Kotlyn: Quest for the Sacred Flame

Leșan Vasile
1207B

1. Proiectarea contextului

Povestea jocului

Totul începe într-o lume magică a unui ev mediu timpuriu, plină de creaturi fantastice, păduri fermecate și peisaje minunate, unde elementele reale și cele fantastice conviețuiesc și evoluează în liniște și pace.

Oamenii sunt prieteni cu zânele și elfii, zmeii și spiridușii, îngrijesc dragoni și cultivă pământul, goblinii făuresc arme și tot felul de unelte, iar gnomii sunt cei mai buni meșteri din tot ținutul Kotlyn. Toți trăiesc fericiți în armonie și prosperitate având grijă de Flacăra Eternă a lui Adranos, zeul celest al Luminii și Focului. Flacăra dăinuiește de mii de ani în acest loc fantastic, fiind dăruită de însuși Adranos, într-o vreme când acest ținut era dominat de un întuneric adânc și malefic, unde sălășluiau tot felul de creaturi demonice, spirite și duhuri rele, iar pericolul apărea la orice pas. Odată cu primirea acestui foc mistic, lumina s-a răspândit peste întreg Kotlyn-ul, răspândind căldură și cunoaștere peste tot și toate, transformând acest ținut nefast, rece și neprimitor într-unul prosper, călduros și ospitalier.

Toate acestea până într-o zi, când un vrăjitor malefic, pe nume Kali, plănuiește să fure acest foc, lăsând ținutul Kotlyn din nou în întuneric, pradă răului și creaturilor malefice. Prin multe șiretlicuri, deghizări și alte trucuri, Kali reușește să pună mâna pe Flacăra Eternă, fugind cu aceasta și ascunzându-se în ținuturi îndepărtate, în creierii munților, într-un loc în care, crede el, nu ar putea fi găsit niciodată. Totuși, un cavaler curajos pe nume Cyrus nu acceptă ca lucrurile să rămână astfel și hotărăște să plece în căutarea vrăjitorului pentru a recupera Flacăra Eternă și a restabili ordinea lucrurilor. În aventura sa Cyrus va întâlni provocări neobișnuite, obstacole magice și creaturi mitice ce vor încerca să-l oprească din această călătorie, iar puterile magice ale vrăjitorului nu îi vor face misiunea mai ușoară.

2. Proiectarea Sistemului

Jocul este de tip Platformer RPG (Role-Playing Game), vederea fiind laterală și ortografică. Tematica jocului este fantastică și de acțiune, jucătorul preluând controlul asupra protagonistului, Cyrus, și ghidându-l în misiunea sa.

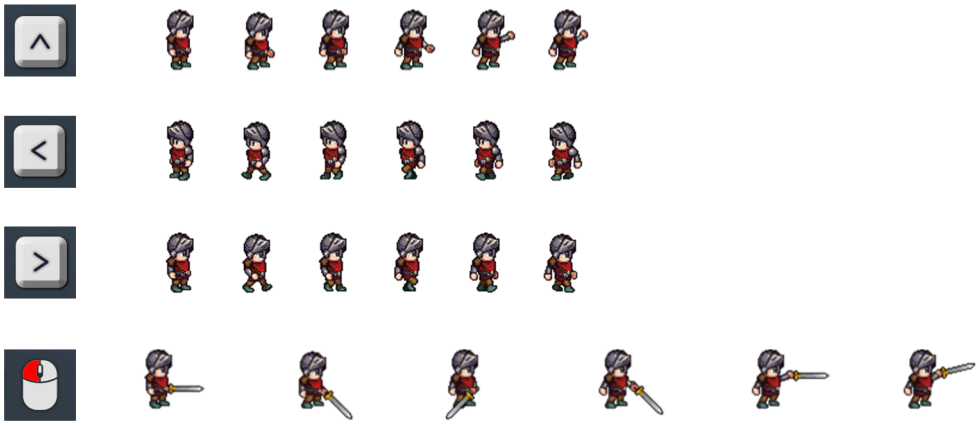
Jocul are 3 niveluri, în care protagonistul va înfrunta diferiți dușmani și va întâlni diverse provocări pe care trebuie să le depășească. Pe parcursul celor 3 niveluri se întâlnesc două tipuri majore de gameplay: defensiv și ofensiv. Jucătorul, aflat în poziția lui Cyrus, trebuie să se ferească de anumiți dușmani de care poate fi rănit, dar pe care nu îi poate elimina (spiritele). De asemenea, poate alege ca pe unii dușmani să îi rănească sau doar se se ferească de ei (lichi, demoni), iar pe alții e nevoit să îi elimine deoarece aceștia nu îl vor lăsa să treacă la următorul nivel.

Starea jucătorului este vizibilă în bara de sănătate (Health Bar), aceasta având un rol foarte important în evoluția jocului, deoarece pe parcursul nivelurilor sănătatea jucătorului nu se va regenera. În momentul în care bara de sănătate se golește complet, Cyrus moare, jocul se termină, iar vrăjitorul Kali scapă cu Flacăra Eternă a lui Adranos, ținutul Kotlyn rămânând pentru totdeauna în întuneric și teroare. Pe parcursul jocului cavalerul trebuie să găsească ieșirea din turnul în care Flacăra a fost păstrată, să parcurgă întreg ținutul Kotlyn care este împânzit acum de creaturi rele și să se aventureze în munți pentru a-l prinde pe Kali și a recupera focul magic.

Ca abilități, Cyrus este un bun atlet și mânuiește sabia cu precizie, înușiri ce îl vor ajuta să sară de pe o platformă pe alta în ascensiunea sa și să se lupte cu inamicii pe care îi va întâlni.

Jucătorul are la dispoziție următoarele controale:

- WASD sau respectiv ↑←↓→ pentru salt, mișcare stânga, jos, mișcare dreapta;
- SPACE pentru a sări;
- Click stânga pentru a folosi sabia;
- P/ESC pentru a deschide/închide meniul de pauză;



3. Proiectarea conținutului

Caractere:

Cyrus: protagonistul jocului, un cavaler curajos și însetat de dreptate ce refuză cu îndârjire să accepte noua lume în care este nevoit să trăiască de acum înainte. Determinat să facă dreptate și să nu lase răul să câștige, el pleacă pe urmele vrăjitorului malefic pentru a recupera Flacăra Eternă a lui Adranos. Evoluția lui Cyrus are o influență foarte mare asupra deznodământului jocului, deoarece, pe parcursul acestuia, starea sănătății cavalerului nu se va regenera. Dacă pe parcursul nivelurilor cavalerul pierde prea multă viață, la final, când se va întâlni cu vrăjitorul și va lupta cu acesta, abilitățile protagonistului vor fi mult diminuate, fiind obosit și rănit.

Kali: vrăjitorul malefic ce are puteri asupra numeroaselor elemente ale răului, acesta este foarte nemulțumit de actuala formă a lumii în care trăiește. El fură Flacăra Eternă a lui Adranos pentru a pune stăpânire peste ținutul Kotlyn și pentru a-i face pe locuitorii acestuia slugile sale. După ce fură flacăra și se ascunde în munți, acesta află de cavalerul ce este pe urmele lui și trimite tot felul de supuși pentru a-l împiedica să ajungă la el. Ca abilități, vrăjitorul poate ataca cu ajutorul toiagului sau magic sau poate folosi puterile magice pentru a planta obstacole înaintea cavalerului. În momentul în care vrăjitorul este înfrânt, pierde focul magic, iar lucrurile revin la normal, jocul fiind câștigat.

Mazikeen: demonul trimis de Kali pentru a-l împiedica pe Cyrus să-și ducă la bun sfârșit misiunea. Pe parcurs, el încearcă să îi întindă capcane cavalerului sau chiar îl înfruntă în luptă liberă încercând să-l elimine. Ca abilități, Mazikeen este foarte agil și rapid, fiind capabil a lupta corp la corp sau cu ajutorul tridentului său făurit în flăcările iadului.

Creaturi malefice: reprezentări ale răului precum spirite, lichi sau demoni. Acestea au un rol esențial în dinamica jocului, fiind la tot pasul și încercând să-l rănească sau să-l elimine pe protagonist. Ele au diferite însușiri și abilități în funcție de ce reprezintă fiecare.

- *Spiritele* sau fantomele sunt entități imateriale, fără formă sau trup, fiind imposibil de atins sau de rănit. Acestea seacă de viață și de energie orice ființă vie pe care o ating, fiind singurele creaturi malefice imposibil de învins.

- *Lichii* sunt creaturi mitice, nemuritoare, care au fost odinioară oameni și care au obținut nemurirea cu ajutorul magiei negre, căpătând ulterior o formă scheletică, inumană. Principala lor abilitate constă în securea cu două tăișuri cu care pot ataca adversarul cu care se confruntă. Aceștia pot fi înfrânți doar prin distrugerea completă a trupului lor scheletic.

- *Demonii*, creaturi malefice ce provin din iad, pot ataca protagonistul sau îi pot întinde diferite capcane acestuia pentru a-l face să avanseze mai greu. Aceștia pot fi infranti doar prin eliminarea în luptă corp la corp.

Personajele au diverse animații precum: mers, salt, atac, ghemuire, moarte.

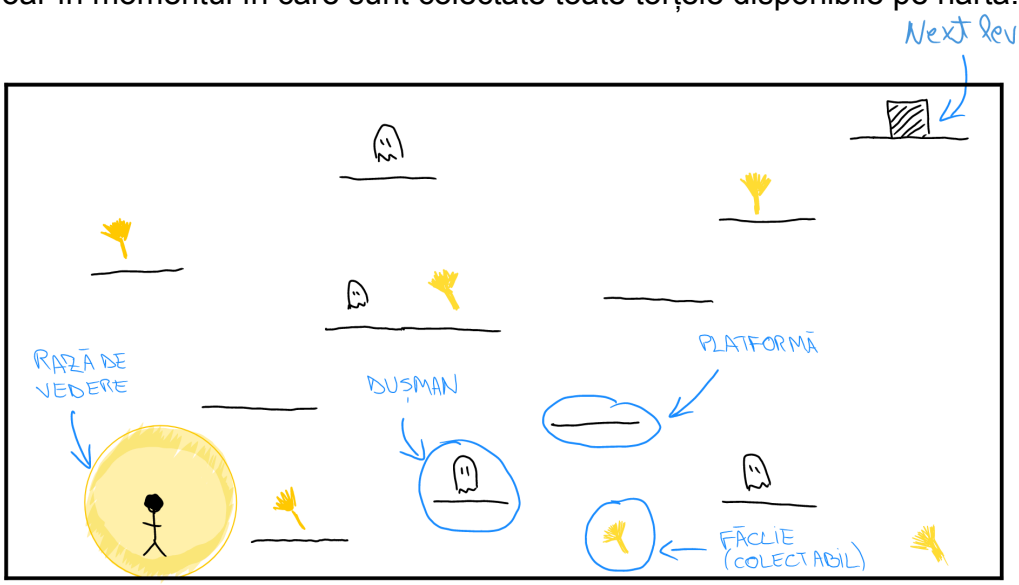
4. Proiectarea nivelurilor

Pentru fiecare nivel este corespunzător câte un tile-set în funcție de cadrul în care se desfășoară acțiunea: interiorul turnului în care a fost păstrată Flacăra, ținutul Kotlyn, munții și peștera unde se ascunde vrăjitorul. Pentru implementarea fundalului fiecărui nivel sunt atribuite imagini sugestive alcătuite din mai multe layere, implementând tehnica parallax scrolling. Pe parcursul nivelurilor, jucătorul va întâmpina capcane, sub forma gropilor cu țepi, de care trebuie să se ferească.

Toți inamicii sunt generați automat pe hartă, în funcție de ID-urile corespunzatoare ce se găsesc în matricea corespunzatoare hărții.

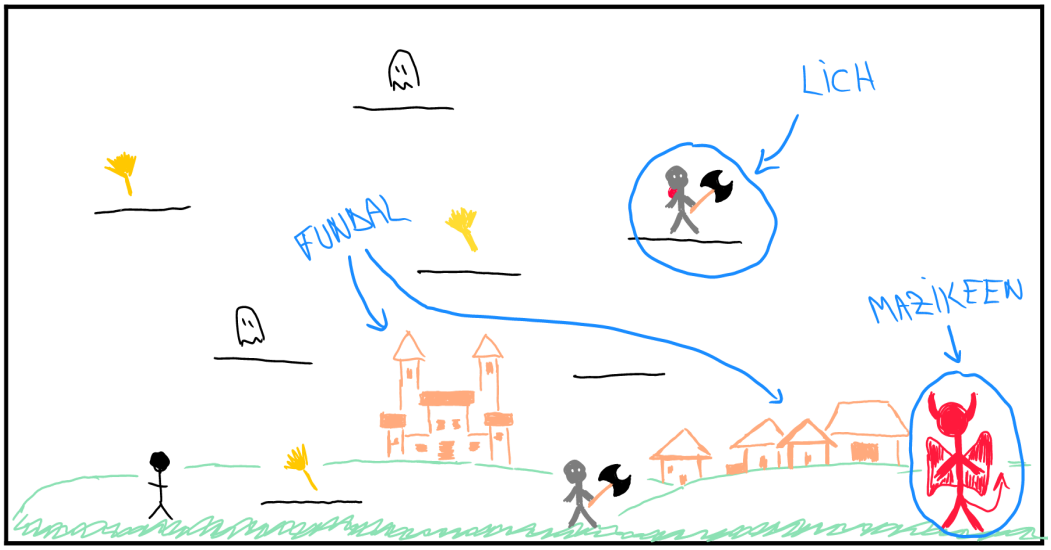
Nivelul 1 - Interiorul turnului

Acțiunea primului nivel se desfășoară în turnul unde a fost păstrată pentru mult timp Flacăra Eternă a lui Adranos. Jucătorul trebuie să găsească ieșirea din turn, lucru dificil din cauza întunericului în care acesta se află. Efectul de întuneric este implementat prin tehnica Fog of War. Protagonistul trebuie să colecteze torțele ce sunt plasate în diferite puncte pentru a lumina harta curentă. De asemenea jucătorul trebuie să se ferească de spiritele ce bântuie prin turn, neavând posibilitatea de a le răni. În momentul intersectării cu un spirit nivelul sănătății jucătorului scade. Pe parcurs ce jucătorul adună torțe, raza de vedere a acestuia crește, ajungând să cuprindă majoritatea ecranului. Nivelul este câștigat, iar ușa spre următorul nivel se deschide doar în momentul în care sunt colectate toate torțele disponibile pe hartă.



Nivelul 2 - Ținutul Kotlyn

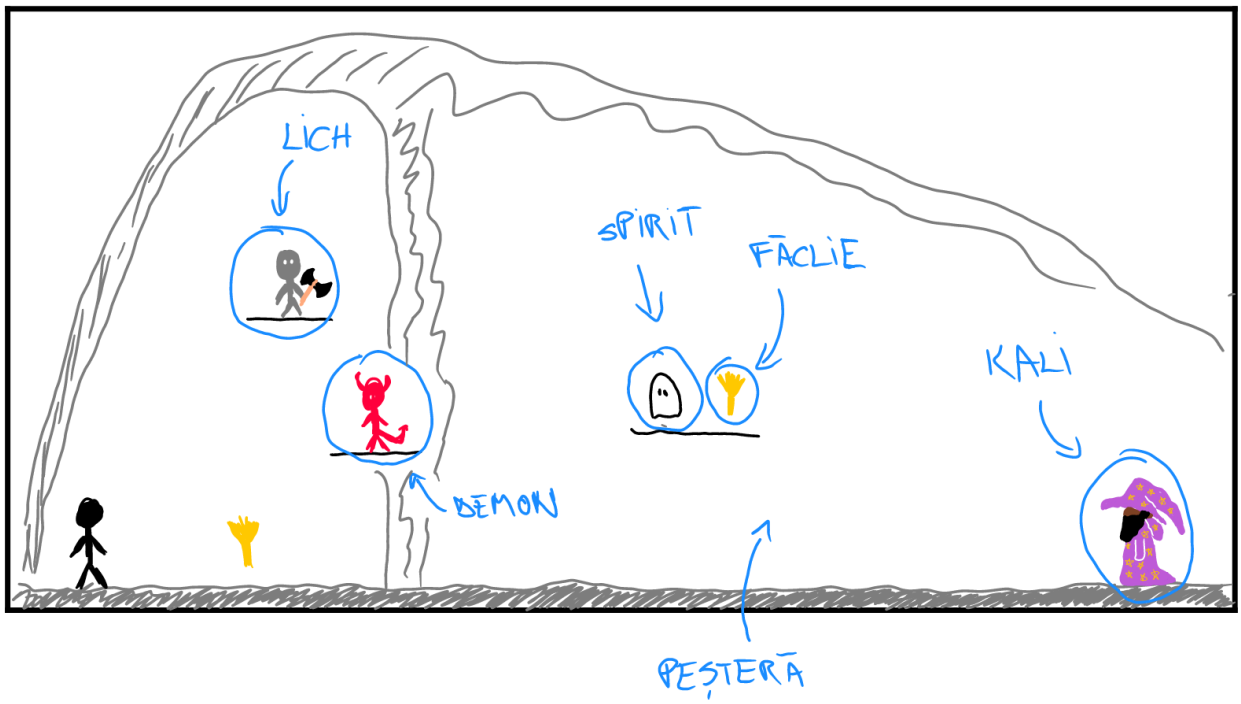
O dată ieșit din turn, jucătorul cutreieră ținutul Kotlyn care este împânzit cu spirite și lichii. Acesta trebuie să se ferească de spirite, la fel ca în nivelul precedent, iar în cazul lichiilor jucătorul poate alege dacă îi va înfrunta sau doar se va feri de aceștia. Scopul este de a trece prin întreg ținutul nevătămat și de a colecta în continuare torțe pentru a menține nivelul luminii constant. La final, pentru a trece spre următorul nivel, jucătorul trebuie să se confrunte cu Mazikeen, demonul vrăjitorului. Nivelul va fi câștigat doar după ce demonul este eliminat, neavând posibilitatea ca acesta să fie evitat.

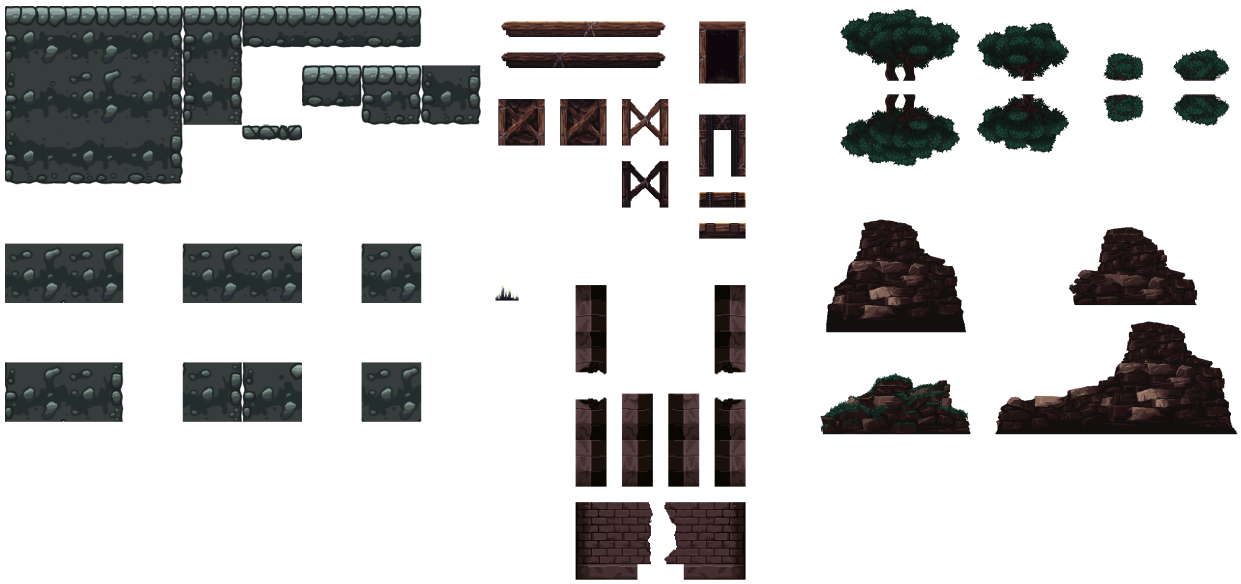


Nivelul 3 - Peștera vrăjitorului

După înfruntarea cu Mazikeen, jucătorul trebuie să parcurgă munții misterioși și să găsească peștera unde se ascunde vrăjitorul. Aici eroul întâlnește noi oponenți: demoni. Și în cazul acestora jucătorul poate opta să îi înfrunte sau să îi evite pentru a-și păstra nivelul sănătății cât mai ridicat. La final, protagonistul trebuie să se lupte cu vrăjitorul pentru a recupera flacăra magică, nivelul fiind terminat doar după ce vrăjitorul este înfrânt.

Jocul câștigat se încheie cu un final fericit în care Flacăra Eternă a lui Adranos revine la locul ei de odinioară, lumina se reinstalează peste întreg ținutul, iar creaturile malefice sunt alungate definitiv, totul revenind la normal.





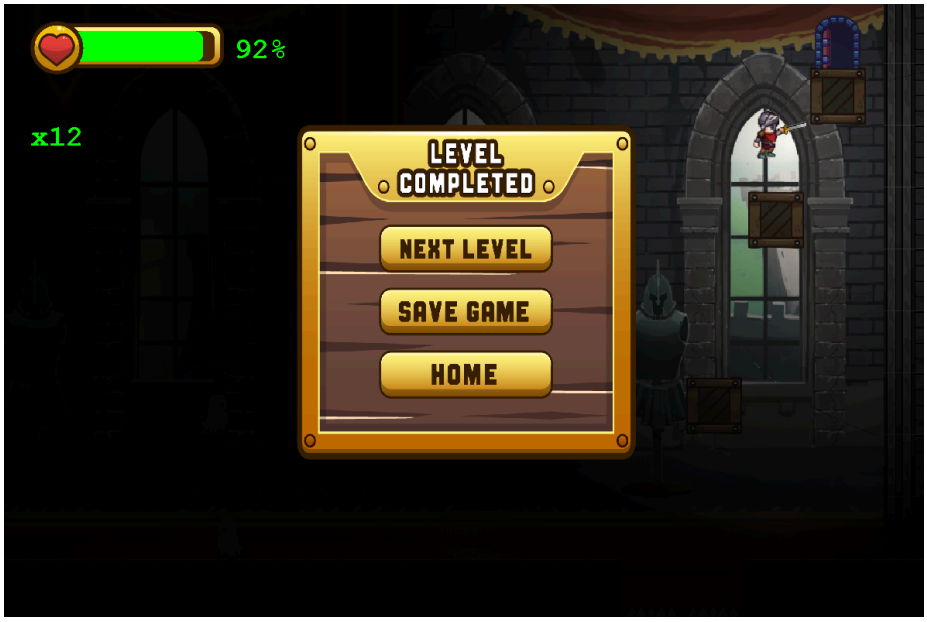
5. Proiectarea interfeței cu utilizatorul

Interfața cu utilizatorul este una simplă în timpul desfășurării jocului, pe ecran fiind vizibilă bara de sănătate, asociată stării jucătorului și numărul de elemente colectabile (făclii) adunate pe parcurs.

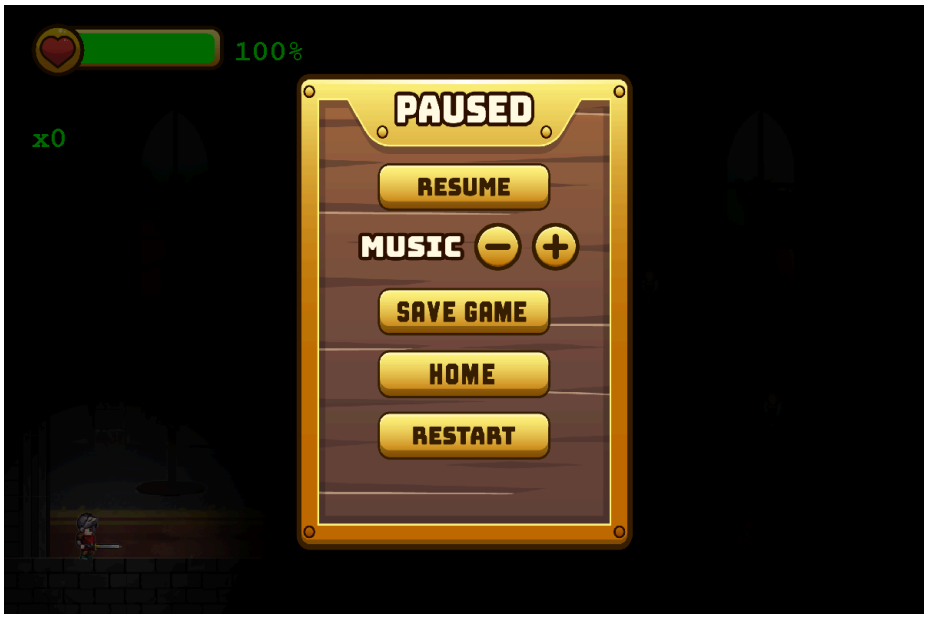
La intrarea în joc, meniul principal oferă jucătorului mai multe posibilități:

- Play - începerea unui nou joc
- Options - deschiderea unui panou de unde jucătorul poate opri/porni sunetul
- Quit - închiderea jocului

Atunci când un nivel este câștigat, pe ecran va fi afișat un meniu de unde jucătorul poate alege să treacă la următorul nivel, să salveze statusul jocului curent sau de a se întoarce în cadrul meniului principal.



Meniul de pauză oferă utilizatorului posibilitatea de a continua, de a salva, de a reîncepe jocul sau de a accesa meniul principal al jocului. De asemenea sunt disponibile doua butoane pentru reglarea volumului muzicii de fundal.



6. Descrierea claselor

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 (de exemplu de a crea instanțe pentru jucători/inamici, pentru a inițializa harta, 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()). Metoda Update() actualizează starea jocului (de exemplu: modifica poziția jucătorilor pe baza tastelor apăsate, schimbă poziția inamicilor folosind chiar tehnici de inteligență artificială, crează diferite tile-uri (dale), etc. Metoda Draw() va desena pe interfața grafică modificările făcute de metoda Update().

GameSingleton

Această clasă iimplementează un design pattern Singleton pentru clasa Game, asigurându-se că există întotdeauna o singură instanță a jocului în timpul rulării aplicației.

Package-ul GameWindow

GameWindow

După cum discutam mai sus, metoda run() a clasei Game crează o instanță a clasei GameWindow. De asemenea, avem un obiect JFrame care permite desenarea de butoane, controale, textbox-uri, etc.

GamePanel

Această clasă GamePanel reprezintă panoul principal în care jocul este desenat și afișat. În esență, această clasă oferă o interfață grafică pentru desenarea și afișarea jocului. Ea primește evenimente de la tastatură și de la mouse, pe care le transmite mai departe instanței de joc pentru a le gestiona, iar împreună cu clasa GameWindow, formează fereastra unde vor fi afișate acțiunea jocului, meniurile, butoane și alte elemente grafice.

Package-ul Tiles

Tile

Această clasă reține informații despre tile-urile (dalele) 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 mai reține și câte o instanță pentru fiecare sub-tip de tile (*ClosedDoorTile, DirtPlatform, DirtTile, FloorTile, LeftWallTile, RightWallTile, OpenDoorTile, PlatformTile, RightWallDirt*). În funcție de hartă, acel vector de tile-uri va conține tile-uri de acest tip.

ClosedDoorTile, DirtPlatform, DirtTile, FloorTile, LeftWallTile, RightWallTile, OpenDoorTile, PlatformTile, RightWallDirt

Toate aceste clase extind clasa Tile, preluând comportamentul acelei clase sau putând să-l suprascrie folosind @Override. Constructorul acestora primește ca parametru un ID prin care se va putea identifica acel tile și apelează constructorul clasei de bază, adică al clasei Tile, care primește ca parametru un membru static al clasei Assets. Membrul static folosit este ales în funcție de tipul clasei respective, de exemplu pentru ClosedDoorTile se va folosi membrul static closedDoor.

Package-ul Graphics

Assets

Conține câte un membru static de tip BufferedImage pentru fiecare tile/dală, 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).

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 15 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.

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.

Package-ul Entities

Entities

Aceasta este o clasă abstractă ce definește caracteristicile și comportamentele fundamentale ale entităților din joc, precum player sau inamici. Aceasta include coordonatele, dimensiunile, starea animațiilor, viteza de mișcare, sănătatea și gestionarea coliziunilor.

EntitiesHandler

Clasa EntititesHandler gestionează toate entitățile inamice din joc. Aceasta se ocupă de inițializarea, actualizarea, desenarea și resetarea entităților, precum și de verificarea coliziunilor între atacurile jucătorului și inamici. Clasa interacționează strâns cu starea jocului (Playing) și cu nivelurile definite în joc.

Enemy

Clasa Enemy este o clasă abstractă care extinde clasa Entity și definește comportamentele și atributele specifice entităților inamice din joc, permițând extinderea și specializarea acestora pentru diferite tipuri de inamici cu comportamente unice. Aceasta gestionează mișcarea, animațiile, detectarea și atacul jucătorului, precum și stările inamicului (sănătatea și direcția de mers).

Player

Clasa Player este o extindere a clasei Entity și reprezintă jucătorul în joc. Aceasta gestionează mișcarea, atacurile, sănătatea, animațiile și interacțiunile jucătorului cu mediul și entitățile din joc, asigurându-se că toate aceste aspecte sunt gestionate corect.

Spirit, Demon, Lich, Maze, Kali

Toate aceste clase sunt extinderi ale clasei Enemy și reprezintă inamici de diverse tipuri în joc: Spirit, Demon, Lich, Maze, Kali. Acestea gestionează mișcarea, atacurile și animațiile acestor inamici, precum și interacțiunile lor cu jucătorul și mediul înconjurător, preluând sau suprascriind comportamentul clasei Enemy și specializându-l conform fiecărui inamic.

Collision

Clasa Collision conține metode statice pentru a verifica coliziunile între entitățile jocului (de exemplu, jucătorul și inamicii) și mediul înconjurător reprezentat de tile-urile nivelului curent. Aceste metode ajută la gestionarea mișcării entităților și determină dacă o anumită zonă din joc este accesibilă sau solidă.

Package-ul Gamestates

StateMethods

Această interfață, StateMethods, definește un set de metode pe care fiecare clasă ce reprezintă o stare a jocului trebuie să le implementeze. Aceste metode sunt

responsabile pentru actualizarea stării, desenarea acesteia pe ecran și gestionarea evenimentelor de input, cum ar fi clicuri de mouse sau apăsări de taste.

GameState

Enumerarea GameState definește diferitele stări în care jocul poate fi pe parcursul execuției sale. Aceasta este folosită pentru a controla fluxul de execuție al jocului, permițând trecerea între meniuri, jocul propriu-zis, opțiuni, și alte funcționalități.

State

Aceasta servește ca superclasă pentru fiecare stare a jocului din cadrul jocului. Clasa nu conține implementări specifice ale stărilor jocului, ci mai degrabă definește structura de bază și funcționalitatea comună care va fi împărtășită de toate stările, permițând o interacțiune bună între stări și joc. Implementând aceste metode în clasele specifice stărilor jocului, se asigură că fiecare stare poate reacționa la evenimentele de input și poate fi actualizată și desenată corespunzător pe ecran.

Menu

Această clasă Menu reprezintă o stare specifică a jocului care afișează un meniu. Este subclasă a clasei State și implementează interfața StateMethods, astfel încât să poată gestiona evenimentele de input și actualizările stării.

Playing

Această clasă Playing reprezintă starea jocului în care player-ul se află. Ea este subclasă a clasei State și implementează interfața StateMethods, permițând gestionarea evenimentelor de input și actualizarea stării jocului.

Package-ul Inputs

KeyHandler

Clasa KeyHandler este responsabilă pentru gestionarea evenimentelor de la tastatură în cadrul jocului. Implementează interfața KeyListener pentru a putea gestiona evenimentele tastaturii. Această clasă asigură o modalitate eficientă de gestionare a evenimentelor de la tastatură și de rutare a acestora către starea de joc corespunzătoare pentru a fi procesate. Astfel, logica specifică a tastelor poate fi gestionată separat în clasele de stare ale jocului.

MouseHandler

MouseHandler implementează interfețele MouseListener și MouseMotionListener, permițând gestionarea evenimentelor mouse-ului în cadrul jocului. Clasa MouseHandler oferă o modalitate simplă și eficientă de a captura și gestiona evenimentele mouse-ului în cadrul jocului, permițând interacțiunea fluidă cu interfața utilizatorului, permițând gestionarea separată a evenimentelor mouse-ului în clasele de stare ale jocului.

Package-ul Levels

Level

Această clasă Level este responsabilă pentru gestionarea datelor și entităților asociate unui nivel din joc, facilitând accesul și manipularea acestora.

LevelManager

Clasa LevelManager este responsabilă pentru gestionarea diferitelor niveluri din joc, inclusiv încărcarea lor, actualizarea și desenarea fundalului, precum și manipularea entităților și obiectelor asociate nivelului curent, fiind managerul central al nivelurilor.

Package-ul Objects

GameObject

Această clasă GameObject reprezintă un obiect generic din joc și servește ca superclasă pentru diferite tipuri de obiecte, cum ar fi capcane, obiecte colectabile sau de mediu. GameObject oferă funcționalități de bază care sunt utile pentru majoritatea obiectelor din joc, cum ar fi gestionarea coliziunilor, desenarea hitbox-ului și actualizarea animațiilor. Subclasele pot extinde această clasă și pot adăuga comportamente specifice tipurilor de obiecte pe care le reprezintă.

ObjectHandler

Clasa ObjectHandler este responsabilă pentru gestionarea obiectelor din joc, cum ar fi obiectele colectabile și capcanele, permițându-le să interacționeze cu jucătorul și cu mediul înconjurător într-un mod corespunzător.

Spike

Clasa Spike este utilizată pentru a crea obiecte capcană de tip spike („țepușă”) în joc și definește caracteristicile specifice ale acestor obiecte, precum dimensiunile hitbox-ului și offset-urile de desenare.

Torch

Clasa Torch este utilizată pentru a crea obiecte colectabile de tip torch („torță”) în joc și definește caracteristicile specifice ale acestor obiecte, precum dimensiunile hitbox-ului, actualizarea animației, levitarea și offset-urile de desenare.

Package-ul UserInterface

MenuButton

Clasa MenuButton definește comportamentul general al butoanelor din meniul jocului. Ea servește ca șablon pentru butoanele din meniul jocului, furnizând metode generice pentru manipularea și desenarea acestora. Subclasele acestei clase pot adăuga comportament specific pentru fiecare tip de buton. De asemenea, clasa MenuButton mai reține și câte o instanță pentru fiecare sub-tip de buton (.....).

PauseButton

Clasa PauseButton definește comportamentul general al butoanelor din meniul de pauză a jocului. Ea servește ca șablon pentru butoanele din meniul de pauză a jocului, furnizând metode generice pentru manipularea și desenarea acestora. Subclasele acestei clase pot adăuga comportament specific pentru fiecare tip de buton. De asemenea, clasa PauseButton mai reține și câte o instanță pentru fiecare sub-tip de buton (.....).

LoadGameButton, OptionsButton, PlayButton, QuitButton

Toate aceste clase extind clasa MenuButton, preluând comportamentul acelei clase sau putând să-l suprascrie folosind @Override. Aceste clase reprezintă implementări specifice a butoanelor de meniu pentru diferite acțiuni implementând funcționalitățile generice definite în clasa lor părinte, MenuButton, pentru manipularea și desenarea butonului.

HomeButton, NegativeButton, NextLevelButton, PositiveButton, RestartButton, ResumeButton, RestartButton, SaveGameButton

Toate aceste clase extind clasa PauseButton, preluând comportamentul acelei clase sau putând să-l suprascrie folosind @Override. Aceste clase reprezintă implementări specifice a butoanelor de meniu pauză pentru diferite acțiuni implementând funcționalitățile generice definite în clasa lor părinte, PauseButton, pentru manipularea și desenarea butonului.

GameOverOverlay, LevelCompletedOverlay, PauseOverlay

Aceste clase se ocupă de desenarea și tratarea evenimentelor asociate ecranului de sfârșit de joc, a ecranului de nivel complet, respectiv a ecranului de pauză în timpul jocului. Aceste clase oferă interfețe grafice pentru gestionarea ecranelor mai sus menționate. Ele se integrează cu alte componente ale jocului pentru a asigura o experiență de joc coerentă și intuitivă pentru utilizatori.

Package-ul Utilities

Constants

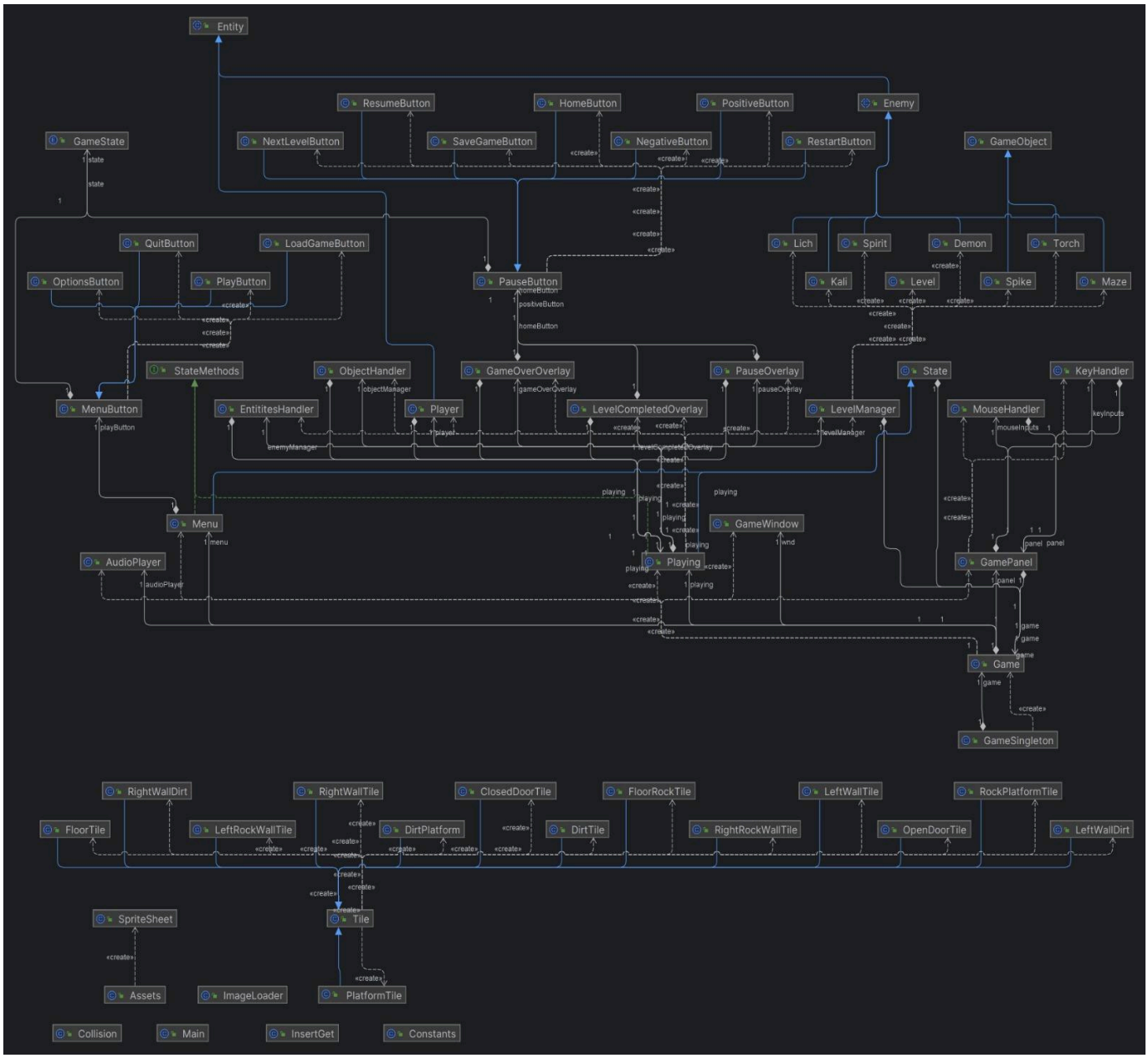
Clasa Constants este utilizată pentru a defini constantele folosite în joc. Aceste constante sunt utilizate în întreaga aplicație pentru a asigura consistența și ușurința în actualizarea și gestionarea valorilor fixe folosite. Definirea lor într-o clasă separată facilitează modificările și actualizările ulterioare.

7. Documentația proiectului și diagrama UML

Documentația proiectului Java poate fi consultată la adresa: https://drive.google.com/drive/folders/1PeiBt69mMM7rpgqKuG_V5zfE5QDQhPNr?usp=sharing

🗀 Documentatie Joc

Diagrama UML:



8. Bibliografie / Webografie

Bibliografie:

- Informații curs PAOO
- Documentația proiectului a fost generată cu ajutorul programului Doxygen
- Diagrama UML de clase a fost generată cu ajutorul IntelliJ IDEA Ultimate

Webografie:

- Imaginea de start a jocului a fost creată cu ajutorul inteligenței artificiale:
  Image Creator from Microsoft Designer

- Sprite-urile asociate caracterelor: Cyrus, Kali, Mazikeen, Lichi, Demoni sunt create de mine cu ajutorul site-ului:
  Universal LPC Sprite Sheet Character Generator

- Sprite-ul asociat caracterului Spirit a fost preluat de pe site-ul:
  Ghosts sprite - RPG TileSet Free Curated Assets for your RPG Maker MV Games!

- Tileset-urile asociate nivelurilor, elemente de decor, fundal și muzică ambientală:
  https://shorturl.at/mrC13