Titlu proiect (First Dungeon)

Autor (Murariu-Tănăsache Iulian)

Grupa(1211A)

Povestea jocului: Jocul este situate într-o lume fantastică, plină de comori ascunse peste tot in peșteri și temnițe și, de asemenea, de aventurieri în căutare de faimă și avere. Dintre acești aventurieri, se remarcă personajul principal al jocului, Davernius, care încearcă să găsească cea mai râvnită comoară dintre toate. Problema lui este că nu a mai fost niciodată într-o astfel de aventură și pleacă complet nepregătit pentru dificultățile ce le va întâmpina spre comoară și chiar la întoarcere, unde este așteptat de numeroase surprize (mai ales că este urmărit de cineva în aventura sa).

Prezentare joc: Campanii pentru un singur jucător în care jucătorul trebuie să parcurgă temnițe generate procedural pentru a ajunge la comoară și apoi să se întoarcă la ieșirea din dungeon.

Reguli joc:

- Jocul implică parcurgerea spre dreapta a temniței, jucătorul încercând să evite capcane și monștrii pentru a ajunge la camera comoarei. Odată ce găsește comoara, trebuie să se întoarcă prin camerele prin care a venit, luptând cu monștrii, acum fiind înarmat, pentru a ieși înapoi la suprafață.
- Jucătorul are trei vieți la început (4 sau 5 când obține comoara), iar în funcție de dificultatea aleasă poate să piardă o jumătate de inimă/o inimă întreagă când este atacat de inamici sau atinge capcane. De asemenea are și trei cercuri de stamina(rezistență). Stamina se umple când intri într-o camera nouă. Pentru a umple viața, trebuie culese inimioare de pe jos din diferite camere(valabil doar in modul normal).
- Jocul se încheie când ieși din temniță în viață și cu comoara sau când jucătorul rămâne fără inimi de viață și moare, astfel campania este încheiată.
- Jucătorul nu poate să treacă prin pereți solizi, poate părăsi camera doar printr-o ușă deschisă.
- Personajul se poate mișca în stânga/dreapta, poate sări și de asemenea poate să se “lanseze/avânte”(dash forward) în direcția de deplasare, lucru care îi va consuma un cerc de stamina. Poate ataca inamici doar dacă are o armă(în general, în partea a doua a jocului, după ce găsește comoara).

Personajele jocului:

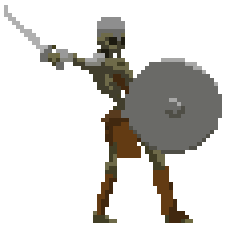
- Davernius, personajul principal al jocului și jucătorul-personaj. Un tânăr care vrea să se remarce în lumea aventurierilor și care vrea să obțină cea mai râvnită comoară, însă este complet lipsit de experiență. O persoană agilă, optimistă, care nu se dă bătută, indiferent de circumstanțe. Are totuși competențele necesare să mânuiască arme de bază(săbii, arcuri, topoare, etc.).

- Monștrii: diverse NPC-uri care vânează jucătorul și apără calea către comoară.
    - Monstrul Ochi de Ceapă

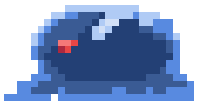
    - Monstrul Goblin

    - Monstrul Ciupercă

- Monstrul Scheletron



- Monstrul Slinos



- Aventurier malefic: Boss-ul final (NPC) în variant hardcore, un aventurier care l-a pândit pe Davernius și l-a așteptat să se întoarcă cu comoara, pentru a-l tâlhări.

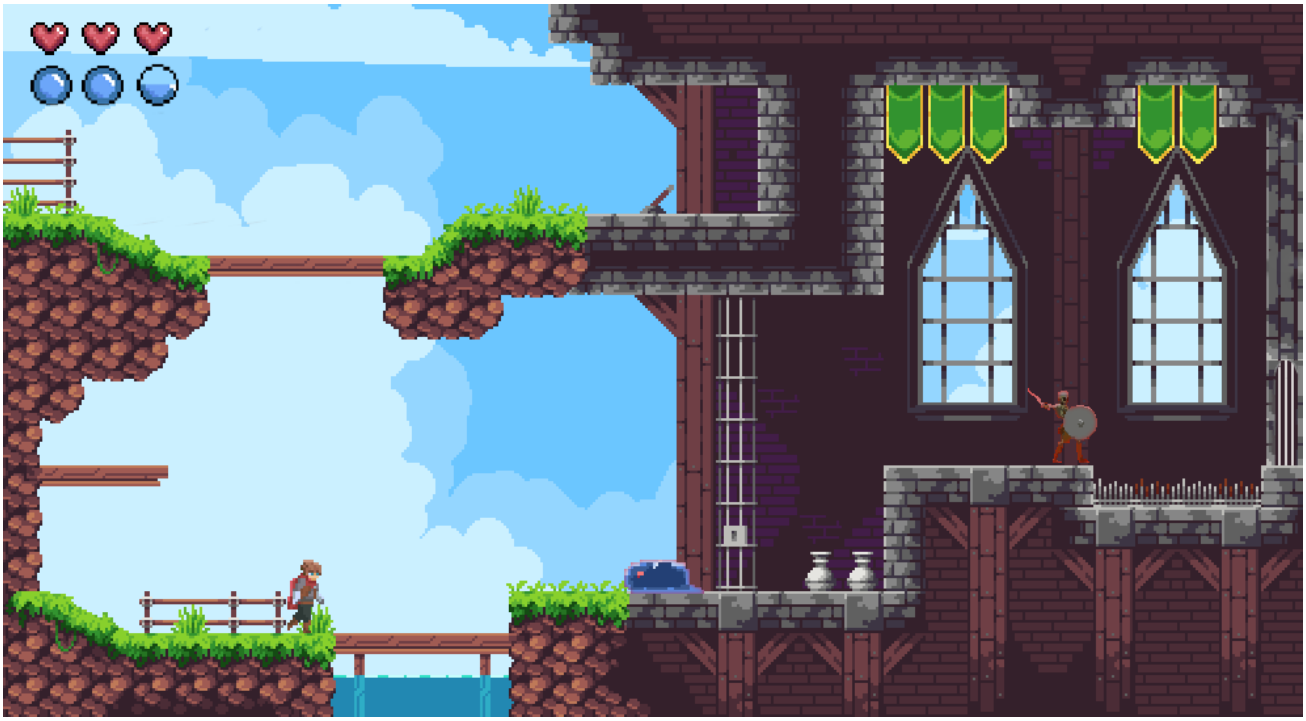
Tabla de joc:

- *Componente pasive* :
  - iarbă, sol, zid: imobile, solide, nu se trece prin ele.
  - ușă: solidă, nu se trece prin ea când este închisă, se poate închide sau deschide în funcție de cameră.
  - inimi: imobile, pot fi culese când sunt atinse și când viața jucătorului nu este plină, restaurează o inimă de viață
  - comoara: imobilă, poate fi culeasă, oferă viață în plus(armură) și o armă jucătorului, una singură per joc.
- *Componente active*:

  - inamici(monștrii + boss): personaje NPC, dacă sunt atinse iau din viață, se poate trece prin ele, vânează jucătorul.

  - capcane: imobile, dacă sunt atinse iau din viață, se poate trece prin ele.

- *Structura tablei de joc*: (elemente minime, dispunere etc.) si modul in care este construita :



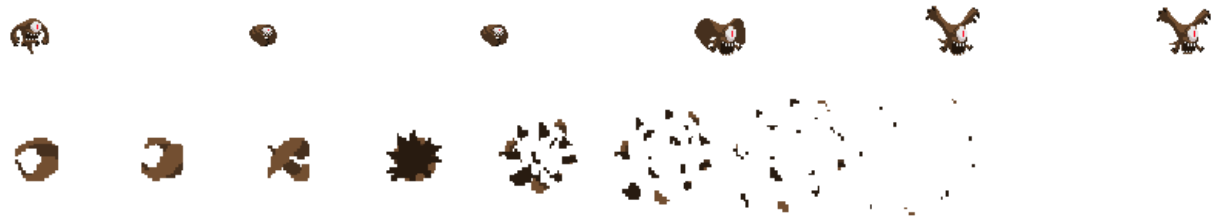
Mecanica jocului:

- *Controale*:
  - mers stânga/dreapta – A/D
  - săritură – SPACE
  - coborâre de pe platforme – S?
  - mers ghemuit / alunecare(+DASH?) – LEFT CTRL
  - lansare/avânt (DASH) - LEFT SHIFT
  - atac – L
- *Scor*: bazat pe ușurința cu care s-a găsit și adus comoara.
- Inamici și capcanele scad viața jucătorului la atingerea lor; Inamicii vânează jucătorul.
- Mecaniciile jocului sunt destul de intuitive și generale, tutorial scurt la început, în afara temniței, cu câteva demonstrații ale controalelor.
- Mai multe căi către comoară
- Perspectivă perpendicular + parallax

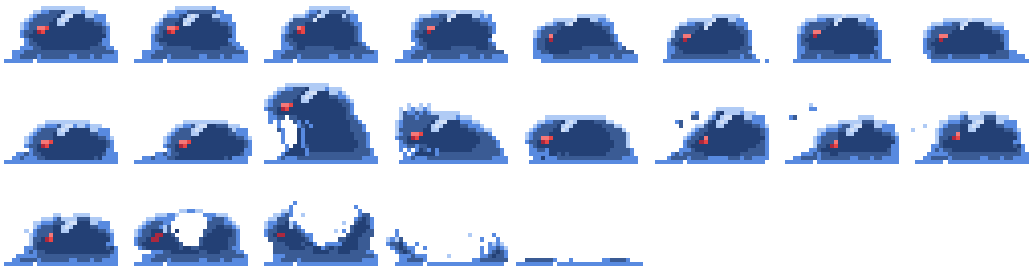
Game sprite:

- Câteva exemple de sprite sheet-uri:

- Ochi de Ceapă:



- Slinosul:



- Davernius în prima parte a jocului:

Descriere fiecare nivel:

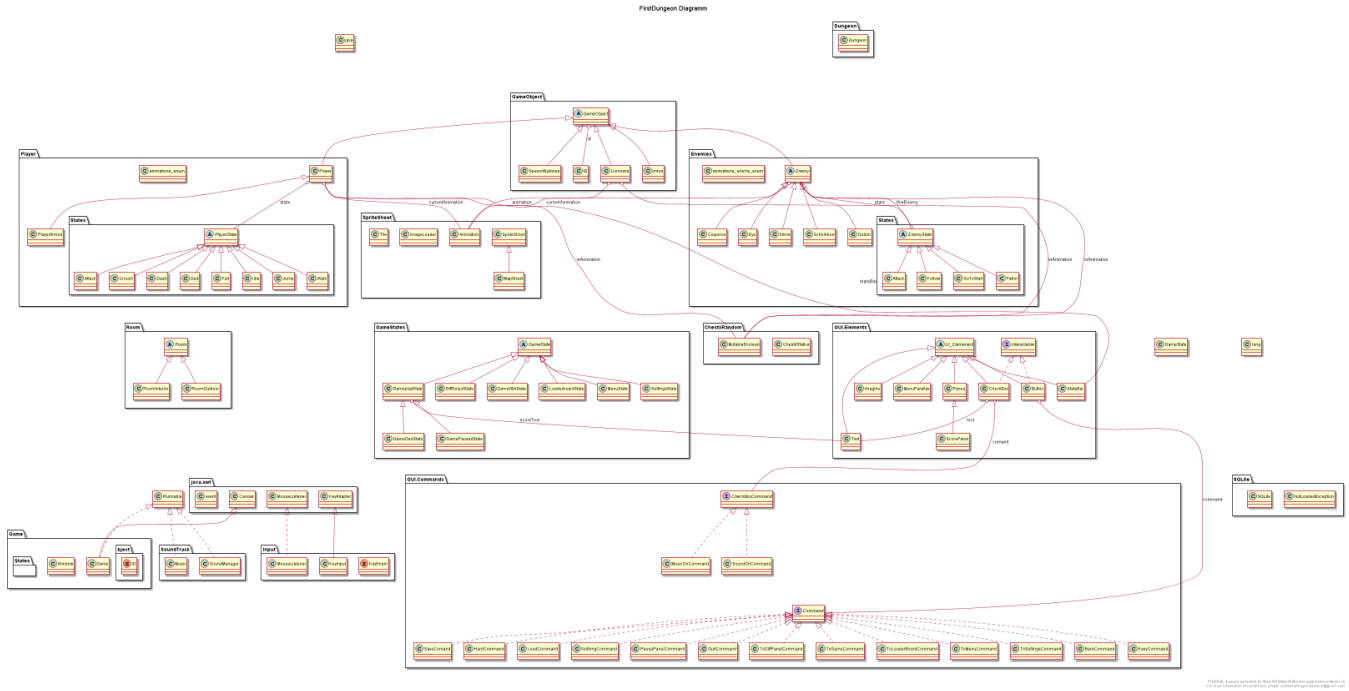
- *Normal*: Inamicii și capcanele iau o jumătate de inimă la atingere; navigarea camerelor va fi ceva mai ușoară și direct;
- *Hardcore*: Inamicii și capcanele iau o inimă la atingere; navigarea camerelor va fi ceva mai grea și întortocheată;

Descriere meniu :

- Meniu principal:
  - START: începe un nou joc
  - CONTINUE: continuă jocul început deja
  - SETTINGS: meniu cu setările jocului
  - QUIT: ieșire din joc

- Meniu de setări:
  - Efecte sonore pornite/oprite
  - Muzica de fundal pornită/oprită
  - Back: buton de întoarcere în meniul principal

Diagrama de clase:



Arhitectura:

Jocul are un thread principal pe care ruleaza loop-ul jocului, unde are loc toata logica acestuia și, de asemenea, mai are un thread destinat pentru coloana sonoră (care există într-o clasă Singleton) și există și o clasă de SoundManager care creează thread-uri într-un ThreadPool pentru a reda efectele sonore.

Loop-ul principal al jocului ruleaza funcțiile de render si tick de 60 de ori pe secundă, funcții care realizează partea de desenare și respectiv logică a elementelor jocului. Aceste funcții doar apelează cele 2 funcții din starea curentă a jocului. Starea jocului este implementată ca un șablon State, prin care se descriu stări ale jocului, precum starea de meniu principal, de meniu de setări, de joc etc.. Aceste state-uri au liste de elemente de interfață grafică pentru fiecare stare care sunt făcute visibile în momentul în care jocul intră în acea stare.

Elementele de interfață moștenesc toate o clasa abstractă UI_Element, dar sunt și unele care moștenesc o interfață numită Interactable. Aceste elemente sunt adăugate într-o listă a unei clase MouseListener care transmite, asemănător unui șablon Observer, evenimentele mouse-ului, precum apăsarea unui buton de pe mouse, unde, dacă mouse-ul se află deasuprea respectivului element, se întâmplă diverse acțiuni. Aceste acțiuni implementează o interfață numită Command, implementând șablonul Command, iar elementele de tip Interactable le efectuează la evenimente ale mouse-ului.

Jocul propriu-zis execută funcțiile de render si tick, atât pentru elementele de interfață, cât și pentru clasele Player și Dungeon, ambele fiind clase Singleton. Clasa Player gestionează tot ce ține de jucătorul principal, precum mișcarea, coliziunile cu harta, viața și altele. Miscarea jucătorului este gestionată tot printr-un șablon State. Jucătorul este inițial neînnarmat când intră în temniță dar când

găsește comoara primește o armă. Atunci instanța de singleton a jucătorului este schimbatăcu o clasă derivată din Player, care încarcă un nou sprite sheet si acum poate ataca inamicii.

Clasa Dungeon se ocupă cu generarea și controlarea temniței. Are o matrice de camere generate procedural pentru a crea o cale către camera comorii. La fiecare tick se realizează update doar la camera în care jucătorul se află. Fiecare cameră are o listă de obiecte din camera respectivă, precum inimi sau inamici, care sunt activi doar când camera în care se aflî este cea curentă.

Inamicii sunt toți derivați din clasa Enemy, care la rândul ei derivează GameObject. Ei funcționează ca un automat finit de stări simplu, implementat tot prin șablonul State. Aceștia patrulează într-o zonă până când hucătorul se apropie de ei, moment în care încep să îl urmărească. Când ajung lângă acesta, îl atacă. Dacă jucătorul iese din raza inamicului, acesta se întoarce în zona sa de patrulă.

Datele jocului, precum setări sau jocuri salvate, sunt introduse într-o bază de date cu SQLite cu ajutorul clasei SQL. Clasa SQL se ocupă cu tot ce ține de partea de SQLite pentru a salva si obține informatții din baza de date, apoi set-urile de informații rezultaet din query-uri sunt transmise mai departe în clasele care au nevoie de erspectivele informații. În cazul în care aceste informații sunt corupte sau nu corespund, se aruncă o excepție de tip NotLoadedException care este tratată diferit, în funcție de clasa care eșuează în a încărca respectivele date.