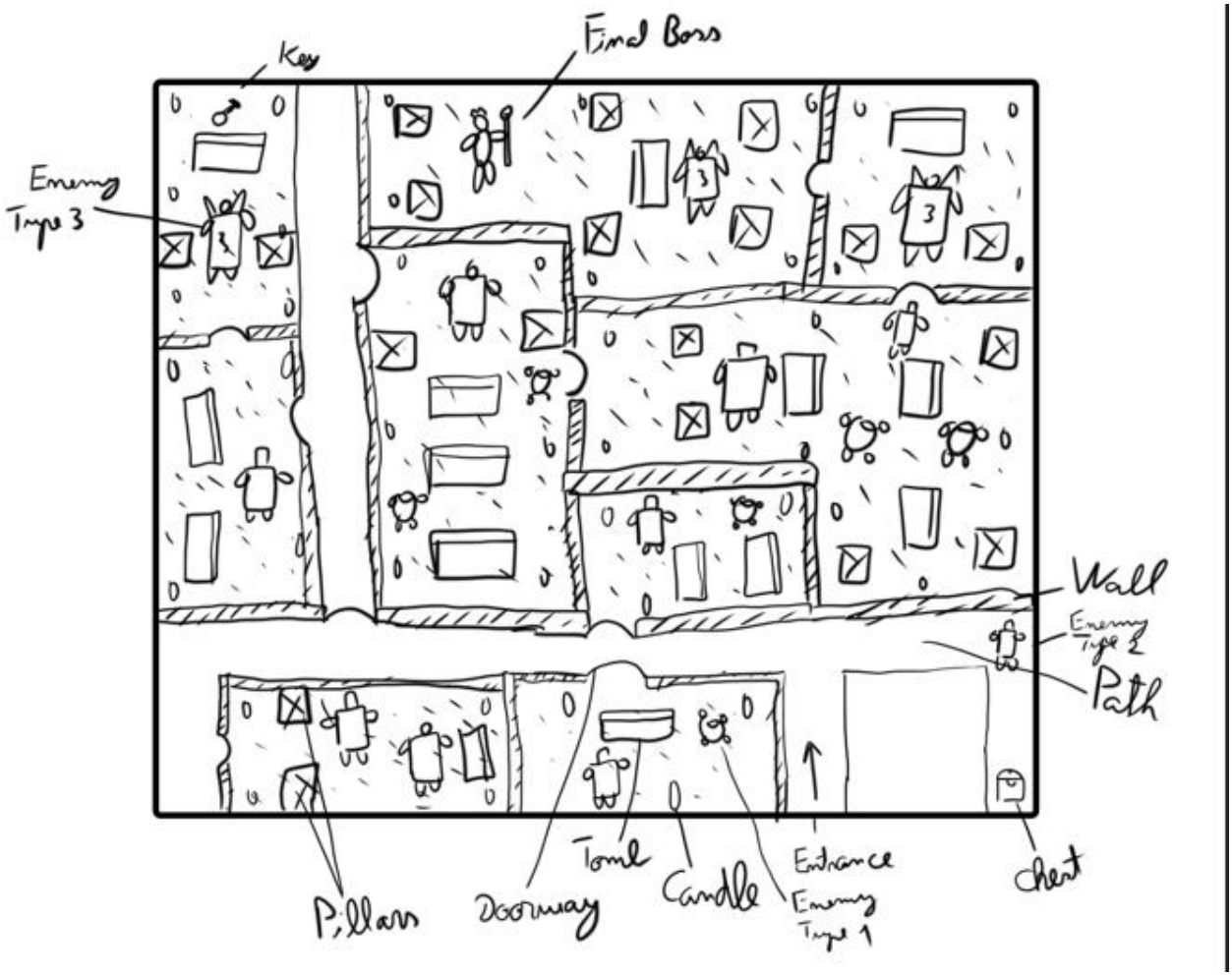
Poziția caracterelor și obiectelor în scena nivelului este următoarea:



Pe langă tipul de inamic din nivelul anterior, în acest nivel apare un alt tip de inamic care are mai multă viață și cauzează o daună mai mare nivelului de viață a jucătorului.

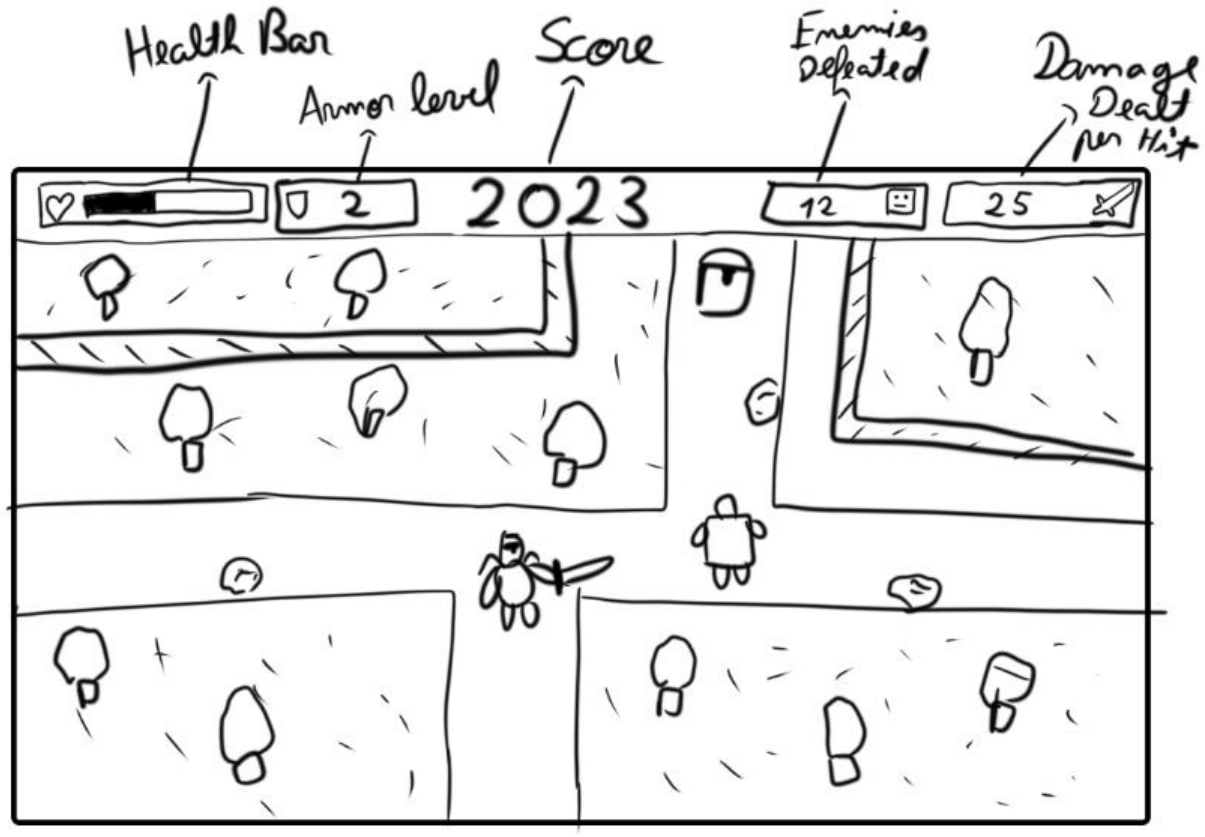
Nivelul 3

Poziția caracterelor și obiectelor în scena nivelului este următoarea:



Precum și în nivelul anterior, în acest nivel final, apare un nou tip de inamic mai puternic decât precedenții precum și personajul negativ principal pe care jucătorul trebuie să-l înfrângă pentru a câștiga jocul.

Proiectarea interfeței cu utilizatorul



Nivelul de viață a jucătorului va fi reprezentat printr-o bară a cărui lungime scade atunci când nivelul de viață scade.

Nivelul de armură reprezintă tipul de armură pe care îl are jucătorul. Atunci când acesta găsește un upgrade de armură într-un cufăr, valoarea se modifică.

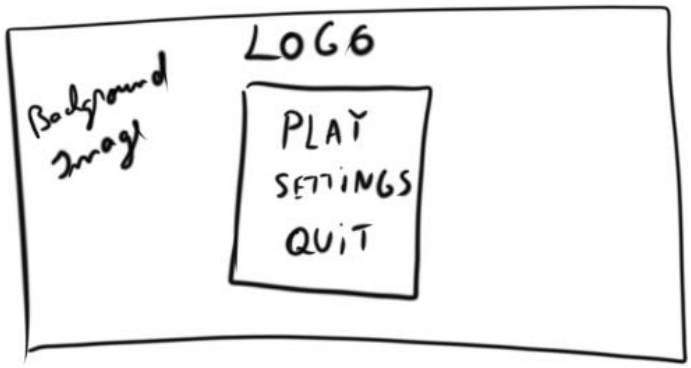
Scorul reprezintă suma tutoror acțiunilor ce îi aduc puncte jucătorului (ucide inamici, deschide cufere).

Numărul de inamici uciși este reprezentat în partea dreaptă sus și se modifică de fiecare dată când un inamic este ucis.

Dauna provocată de jucător este reprezentată prin valoarea ei. Aceasta se modifică atunci când jucătorul găsește un upgrade de daună într-un cufăr.

Meniu Principal

Odată cu pornirea jocului, jucătorului îi este prezentat meniul principal al jocului care îi permite acestuia să pornească jocul, să modifice anumite setări sau să părăsească jocul.

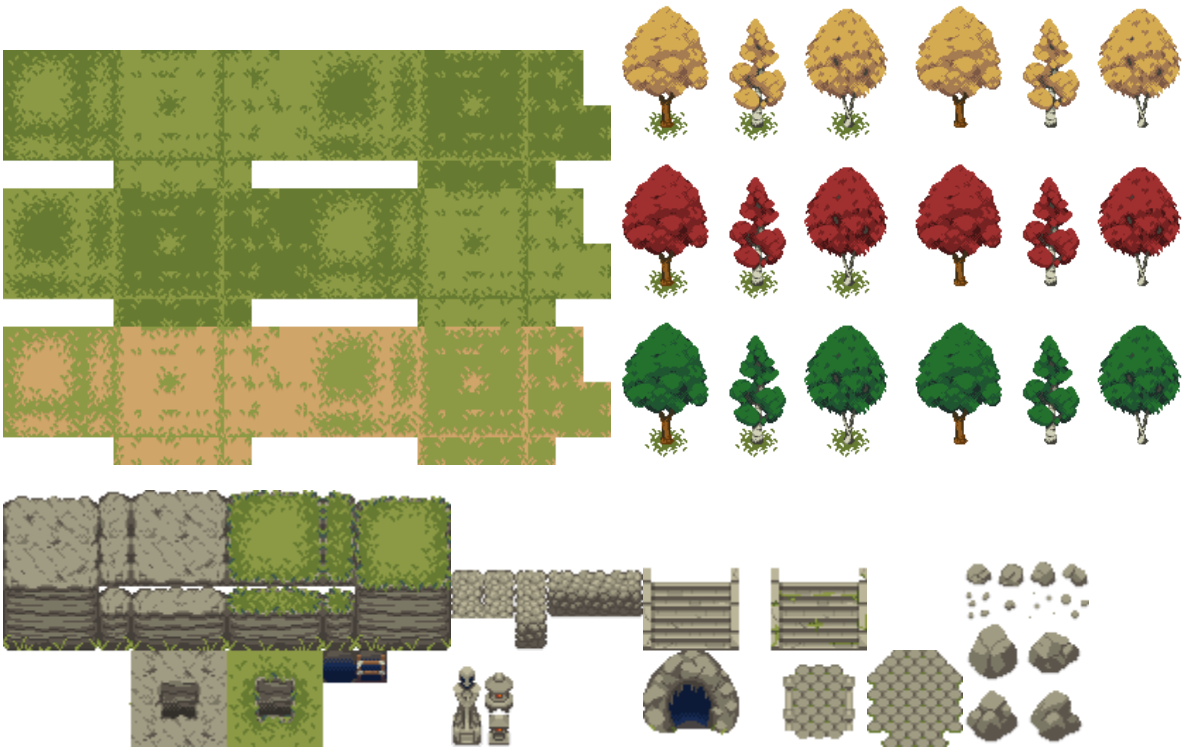


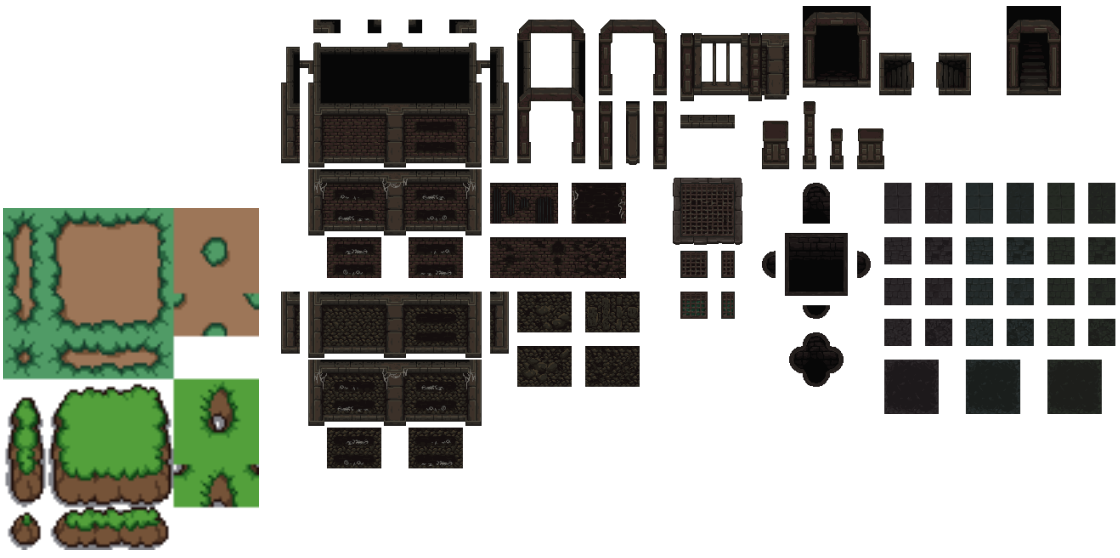
Meniu Setări

În meniul de setări, jucătorul poate modifica diferite aspecte ale jocului precum: dezactivarea efectelor sonore, muzicii de fundal sau activarea/dezactivarea CHEAT modului, care modifică niște valori ale caracterului principal astfel încât jocul să devina semnificativ mai ușor de trecut.

Atunci când jucătorul este învins, pe ecran va aparea un mesaj precum următorul:

Sprite-uri folosite





Bibliografie

Grandmaster Knight: https://opengameart.org/content/simple-knight

Forest: https://pixivan.itch.io/top-down-forest-tileset

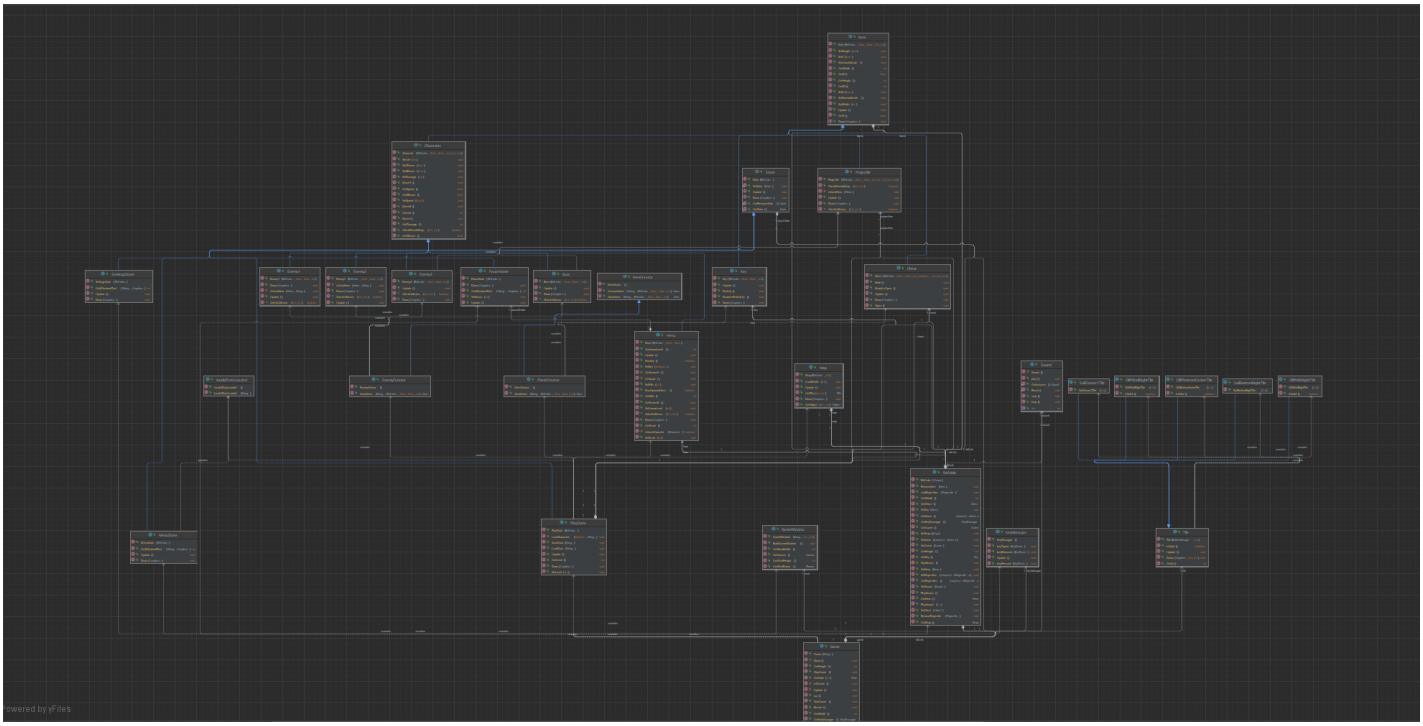
Mystic Woods: https://game-endeavor.itch.io/mystic-woods

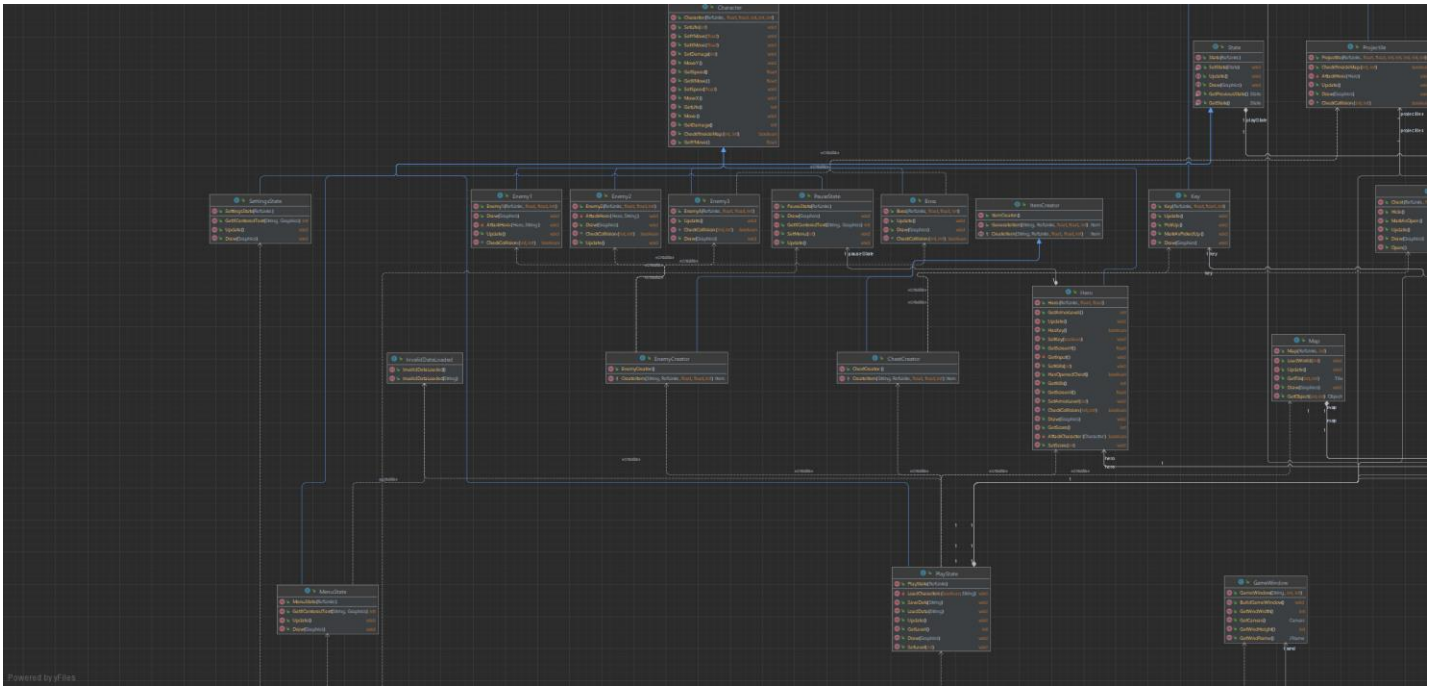
Catacombs: https://szadiart.itch.io/rogue-fantasy-catacombs

Enemies: https://deepdivegamestudio.itch.io/magical-asset-pack

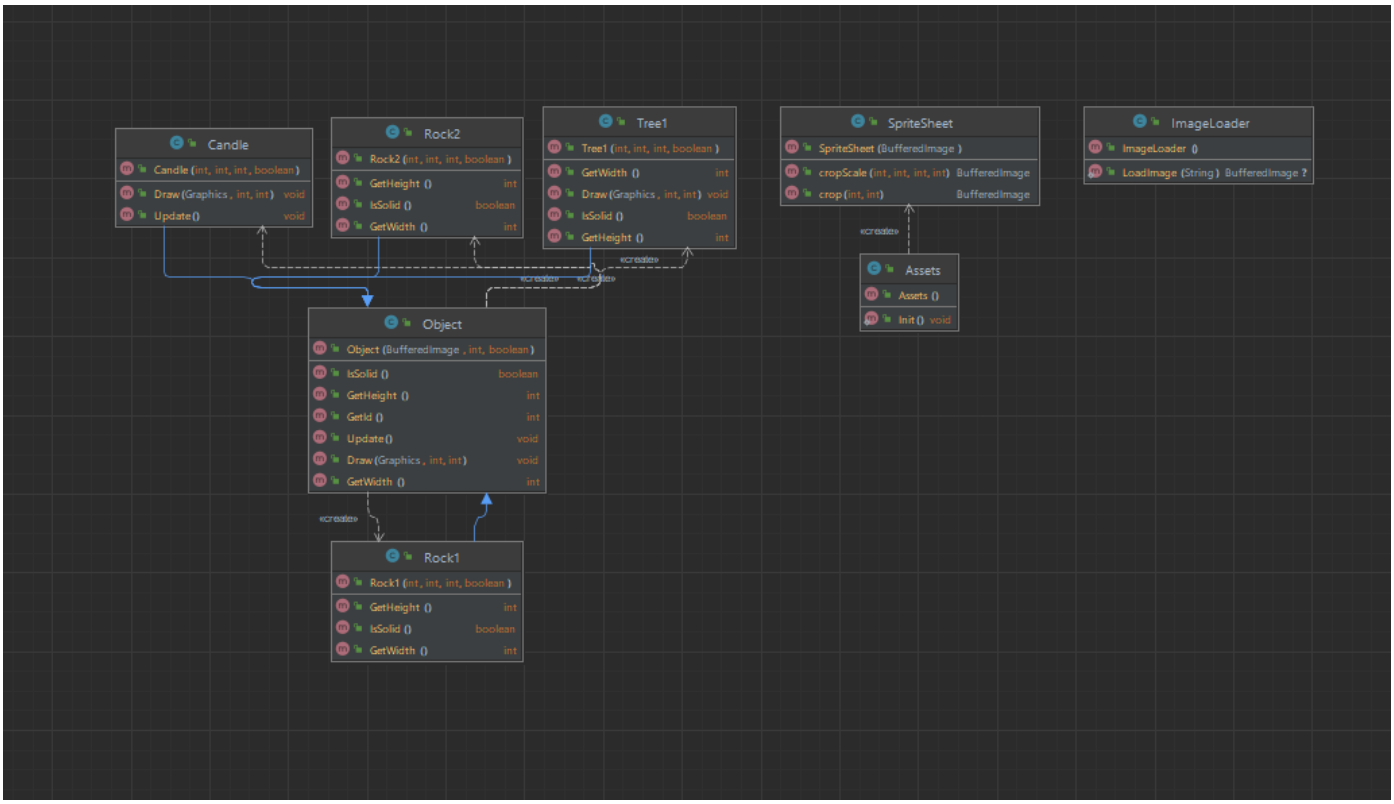
Diagramă Proiect

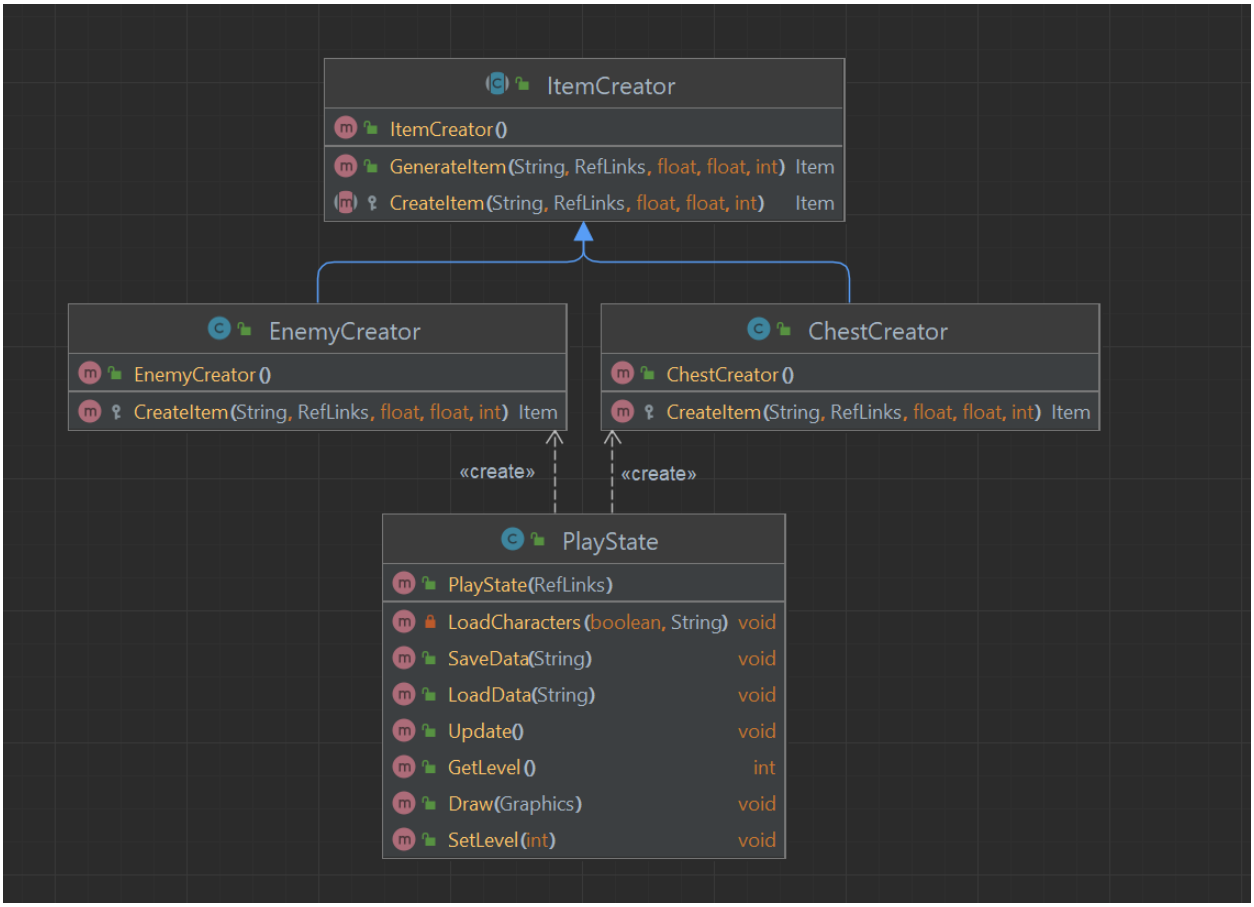
Se recomandă vizualizarea imaginii atașate proiectului pentru o întelegere mai bună a diagramei.






Item
Slate
Projectile
Boss
ItemCreator
Key
Chest
Hero
Map
Sound
SoilCornerTile
CliffMidRightTile
CliffBottomCenterTile
SoilCornerRightTile
CliffMidRightTile
ChestCreator
Powered by yFiles
RefLinks
GameWindow
KeyManager
Tile
Game
Main
Powered by yFiles


Candle
Candle (int, int, int, boolean)
Draw (Graphics , int, int) void
Update() void
Rock2
Rock2 (int, int, int, boolean)
GetHeight () int
IsSolid () boolean
GetWidth () int
Tree1
Tree1 (int, int, int, boolean)
GetWidth () int
Draw (Graphics , int, int) void
IsSolid () boolean
GetHeight () int
SpriteSheet
SpriteSheet (BufferedImage)
cropScale (int, int, int, int) BufferedImage
crop (int, int) BufferedImage
ImageLoader
ImageLoader ()
LoadImage (String) BufferedImage ?
Assets
Assets ()
Init () void
Object
Object (BufferedImage , int, boolean)
IsSolid () boolean
GetHeight () int
GetId () int
Update() void
Draw (Graphics , int, int) void
GetWidth () int
Rock1
Rock1 (int, int, int, boolean)
GetHeight () int
IsSolid () boolean
GetWidth () int
«create»



s

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

Descriere clase proiect

Main

După cum se poate observa din diagrama prezentată mai sus, această clasă creează o instanță de tipul Game, iar mai apoi pornește fluxul activităților prin apelul metodei StartGame() din Game.

Game

Rolul acestei clase este de a face inițializările (setează state-ul inițial, inițializează dalele, etc.), dar și de a menține interfața grafică la zi cu cea ce se întâmplă în joc. Această clasă implementează interfața Runnable pentru a avea comportamentul unui fir de execuție (thread). În momentul apelului metodei StartGame() se instanțiază un obiect de tip Thread pe baza instanței curente a clasei Game.

Orice obiect de tip Thread trebuie să implementeze metoda run() care este apelată atunci când firul de execuție este pornit (start()). Această metodă run() inițializează jocul prin crearea unei instanțe GameView, iar mai apoi controlează numărul de cadre pe secundă printr-o buclă while și "pregătește" noua scenă (Update()) pe care o va desena pe interfața grafică (Draw()).

Metoda Update() actualizează starea jocului (de exemplu: modifica poziția jucătorilor pe baza tastelor apăsate, crează diferite tile-uri (dale), etc.

Metoda Draw() va desena pe interfața grafică modificările făcute de metoda Update(). Interfața grafică este un canvas, făcând o analogie cu realitatea, poate fi considerată o pânză pentru desen, pe care sunt desenate diverse obiecte.

GameWindow

Această clasă este responsabilă cu fereastra în care vor fi desenate obiectele pe un canvas (se poate observa ca avem o metoda GetCanvas care întoarce canvasul). De asemenea, avem un obiect JFrame care permite desenarea de butoane, controale, textbox-uri, etc, dar poate conține și un canvas în care pot fi desenate diverse obiecte folosind texturi.

Tile / Object

Această clasă reține informații despre tile-urile (dalele) / obiectele din joc. În clasa Game există o instanță a clasei Tile, deoarece clasa Tile conține un vector cu obiecte tot de tipul Tile,așadar în acest vector vor fi stocate toate „dalele/tile-urile„

din joc în așa fel încat acest vector poate fi parcurs și pentru fiecare element se apelează metoda Update() și apoi Draw(). De asemenea, clasa Tile / Object mai reține și câte o instanță pentru fiecare sub-tip de tile (GrassTile, SoilTile, CliffTile, etc. precum și pentru obiectele Tree,Rock). În funcție de hartă, acel vector de tile-uri va conține tile-uri de acest tip.

GrassTile,SoilTile,CliffTile,Tree,Rock,etc

Toate aceste clase extind clasa Tile / Object, preluând comportamentul acelei clase sauputând să-l suprascrie folosind @Override. Constructorul acestora primește ca parametru un ID prin care se va putea identifica acel tile/object și apelează constructorul clasei de bază, adică al clasei Tile/Object, care primește ca parametru un membru static al clasei Assets. Membrul static folosit este ales în funcție de tipul clasei respective, de exemplu pentru GrassTile se va folosi membrul static grass.
Pentru obiecte, există și informații despre lățimea și înalțimea acestora. De asemenea, metoda Draw este suprascrisă pentru unele obiecte de dimensiuni diferite față de cele prestabilite, pentru a poziționa obiectul conform punctului de coliziune.

Assets

Conține câte un membru static de tip BufferedImage pentru fiecare tile/dală/obiect, chiar și pentru jucători/inamici. In acești membri sunt stocate imaginile, adică texturile acestora care vor fi desenate prin apelarea metodelor Draw() din fiecare clasa tile apelate din metoda Draw() din clasa Game. În acest exemplu este doar o imagine care conține toate texturile folosite, așadar pentru a putea instanța acest obiecte statice trebuie să decupăm fiecare textura din acea imagine. Pentru a face asta se folosește o instanță a clasei SpriteSheet în metoda Init(). Metoda folosită din SpriteSheet este crop(x, y) sau cropScale(x,y,scaleX,scaleY).

SpriteSheet

Constructorul acestei clase primește imaginea, iar clasa conține 2 membri constanți (final in Java) pentru înălțimea, respectiv lățimea texturilor (trebuie sa aibă toate aceleași dimensiuni). Metoda crop(x, y) primeste doi parametrii, x specifică pe ce coloană se află textura pe care dorim să o decupăm în imaginea cu toate texturile, iar y specifică linia. Pentru a afla efectiv poziția în imagine se determină pixelii de unde încep texturile de interes prin înmulțirea liniei/coloanei cu înălțimea, respectiv lățimea texturilor. Astfel se obține colțul din stânga sus al texturii, iar

pentru a afla celelalte colțuri se folosesc înălțimea, respectiv lățimea. Alternativ, se poate folosi metoda cropScale(x,y,scaleX,scaleY) pentru a obține o imagine cu o dimensiune mai mare decât cea prestabilită. (32px în loc de 16px)

ImageLoader

Înainte de a extrage fiecare textură, imaginea cu toate texturile trebuie să fie citită din memorie, iar pentru asta se folosește clasa ImageLoader cu metoda statică LoadImage(path) care primește ca parametru calea către imagine în memoria calculatorului.

RefLinks

Clasa RefLinks este o clasă care stochează o serie de referințe la diferite obiecte din joc pentru a fi ușor accesibile. Această clasă conține referințe la obiectul Game, la harta curentă, la eroul jocului și la o listă de obiecte de tip Item (obiecte cu care se poate interacționa).

Clasa conține câteva metode utile pentru a accesa aceste referințe. De exemplu, metoda GetKeyManager() returnează referința către managerul de evenimente de tastatură, iar metodele GetWidth() și GetHeight() returnează lățimea și înălțimea ferestrei jocului.

De asemenea, clasa conține metode pentru a seta sau accesa referințele la obiectele respective. De exemplu, metoda SetGame(Game game) setează referința la obiectul Game.

Această clasă este esențială pentru a crea legături între diferitele părți ale jocului și pentru a face programarea mai ușoară.

State

Clasa abstracta State implementează noțiunea abstractă de stare a jocului sau programului. Un joc avea diferite stări precum: Play, Settings, Menu, Pause, etc.

Clasa contine doua atribute statice, previousState si currentState, pentru a pastra referintele la starea anterioara si la starea curenta a jocului sau programului. De asemenea, clasa are si un atribut protected refLink, care contine o referinta la clasa RefLinks.

PauseState/MenuState/PlayState/SettingState

Clasele sunt niște subclase ale clasei abstracte State, care implementează diferite noțiuni de joc precum: Meniul, Setările, Jocul propriu-zis, etc.

Clasa PlayState controlează aspecte legate de jocul în sine. În interiorul acestei clase este definit eroul controlat de jucător, harta curentă precum și inamicii.

Pentru a genera inamicii, se folosește o fabrică de tipul EnemyCreator care prin metoda GenerateItem crează un inamic de tipul specificat la coordonatele specificate.

Metodele Update() și Draw() sunt responsabile pentru actualizarea și desenarea stării jocului, prin apelarea metodelor de actualizare și desenare ale eroului, hărții și inamicilor.

KeyManager

Clasa KeyManager este o clasă responsabilă de gestionarea intrării tastaturii și implementează interfața KeyListener. Scopul clasei este de a detecta atunci când o tastă a fost apăsată și de a seta un flag corespunzător, astfel încât programul să poată verifica ce taste sunt apăsate într-un anumit moment.

Clasa conține un vector de flaguri pentru toate tastele, unde fiecare tastă este reprezentată de un cod între 0 și 255. În plus, există un set de flaguri pentru tastele frecvent utilizate, cum ar fi sus, jos, stânga, dreapta și spațiu.

Metoda Update() este responsabilă pentru actualizarea valorilor flagurilor. Aceasta verifică dacă tastele relevante pentru sus, jos, stânga, dreapta și spațiu sunt apăsate și actualizează valorile corespunzătoare ale flagurilor.

Metodele keyPressed() și keyReleased() sunt apelate atunci când o tastă este apăsată sau eliberată. Ele setează flagurile corespunzătoare în vectorul de flaguri pentru a indica faptul că o tastă a fost apăsată sau eliberată.

Map

Clasa Map, implementează noțiunea de hartă și stocând informații despre codurile dalelor și obiectelor construite pe această hartă.

De asemenea, clasa Map conține și funcții de actualizare și de desenare a hărții, care parcurg matricea de dale și matricea de obiecte și desenează fiecare dală și obiectul corespunzător la poziția sa în spațiul de ecran. Funcțiile GetTile() și GetObject() returnează referințe la dalele și obiectele corespunzătoare codurilor

primite ca parametri, sau returnează o valoare predefinită în caz de eroare sau dacă obiectul respectiv nu a fost găsit.

ItemCreator

Clasa ItemCreator reprezintă o implementare a sablonului de proiectare "Metoda Fabrica". Aceasta este o clasă abstractă, care permite subclaselor să decidă ce clasă să instantieze pentru un anumit obiect, în funcție de tipul specificat.

Metoda GenerateItem este metoda principală a clasei, care primește ca parametri tipul de obiect dorit, referințele la obiectele și variabilele necesare pentru crearea obiectului, precum și coordonatele la care trebuie creat obiectul. Această metodă instantiază un obiect folosind metoda CreateItem a subclasei.

EnemyCreator

Clasa EnemyCreator este o subclasă a clasei abstracte ItemCreator și își asumă responsabilitatea de a crea inamici în joc.

Metoda CreateItem() din această clasă implementează funcționalitatea specifică de creare a unui obiect de tip inamic, în funcție de tipul specificat printr-un șir de caractere ("Enemy1","Enemy2",”Enemy3” sau “Boss”).

În cazul în care tipul specificat nu este unul dintre cele două tipuri posibile, metoda returnează null. Această metodă este apelată de metoda GenerateItem() din clasa de bază ItemCreator, care întoarce obiectul creat prin apelarea acestei metode.

Sound

Clasa Sound este o clasă responsabilă de gestionarea sunetelor într-un joc. Aceasta conține mai multe funcționalități pentru redarea sunetelor și setarea fișierelor audio corespunzătoare. Metodele folosite pentru a reda sunetele sunt setFile(), play(), loop() și stop().