MONO-FMI

Proiect Metode Dezvoltare Software

1 Descrierea proiectului

Ideea proiectului este de a realiza o simulare a boardgame−ului "Monopoly" intr−o aplicatie software accesibila. Pentru realizarea acesteia se va folosi ca limbaj de programare principal C++, la care se adauga librarii aditionale pentru rendering(SDL). Sistemul de rendering nu este unul avansat care sa permita o interfata cu un design inovator, in schimb permite organizarea jocului la un nivel low-level pentru a implementa regulile jocului corect si eficient.

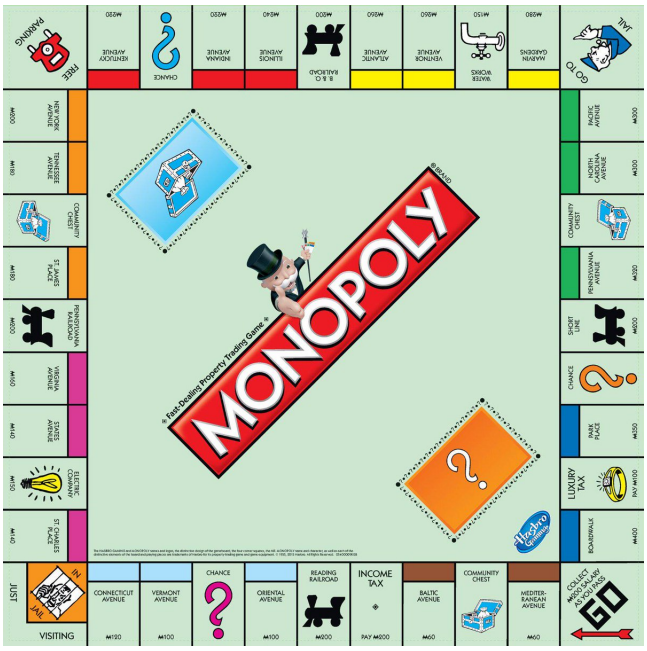

Figure 1: Harta de Monopoly folosita

(a) 1 (b) 5 (c) 3 (d) 6

Figure 2: Imaginile zarurilor

2 Organizarea proiectului

Proiectul va fi organizat in 2 parti, care vor fi dezvoltate in paralel: cea vizuala si cea de logica a jocului.

Partea vizuala va trebui sa urmareasca rendering-ul, miscarile pionilor pe harta, zarurile, organizarea butoanelor de buy, sell, trade, mortgage, meniul pentru jucator.

Partea de logica a jocului va trebui sa stabileasca si sa implementeze ce se intampla in momentul cand un jucator ajunge pe o proprietate, sa retina toate datele despre jucatori, sa creeze actiunile din momentul apasarii unui buton, a zarului si functionalitatea corecta a jocului (sa nu se dea de prea multe ori cu zarul, sa se organizeze pe ture).

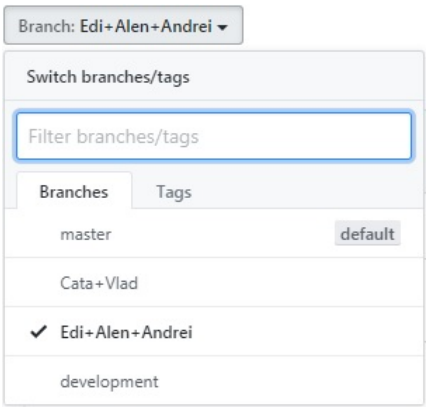

Figure 3: Branch−urile folosite pentru organizare

Pentru o organizare buna a developmentului, au fost folosite *GitHub*, *Sketchboard* si *Trello*.

Pentru a nu incurca progresul partii grafice cu cel al logicii jocului, am lucrat in 2 echipe pe 2 branch-uri diferite, urmand ca la momentele importante (saptamanal) sa dam merge la cele doua branch-uri in master.

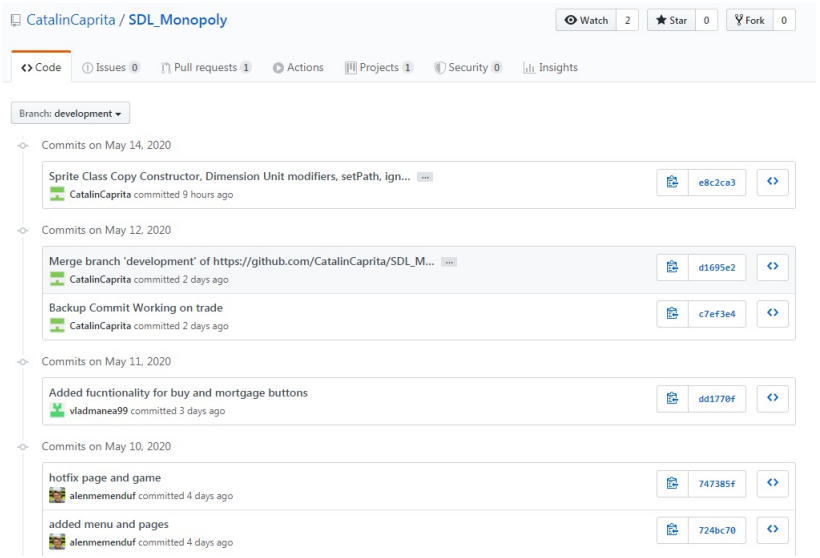

Figure 4: Cateva dintre commit-urile de pe GitHub

Codul din commit−uri a fost scris la nivel de echipa, pe Discord, urmand un sistem de "Pair Programming". Cand era nevoie de merge pe cele doua branchuri, organizam o intalnire la nivel de echipa tot pe Discord, unde stabileam ce este de facut in continuare, ce s-a realizat si ce probleme mai sunt de rezolvat, urmand sistemul de "Sprint".

3 Partea grafica

Pentru realizarea ferestrei in care ruleaza jocul a fost folosita biblioteca externa SDL.

Structurile sunt simple, la un nivel de baza pentru a permite organizarea lor, fara a da un design special aplicatiei, pentru a mentine implementarea functionalitatilor.

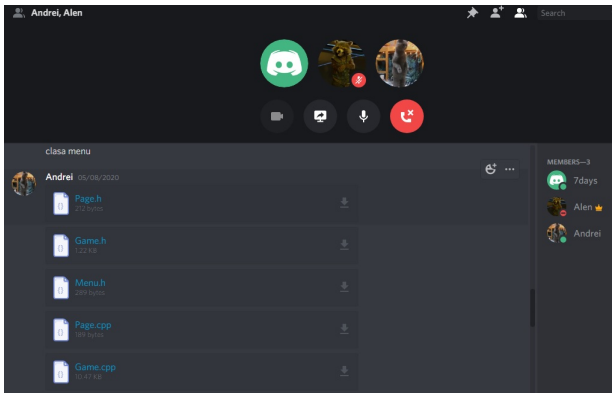

Figure 5: Conferinta pentru partea grafica pe Discord

Funcții utile pentru render sunt: SDL_RenderClear(renderer). Pentru rendering au fost create clase speciale (TextureMaker - folosita pentru a incarca si pentru a afisa imaginile in aplicatie).

4 Partea de logica a jocului - User Stories

Ideea de baza a jocului de Monopoly este desfasurarea succesiva pe ture, intre mai multi jucatori, pe baza "aruncarii" a 2 zaruri. Pentru a face cat mai sugestiva pentru utilizator aceasta experienta, am adaugat o animatie speciala pentru momentul "aruncarii" cu zarurile. Utilizatorul va da click pe cele doua zaruri, iar prin suprapunerea continua a mai multor sprite-uri, se va crea animatia de rotire a zarurilor, de 3 ori, pana se vor afisa valorile finale, iar pionul jucatorului se va muta corespunzator pe harta.

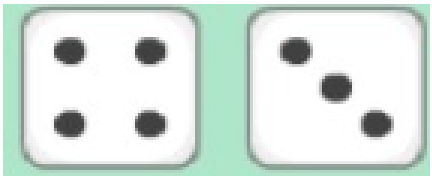

Figure 6: Zarurile

De asemenea, pentru a imbunatati experienta utilizatorului, ar fi nevoie de o interactiune cu aplicatia in momentul in care se incearca realizarea unei actiuni, iar pentru aceasta au fost adaugate

prompt-uri cu animatii (Monopoly Man) pentru a evidentia anumite actiuni care se intampla pe parcursul jocului (cumpararea unei proprietati, plata unei chirii).

Figure 7: Monopoly Man

Ca utilizator, principalele functii de care am nevoie in momentul jocului, sunt butoanele prin care pot lua decizii pe parcursul jocului - cumpara o proprietate, vinde o proprietate, adauga o casa pe o anumita proprietate, trebuie sa am acces la anumite statistici despre joc - cati bani mai am, cati bani mai au si ceilalti jucatori, pentru a planifica strategia pentru joc. Aceste feature-uri sunt baza jocului de monopoly si principalele User Stories.

Pentru o implementare organizata si explicita pentru utilizator, in dreapta hartii de Monopoly, am adaugat un meniu organizat in mai multe taburi, accesibile prin intermediul unor butoane, o singura pagina fiind vizibila la un moment dat. De asemenea, in meniu se gasesc si butoanele de buy, sell si end turn, pe care jucatorul le poate folosi doar in cadrul turei sale. Pentru a evita erorile de joc, s-au adaugat verificari ca un jucator sa nu isi poata incheia tura fara sa dea cu zarul, sau daca a dat o "dubla" sa continue sa joace tura.

La final, ca utilizator experienta jocului trebuie sa fie una continua, pionii sa fie mereu vizibili pe harta de Monopoly, la fel si casele

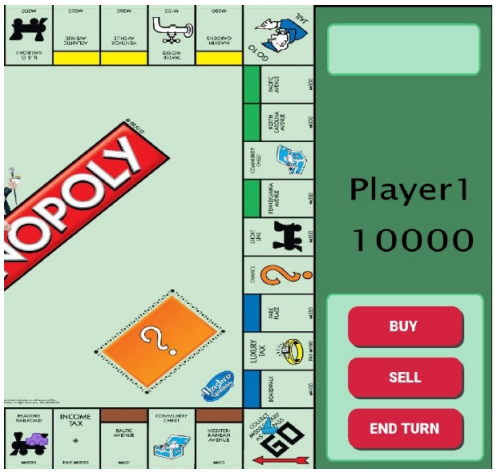

Figure 8: Meniul in partea dreapta

de pe anumite proprietati, feature-uri realuzate prin intermediul rendering-ului. Miscarea pionilor pe harta este secventiala, iar in momentul stationarii pe o proprietate, daca nu este deja cumparata, este dat focus pe acea proprietate, facilitand accesul utilizatorului la informatiile despre acea proprietate (pret, chirie), pentru a putea lua o decizie despre cumpararea ei mai usor si pentru a mentine experienta jocului.

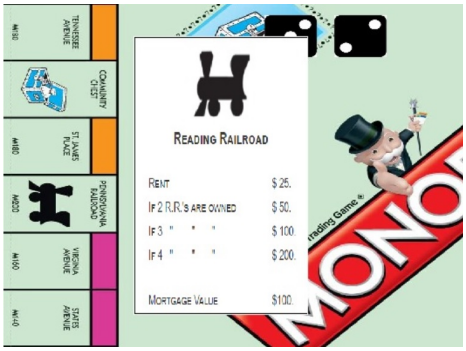

Figure 9: Exemplu de zoom pe proprietati

5 Diagrama UML

Diagrama UML va contine schema legaturilor dintre clasele principale si cateva explicatii despre functionalitatile acestora si interactiunea utilizator-aplicatie.

Principalele legaturi au fost simbolizate prin sageti.

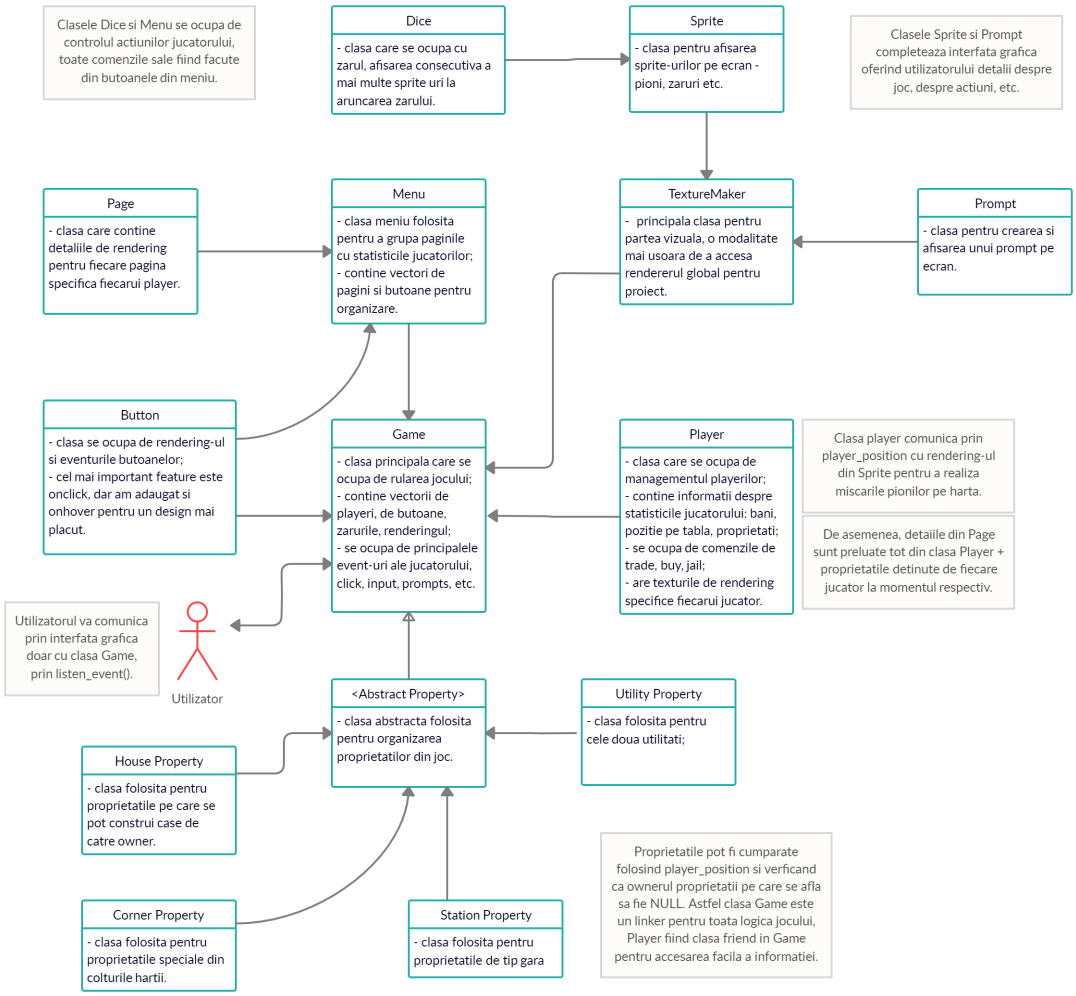

Figure 10: Diagrama UML

6 Source control, refactoring, bug reporting

Pentru versionarea proiectului, cum am mentionat si mai sus, s-a folosit in principal *GitHub* cu mai multe branch-uri pentru development in paralel.

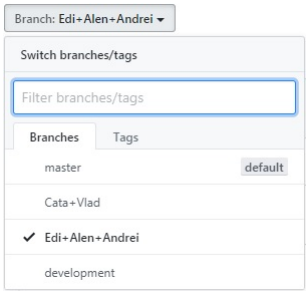

Figure 11: Branch−urile folosite pentru organizare

Codul din commit−uri este in mare parte development, iar in momentul in care apareau erori sau bug-uri in proiect, se organiza o lista cu TO-REVIEW pe *Trello*.

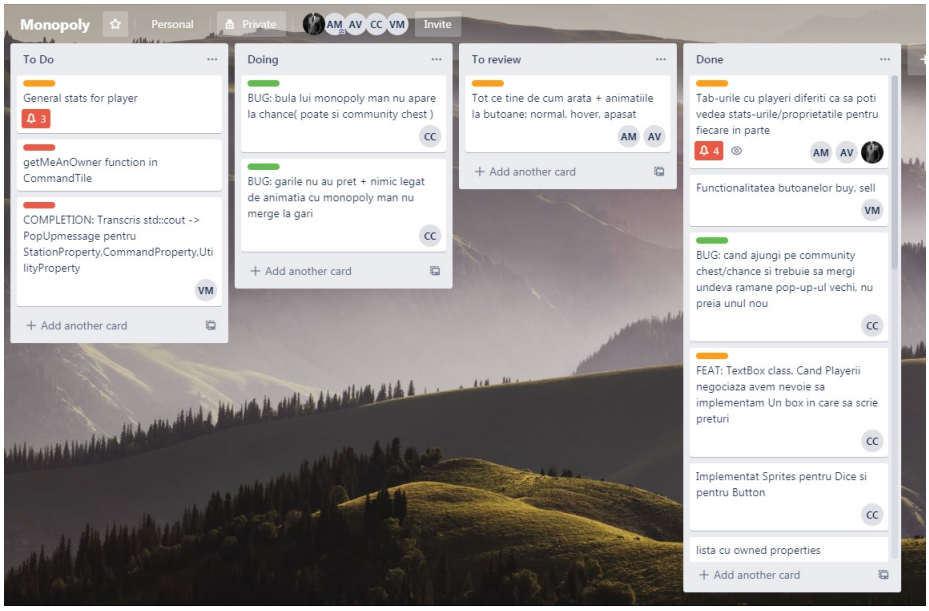

Figure 12: Bug reporting pe Trello

Pentru fiecare bug din lista a fost nevoie de refactoring la cod, de exemplu la inceputul developmentului, jucatorii nu continuau tura daca se dadea o dubla cu zarul, asa cum ar fi trebuit sa se intample, sau in momentul cand erau la inchisoare si dadeau o dubla nu ieseau din inchisoare, anumite animatii nu mergeau cum ar fi trebuit, miscarile pionilor pe harta erau sacadate, etc.

7 Functionalitati

Prima interactiune a utilizatorului cu interfata grafica este fereastra de introducere a numelor jucatorilor. Numarul jucatorilor va fi setat in functie de cate nume au fost introduse de utilizator in fereastra. Numele nu trebuie introduse in casute consecutive. Se pot alege intre 1 si 4 jucatori.

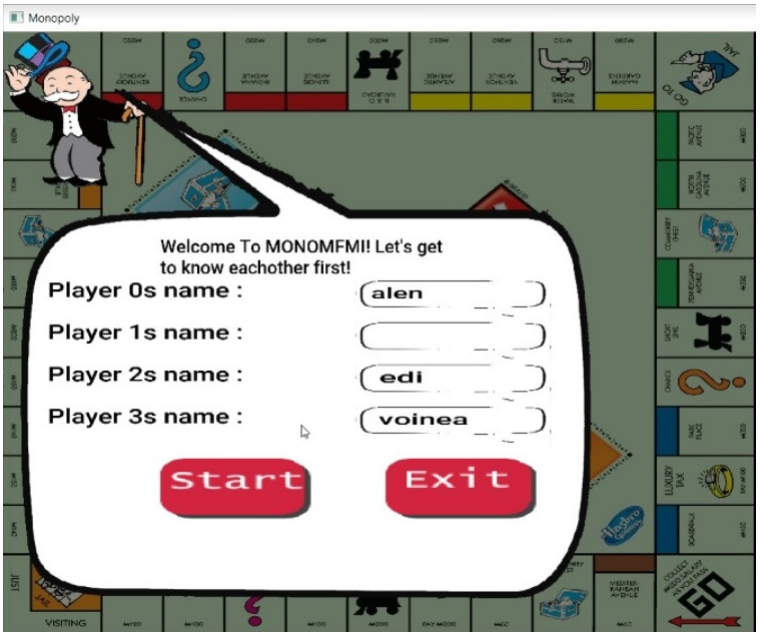

Figure 13: Fereastra pentru alegerea numelor

Functionalitatile principale de buy si sell sunt implementate pentru utilzator cu ajutorul butoanelor din meniu. De asemenea, tot in partea de jos a meniului se gasesc zarurile si butonul de end turn folosit pentru a incheia tura unui jucator.

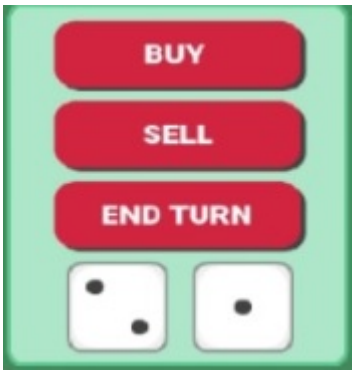

Figure 14: Butoanele din meniu

Pentru statisticile fiecarui jucator, in meniu au fost create mai multe pagini, in functie de pionul asociat fiecarui jucator. In fiecare pagina se gaseste numele jucatorului, banii si proprietatile pe care acesta le detine la momentul respectiv.

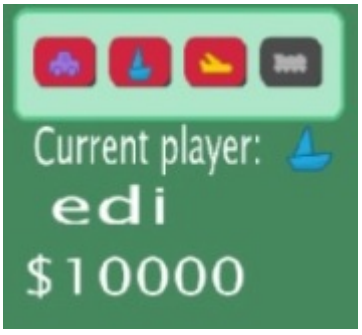

Figure 15: Statistici jucator

In momentul in care un user ajunge pe o proprietate nedetinuta de alt player, acesta primeste optiunea de a o cumpara, in momentul in care se face zoom pe aceasta si se apasa butonul de buy din meniu.

Figure 16: Actiunea de cumparare

Jocul se continua succesiv cu fiecare jucator, dandu-se cu zarul generat random, pionii se muta secvential pe harta in functie de zar si cumpara proprietatile in functie de deciziile utilizatorului.