

Bubble Trouble
Ionita Bogdan Marian
1206B

1.Povestea jocului:

Jocul prezinta incercarea protagonistului nostru de a iesi dintr-un labirint făcut din mai multe camere cu capcane. Astronautul nostru s-a ratacit pe o planeta îndepărtată în care viața lui este pusă în pericol. Fiecare camera prin care acesta trece conține mai multe bile dintr-un material necunoscut care l-ar putea omora prin simpla atingere. Pentru a reuși, eroul și-a modificat costumul cu o sageată specială, săgeata care-l poate ajuta dar și încurca. Aceasta arma reprezinta singurul sau ajutor, o sageată uriașă construită dintr-un material necunoscut, găsit la intrarea în labirint. Labirintul este astfel construit încât ușa camerei se deschide doar atunci cand toate bilele din camera au fost distruse. Astfel, protagonistul este obligat să-și folosească arma construită, care are puterea de a înjumătăți bila în 2 parti egale, pentru a încerca să ajungă înapoi la nava și să se întoarcă acasă.

2.Prezentare joc:

Campanie pentru un singur jucător, în care jucătorul trebuie să distrugă toate amenințările din camera în care se află pentru a continua. Camera conține un sistem de autodistrugere, iar cand jucatorul nu reușește să iasă din camera într-un anumit interval de timp el va muri. Jocul este castigat daca trece prin toate camerele.

3.Reguli joc

Jocul implica caracterul care se poate mișca stanga-dreapta și poate sări în interiorul camerei. La intrarea în camera, un timer se activează, iar protagonistul trebuie să distrugă toate bilele înainte ca tavanul să cedeze. Arma lui funcționează doar în poziție verticală, astfel sageata aruncată de

costum loveste doar ce este deasupra lui. O bila lovită se împarte în 2. Prin mai multe lovituri aceasta devine prea mică pentru a fi observabilă și dispare. Uneori, cand o bila este distrusă, aceasta va arunca un “power up”. Dacă jucătorul alege să-l atingă, costumul lui se va îmbunătăți, iar acesta va arunca săgețile mult mai rapid pentru o perioada scurta de timp. De asemenea mai există un “power up” care-i oferă viteza de mișcare, utilă pentru evitarea bilelor. Jucătorul are la dispoziție 3 vieți, iar de fiecare dată cand este atins de o bila nivelul în care acesta se află se resetează. Pentru a castiga și a reveni la nava, jucătorul trebuie să treacă prin toate camerele fără a-și pierde viețile. Odată ce toate cele 3 vieți au fost pierdute, jocul se încheie și se începe din prima camera.

4. Personajele jocului

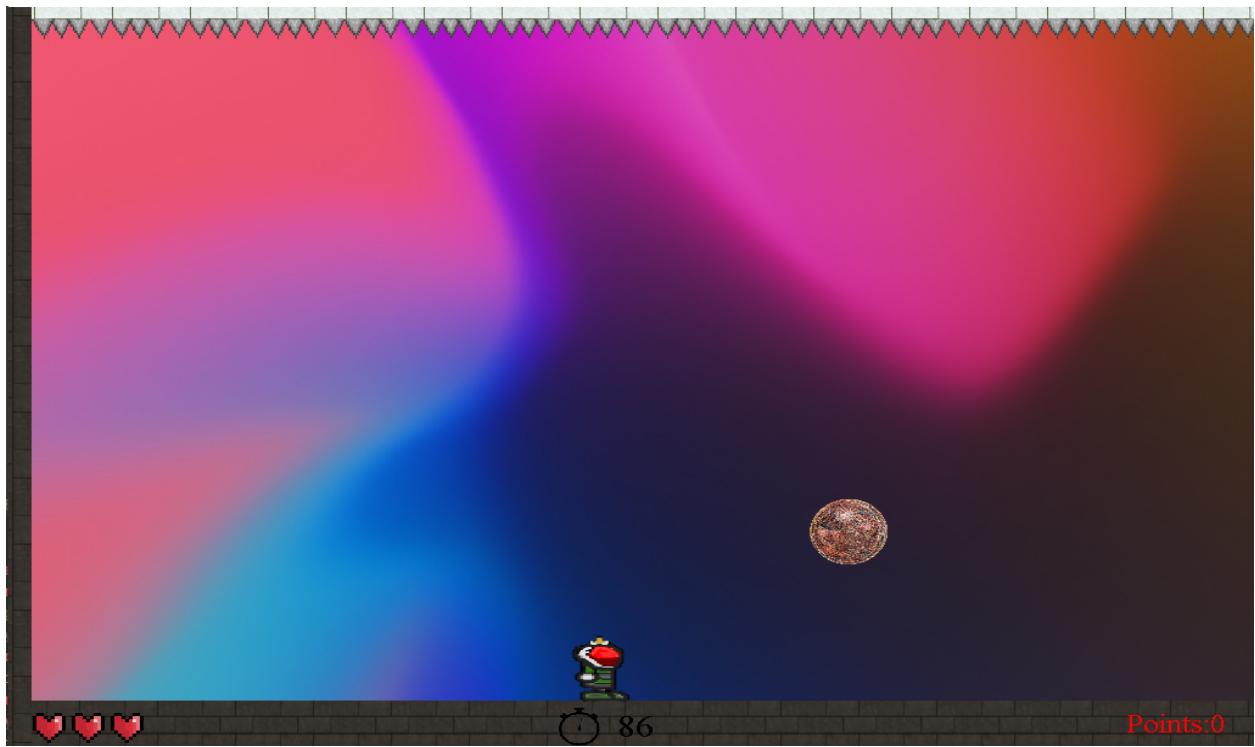
Astronautul este protagonistul și singurul om al jocului. El este un inginer, care încearcă să-și salveze viața pentru a se întoarce acasă. Datorită intelectului său, acesta a reușit să creeze un costum care-l poate ajuta în călătoria sa.

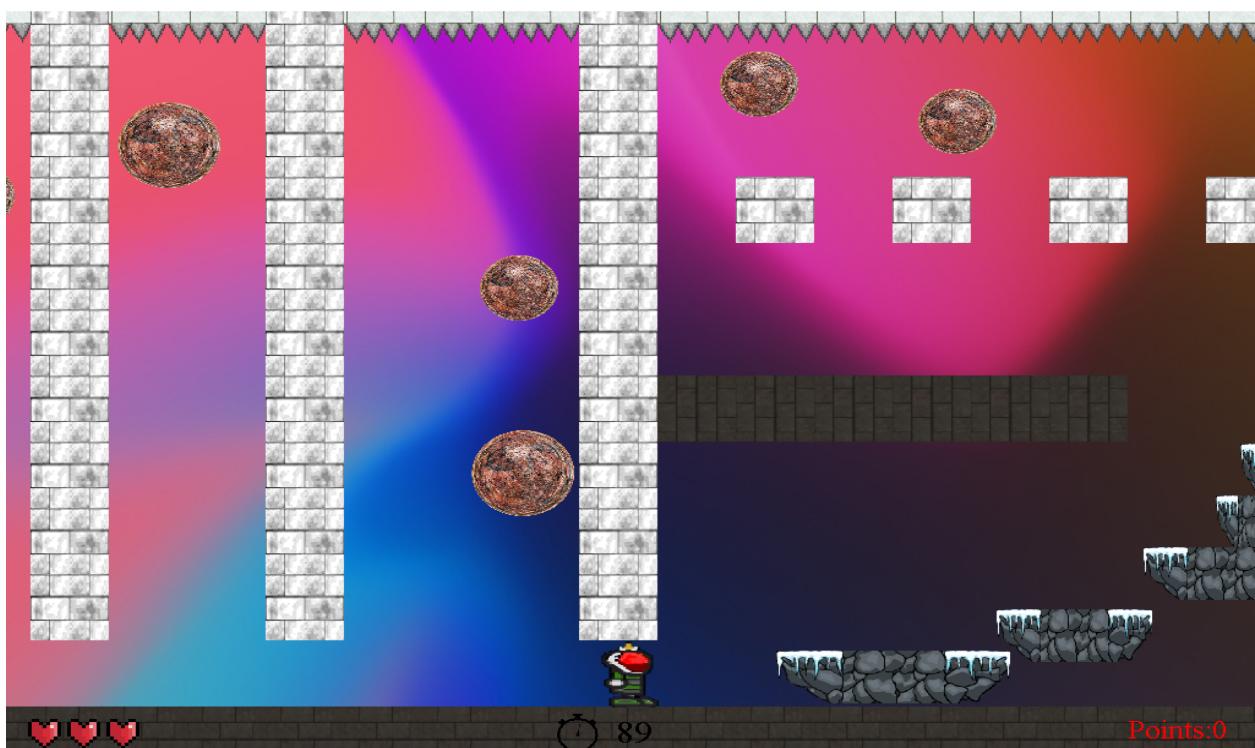
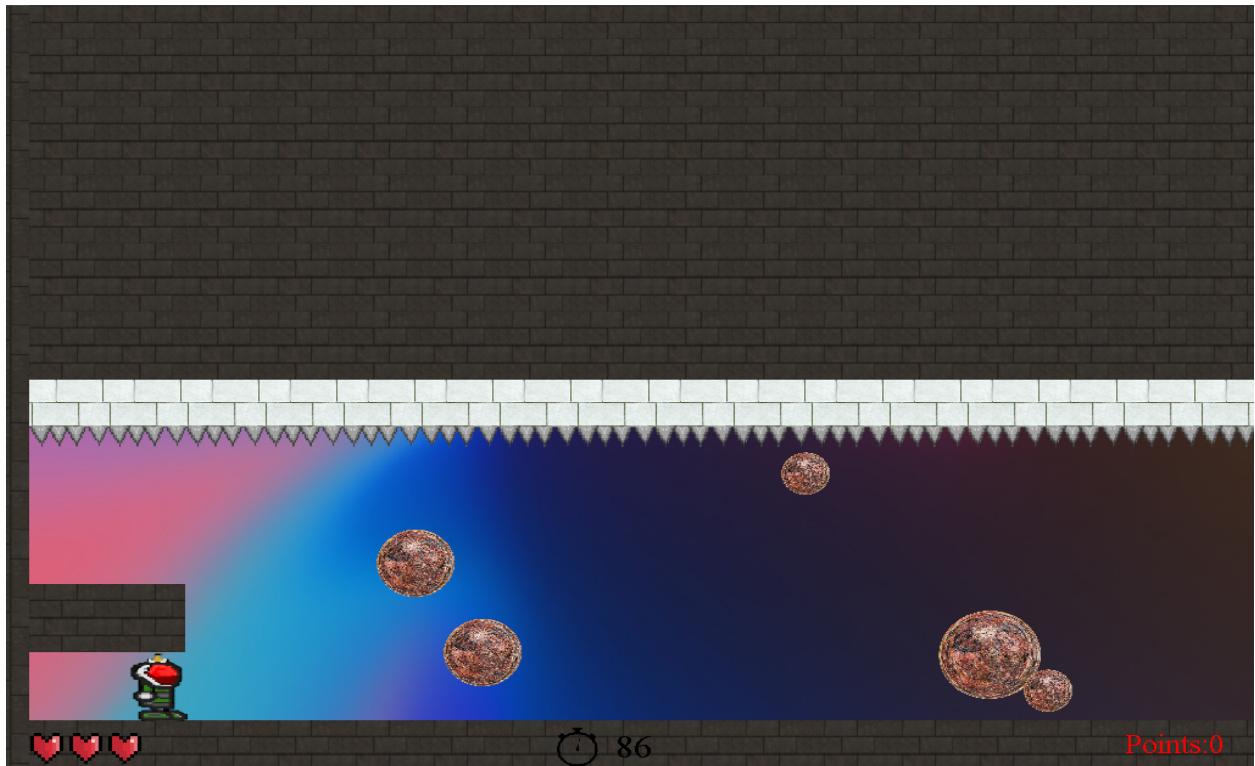


5. Tabla de joc

-Componente pasive: podeaua, tavanul cu tepi, peretii și platformele

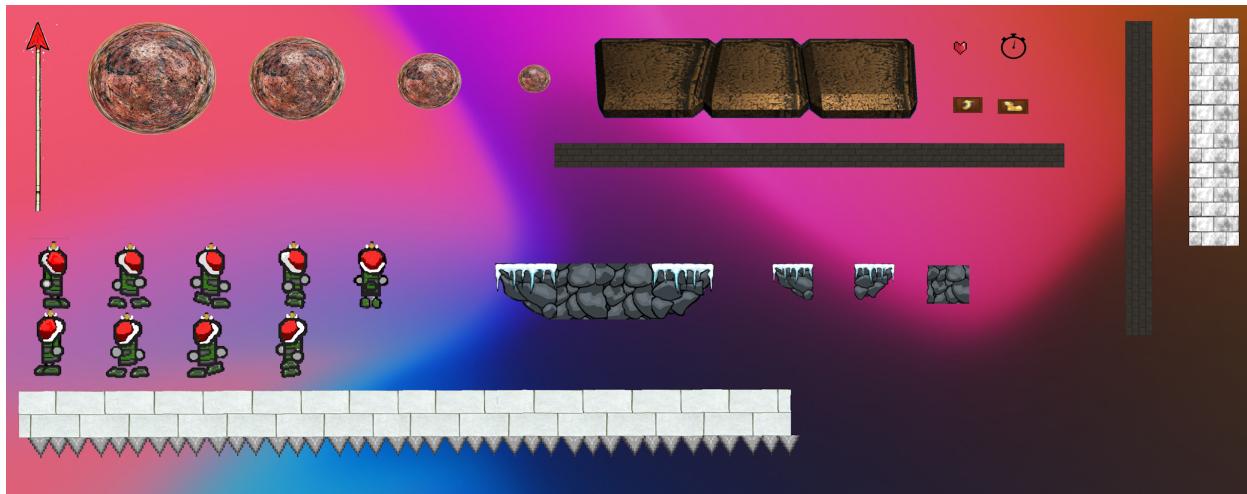
-Componente active: mingile care se lovesc de pereti, eroul care poate fi controlat, power ups-urile care cad cand o bila este distrusă și dispar după un interval de timp.





Mecanica jocului: jocul se va controla prin w/a/d sau prin săgeți pentru a muta caracterul care se misca stanga,dreapta și sare. Prin apăsarea tastei space jucatorul va arunca o sageata verticala cu scopul de a nimeri inamicul.

6.Game sprite



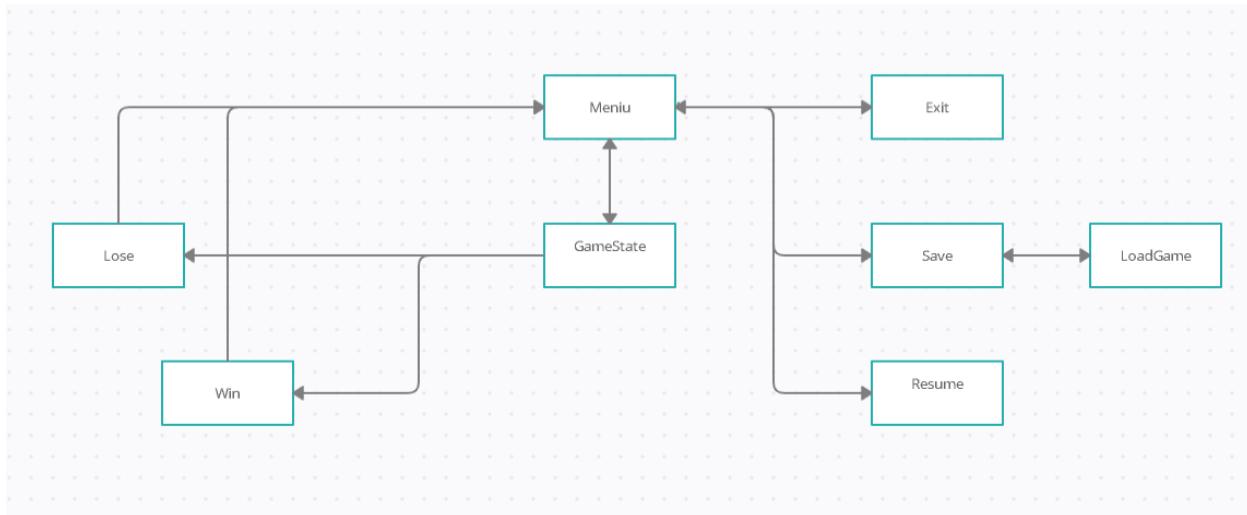
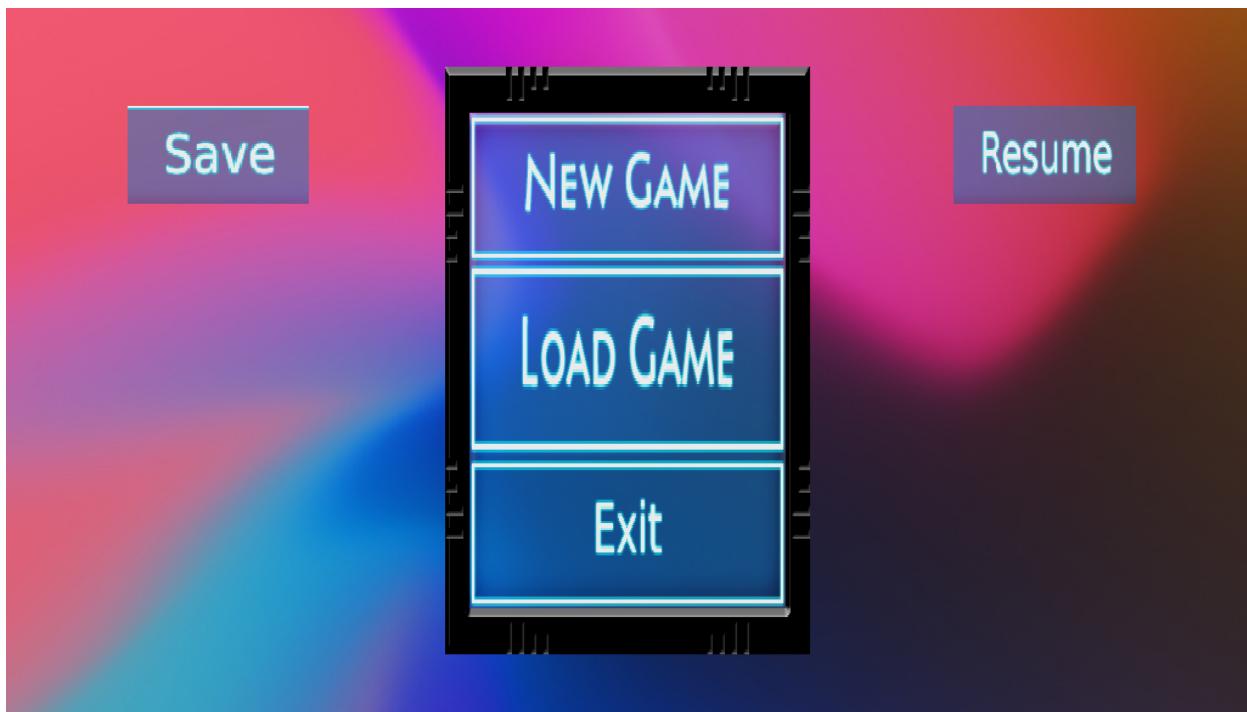
O parte din texturi au fost gasite pe <https://opengameart.org/> si modificate.

Descriere nivel: Jocul va avea 4-5 nivele, dificultatea crescand treptat de-a lungul jocului. Dificultatea in acest caz este reprezentata de numarul de inamici și de ingustarea camerelor. Daca primul nivel este o camera complet goala, cu mult spatiu pentru jucator astfel incat sa se fereasca cu usurinta, ultimul nivel va fi delimitat prin mai mulți pereti, fiecare “mini încăpere” avand un număr de inamici prinși între acele ziduri. Cu cat o parte de încăpere cu inamici este mai ingusta, cu atat jucătorului îi va fi mai greu sa se ferească de coliziunea cu bila/bilele.

Meniul va contine următoarele butoane:

- New Game--porneste un joc nou, de la primul nivel, indiferent de nivelul la care s-a ajuns la ultima încercare.
- Load Game--reporneste jocul de la ultimul nivel salvat
- Save--Salvează nivelul curent,scorul și numărul de vieți intr-o tabela SQL
- Resume--Reporneste nivelul activ curent in caz ca s-a apăsat butonul esc

-Exit--paraseste jocul.



Scor:--Jucătorul va începe cu un scor=0, iar acesta va crește cu 2 puncte pentru fiecare bila mare distrusă și cu un singur punct pentru bila finală, care nu se mai poate micșora. Acesta va fi salvat într-o baza de date când jucătorul pierde cele 3 vieți sau termină jocul.

Mecanica jocului(adaugare)--Odată ce toate nivelele au fost completate, jocul se va termina iar utilizatorul va fi trimis din nou în meniu unde va vedea mesajul că jocul a fost câștigat. Aici el va fi întrebat la

consola ce nume are pentru a i se salva scorul în baza de date. În cazul în care jucătorul pierde acesta va primi mesajul ca jocul a fost pierdut și are de ales dacă salvează scorul sau nu.

SQLClass- Aceasta este o clasa cu toate metodele statice care ne ajuta să inserăm și să scoatem din baza de date fără a ingreuna codul. Clasa contine metoda `returnMapMatrix` care citește un tabel și-l transforma într-o matrice necesara pentru a încarca harta. O alta metoda care ne ajută este `addScore` care primește ca parametri un nume, scor, nivel și timp și inserează aceste informații în tabela de scoruri. Dacă numele există deja, informația veche va fi suprascrisă. De asemenea tabela salvează informații care ne ajuta să salvăm și să încarcăm jocul de la ultimul nivel completat.

SQLite--Jocul realizează o conexiune cu SQLite în care se găsesc 6 tabele diferite. 4 tabele conțin harta celor 4 nivele prezente în joc. Tabela cu numele "Scor" salvează numele, nivelul, numărul de vieți rămase și timpul jucat al jucătorului atunci cand acesta termină jocul. Jucătorul care pierde este întrebăt înainte dacă vrea să salveze scorul. De asemenea mai există o tabela Save care conține informațiile necesare salvării jocului pentru a reveni ulterior la nivelul curent.

Design pattern-uri:

Singleton--Am folosit singleton pentru clasa Game deoarece doar o instanță a jocului trebuie să existe la un moment dat.

Factory--Am folosit acest pattern pentru a crea entitățile care urmează să fie adăugate în EntityManager. Cu ajutorul unui switch se creează o mingă sau un tile.

DAO(Data acces object) pattern--L-am folosit în EntityManager pentru a gestiona mai ușor entitățile prezente în joc. De asemenea este mai ușor să verificam când nivelul a fost completat cu ajutorul lui.

Flyweight pattern--L-am folosit pentru a reduce semnificativ memoria folosită de joc când acesta încarcă tile-urile. Tile-urile se repetă de-a lungul jocului și dacă am încarca pentru fiecare tile asset-ul specific memoria folosită ar deveni foarte mare.

Facade--Clasa Game implementează pattern-ul Facade. Aceasta este apelată în clasa Launcher și ascunde restul codului pentru o interfață mai simplă.

