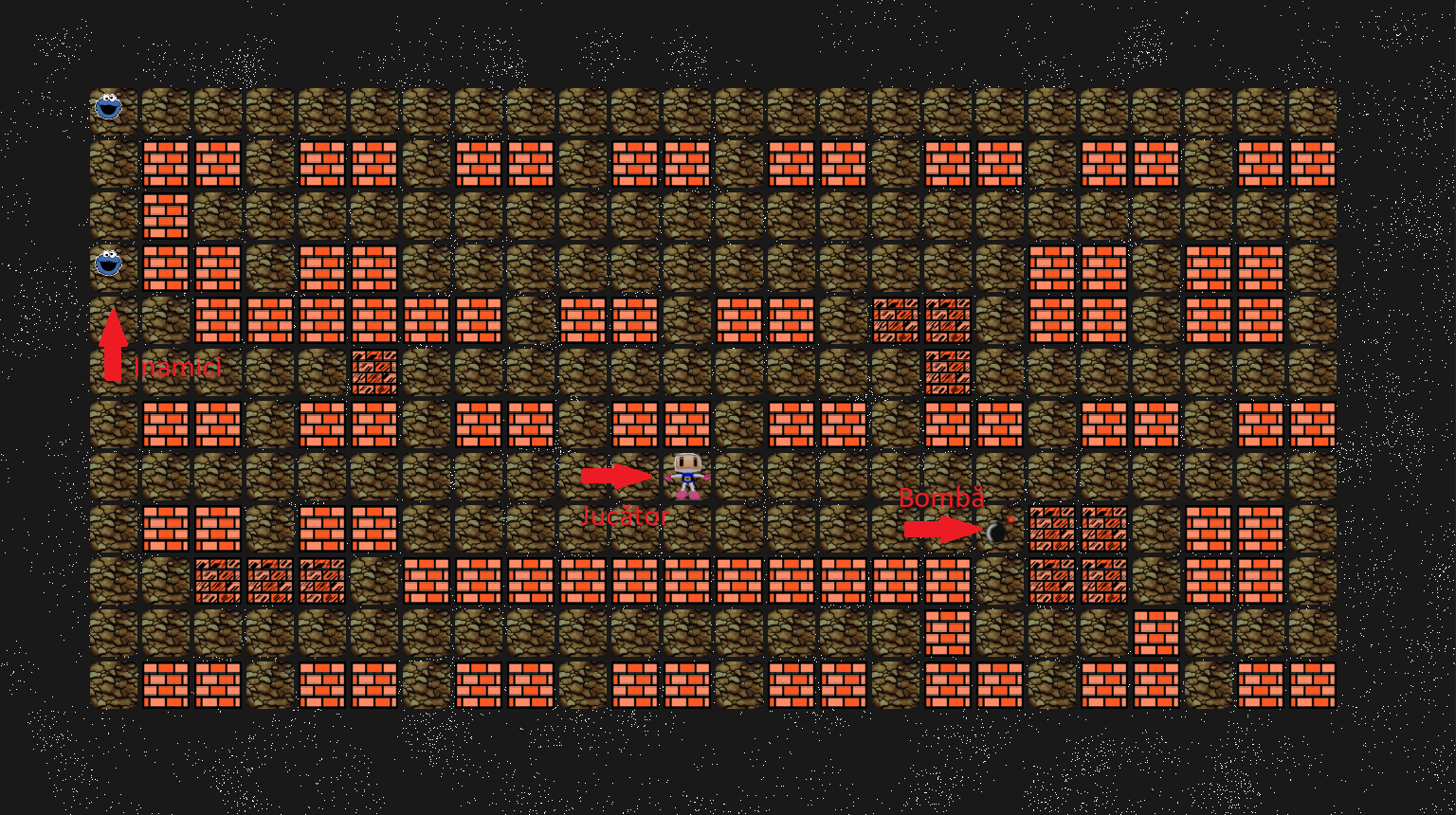
# Bomberman

Echipă:

* Bardaș Denis Adelin
* Brănaci Șerban Mihai
* Chindea Cosmin Mihai

Descriere

Echipa noastră a ales să realizeze jocul Bomberman. Scopul jocului este ca jucătorul să elimine inamicii prezenți pe hartă și să avanseze la nivelul următor. El folosește bombele pentru a sparge blocurile care stau în calea sa. Totodată ele folosește aceleasi bombe pentru a distruge adversarii.



Harta unui nivel al jocului. Am evidențiat principalele elemente ca jucătorul, bomba și inamicii.

UI

Pentru realizarea acestui proiect am folsit framework-ul MonoGame A red and black logo

Description automatically generated pentru partea de afisare a unei interfețe pentru utilizator. El ne-a ajutat cu organizarea și afișarea elementelor grafice care constituie mediul de joc pentru utilizator.

A screenshot of a computer

Description automatically generated

*Proiectul responsabil cu afisarea interfeței este de tip MonoGame Desktop Application si are o structura specifica MonoGame:*

*În folderul Content se afla texturile, fonturile si celelalte asset-uri folosite pentru interfata grafica*

*LogicExtentions contine metode de extensie ale interfetelor din DLL-ul cu logica aplicatiei, pentru a putea afisa fiecare element si updata logica aferenta UI-ului*

*UIHelpers contine diverse clase specifice MonoGame create pentru a respecta principiile clean code si pentru a face cod reutilizabil si mentenabil*

*Clasa Bomberman.cs este "managerul" partii de UI, centralizand informatiile si instructiunile specifice UI.*



*Sprite-urile folosite pentru afișarea jocului.*

Framework-ul Monogame dispune de funcții și clase speciale pentru afișarea pe ecran a resurselor. Noi am folosit aceste clase și funcții în clasa Bomberman.cs care administrează interfața grafică a jocul. Această clasă se găsește în cadrul proiectului BombermanMONO, care este proiectul de front-end al programului. In această clasă am folosit clase specifice precum GraphicsDeviceManager, SpriteBatch și Vector2. Mod de lucru al acestei biblioteci este că ea prima dată încărcă resursele din foldere. Apoi după ce aceste resurse au fost încărcate folosim funcția Draw() pentru a afișa elemtele pe ecran.A screen shot of a computer program

Description automatically generated

A computer screen shot of text

Description automatically generated

Funcțiile pentru încărcarea resurselor.

Back-End

Codul proiectului a fost scris în limbajul C# și IDE-ul Visual Stududio. Pentru diversele necesități am folosit frameworkul MSTest pentru partea de testare.

În cadrul proiectului am lucrat în manieră OOP și am respectat normele de denumire din C#, având un cod cât mai curat și mai ordonat. Am folosit design pattern-urile Observe, Factory și Strategy.

Soluția de Visual Studio noastră cuprinde 3 proiecte distincte, pentru fiecare componentă jocului: BombermanBase pentru logică, BombermanMONO pentru partea de UI și BombermanTests pentru partea de testare.

A screenshot of a computer

Description automatically generated

*Ierarhia proiectului cu cele 3 componente:*

*BombermanTests,*

*BombermanBase*

*și BombermanMONO*

*Partea de logica a aplicatiei este continuta intr-un proiect de tip DLL. Pe scurt, in acesta avem:*

*Clasele principale IEntity, ITile, ITileMap, IBomberman, care sunt elementele de baza ale aplicatiei. Aceste interfete sunt publice, iar ele sunt concretizare in clase interne.*

*Entitatile pot fi de doua tipuri: Player sau Enemy. De aici reiese si nevoia aplicarii design pattern-urilor Factory si Strategy. Factory initializeaza Entity-ul in functie de tip, Player sau Enemy, in timp ce strategy selecteaza cum se actioneaza la apelarea metodei Move*

*Pe langa acestea, se pot remarca diverse principii si implementari specifice OOP, cum ar fi polimorfismul, interfetele sau incapsularea*

*IBomberman.cs este clasa principala, concretizata prin Bomberman.cs. Aceasta trebuie instantiata si utilizata pentru a crea un joc.*

A screenshot of a computer program

Description automatically generated

Design Patterns

Design pattern-urile folosite în proiectul nostrum sunt:

* Observer
* Factory
* Strategy

Factory - acest design pattern a fost folosit la crearea entitățior de Player și Enemy. Aceștia sunt clasa Entity configurată cu design patternul Factory în funcție de nevoi.

A black background with white text

Description automatically generated

A screen shot of a computer

Description automatically generated

Factory Design Pattern



Inițializarea player și enemy

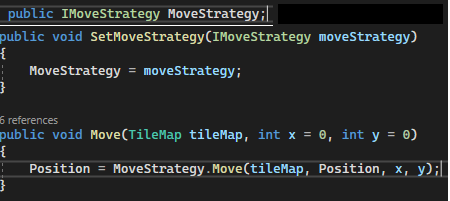
Strategy – acest design pattern a fost folosit pentru separarea tipului de miscare a entităților player și enemy care au restricții diferite pentru deplasare.

A black background with white text

Description automatically generated

A computer screen with text on it

Description automatically generated



Implementarea interfeței IMoveStrategy și a celorlalte două mecanisme

Observer - folosit pentru a notifica proiectul MonoGame asupra diverselor evenimente care apar odata cu derularea jocului. Astfel, avem interfata IBombermanObserver care este implementata de clasa principala din Mono, Bomberman.cs.

A screen shot of a computer code

Description automatically generated

A screen shot of a computer screen

Description automatically generated

Clasa Observable este Bomberman.cs, care implementeaza functii specifice, precum adaugarea si scoaterea subscriberilor, respectiv notificarea sunscriberilor in functie de eveniment

A screenshot of a computer program

Description automatically generated

Unit Testing

Pentru Unit Testing am folosit frameworkul MSTests.

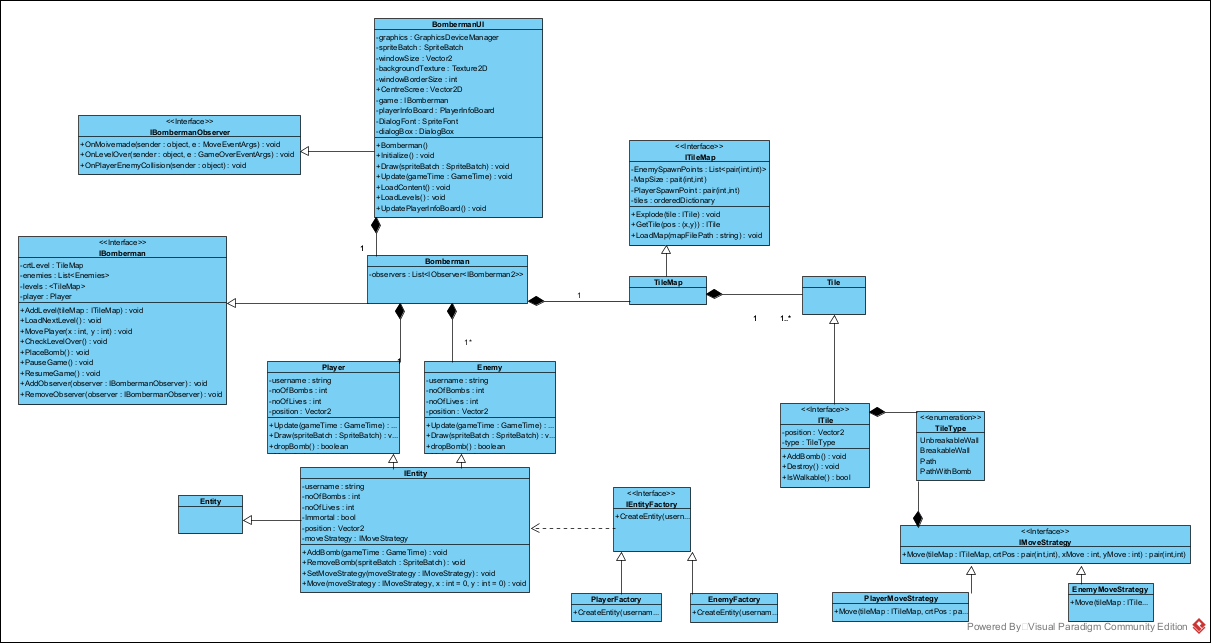
În cadrul unit testelor am verificat ca codul scris să fie corect din punct de vedere logic. Am vrut să fim siguri ca fiecare metoda scrisă își îndeplinește scopul. Am început prin a face teste simple, de exemplu, testarea că inițializarea entităților se face corect în cadrul testului TestEntitiesInitialization().

A computer screen with text

Description automatically generatedÎn continuare am verificat ca funcționalitățile jucatorul să funcționeze corect în harta definită la începutul jocului prin teste ca TestPlayerMoveOutOfMap(), TestPlacingBomb() și LosingLife().

În următorii pași vom testa funcționalitățile hărții și modul în care jucătorul interacționează cu ea. La fel programul de teste a cuprins și funcționalități mai complexe ale proiectului și modul de relaționare al proiectelor între ele, unde am folosit mai mult decât teste simple.

Diagrame

Class

Aici este prezentată diagrama pentru clasa de logica Bomberman. Pentru ea există o interfață IBomberman care o mostenește. Între clasa Bomberman și celelalte clase există relație de compoziție.

A diagram of a diagram

Description automatically generatedUse Case

Diagrama prezintă cum jucatorul interacționează prima dată cu meniul programului și apoi cu partea de joc. Adversarul este un actor virtual de aceea el poate interacționa doar cu partea de joc.

A diagram of a game

Description automatically generatedActivity

Diagrama de activități prezintă la scară largă toate starile prin care trece jucatorul, interfața grafică și jocul. Prima dată jucatorul deschide jocul. Interfața afișează meniul de start unde jucatorul introduce numele. Cu acea informație se inițializează jocul. Jocul trece prin schimburi de informație cu jucatorul iar la final există 2 stări posibile, cea de câștig a jocului și de pierdere a jocului.

A diagram of a game

Description automatically generatedState Diagram

Diagrama de stări prezintă starile jocului: în meniu, desfășurarea jocului, joc terminat și joc câștigat sau pierdut.

Sequence A diagram of a computer process

Description automatically generated

Aici evidențiem cum din punct de vedere temporal se desfășoară acțiunea în program. Jucătorul trimite un input interfeței. Această îl transmite mai departe către partea de logică. Aici se fac modificările necesare iar apoi ele sunt trimise către interfață pentru a fi afișate utilizatorului.

Communication

A diagram of a user interface

Description automatically generated

Diagrama de comunicare arată schimbul de informație de la jucator la cele 2 componente ale programului.