**Memory Game**

**2.09.2024**

|  |  |  |
| --- | --- | --- |
| **Îndrumător:** |  | **Student:** |
| **dr. ing. Daniel Morariu** |  | **Popa Denisa**  **223-1** |

**Istoric Versiuni**

|  |  |  |  |
| --- | --- | --- | --- |
| **Data** | **Versiune** | **Descriere** | **Autor** |
| 17/12/2023 | 1.0 | Introducerea pozelor care se afla pe cărți a fost o provocare și totodată o problema din cauza faptului ca nu știam cum sa le introduc și să mă folosesc de ele într-un mod cât mai eficient pentru a corespunde cu cele 32 de cărți din jocul de Memory.  La început, am mers pe varianta cea mai puțin eficientă prin care atribuiam fiecărei cărți cate o imagine din resurse. De aici, a rezultat un calup imens de cod. În momentul în care am ajuns la partea în care trebuia sa asignez și evenimentul de clic, am realizat ca e o problema în modul în care am gandit jocul până în momentul respectiv. Astfel că am creat un vector de imagini în care am inclus toate imaginile din resurse și ulterior, încă un vector de imagini, de dimensiune dubla fata de vectorul precedent. Scopul celui de-al doilea vector era sa duplice imaginile în așa fel încât să existe aceeași poză de două ori(pentru carte, respectivul perechea ei).Astfel, am ajuns la un mod mult mai eficient de utilizare a imaginilor în proiectul meu. | Eu |
| 26/12/2023 | 1.1 | Evenimentul de click a reprezentat a doua problema. Acesta funcționa corespunzător. Mai exact, se permitea să se dea click pe o carte din interfața grafică, însă la apăsarea uneia se întorceau toate. Ideal, trebuia să se întoarcă doar cartea selectată. Ulterior mi-am dat seama că atribuiam evenimentul de click unei alte funcții (GameArea \_Load) în locul funcției de ClickCard. | Eu |
| 02/01/2024 | 1.2 | O dată mutat evenimentul de click în funcția destinată acestui scop, am întâmpinat probleme în ceea ce privește sincronizarea interfeței grafice cu codul scris de mine. Acest fapt s-a datorat din cauza faptului ca nu am creat propriu-zis în cod PictureBox-urile care urmau sa îmi servească drept cărți. Acestea au fost aduse pe interfața grafica doar prin procesul de Drag&Drop.  Ulterior, am instanțiat un obiect de tip PictureBox în codul interfeței grafice, căruia i-am stabilit acolo, dimensiunile, denumirea, locația, evenimentul de click, tag-ul etc. Astfel, am putut sincroniza mult mai ușor tot ceea ce ținea de cod cu interfața grafică și am putut atribui obiectului meu de tip Card direct acolo PictureBoxul și cardID-ul (tag-ul). | Prietena |
| 05/01/2024 | 1.3 | Pe final, când jocul funcționa corespunzător am observat faptul că dacă o pereche este deja descoperită, aceasta se poate compara în continuare cu alte cărți. Lucru care nu ar trebui sa se întâmple. Mai în detaliu, dacă două cărți erau la fel, acestea rămâneau întoarse pe tabla. Dacă se selecta din nou una dintre ele și o alta carte, acestea se comparau, și se întorceau, dat fiind imposibilitatea de a fi la fel. Astfel rămânea pe tabla o singura carte întoarsă. Rezolvarea acestei probleme a constat prin adăugarea unei liste de obiecte Card care sa retina cartile care au facut deja pereche. | Prieten |
| 18/02/2024 | 1.4 | La prezentarea trecuta, jocul nu s-a deschis pe calculatorul facultatii deoarece am selectat versiunea gresita, asfel ca a trebuit sa ii schimb versiunea | Eu |
| 15/07/2024 | 2.0 | Am schimbat logica functionalitatii de baza a jocului, adica compararea cartilor. Acestea se comparau dupa imagini, intr-un mod mai dificil. In prezent cartile se compara dupa numere ( id-ul lor mai exact), fiecarei carti coresounzandu-i un numar de la 1 la 16. | Eu |

**Cuprins**

**ISTORIC VERSIUNI** 2

**CUPRINS 4**

**1** **SPECIFICAREA CERINȚELOR SOFTWARE 5**

**1.1 Introducere................................................................................................................................ 5**

1.1.1 Obiective 5

1.1.2 Definiții, Acronime și Abrevieri 5

1.1.3 Tehnologiile utilizate 7

**1.2 Cerințe specifice**........................................................................................................................ **7**

**2** **PRIMA** **FUNCȚIONALITATE** 8

**2.1 Descriere.................................................................................................................................... 8**

**2.2 Fluxul de evenimente................................................................................................................ 8**

2.2.1 Fluxul de bază 8

2.2.2 Pre-conditii.........................................................................................................................9

2.2.3 Post-conditii.......................................................................................................................9

**3** **A DOUA** **FUNCȚIONALITATE 10**

**3.1 Descriere.................................................................................................................................. 10**

**3.2 Fluxul de evenimente.............................................................................................................. 10**

3.2.1 Fluxul de bază 10

3.2.2 Fluxul alternativ...............................................................................................................10

3.2.3 Pre-conditii.......................................................................................................................11

3.2.4 Post-conditii.....................................................................................................................11

**4** **IMPLEMENTARE 12**

**4.1 Diagrama de clase................................................................................................................... 12**

**4.2 Descriere detaliată.................................................................................................................. 13**

**5** **BIBLIOGRAFIE...................................................................................................................15**

# Specificarea cerințelor software

## Introducere

În primul rând, Jocul de Memory ar trebui să conțină 2 clase proprii care au o legătură între ele. Pe de altă parte,la nivel de cod ar trebui să conțină încapsulare care să ofere proiectului o anumită securitate. Se evită astfel modificări ale obiectelor care pot duce la deteriorarea proiectului.

Din punct de vedere al perspectivei utilizatorului, jocul ar trebui să afișeze 3 ferestre. Prima este cea în care este prezentat jocul, adică numele acestuia și butonul de Start. În a doua fereastră, se cere introducerea numelui jucătorului, iar în cea de-a treia, jocul propriu-zis. În această secțiune, ar trebui să apară cele 16 perechi de cărți cu imaginile corespunzătoare. Utilizatorul are un timp limitat in care să rețină cărțile, iar mai apoi toate acestea se vor întoarce. Jucatorul are posibilitatea de a întoarce câte 2 cărți, iar în cazul în care acestea sunt la fel, vor rămâne întoarse pe tablă, iar scorul va crește. La terminarea unui joc, exista posibilitatea de a începe un joc nou printr-un buton de Restart sau sa paraseasca jocul.

### Obiective

* 16 perechi de cărți pe interfața grafică
* posibilitatea de a le vedea pe toate la începutul jocului
* un cronometru care sa țină evidența cate secunde sunt afișate cărțile
* posibilitatea de a realiza corespondența între cărți (dacă au aceeași imagine sa ramana întoarse)
* un scor care să crească de fiecare dată când o pereche a fost găsită
* o fereastra de tipul Top Players care sa retina numele jucătorului și scorul acestora în funcție de cel mai bun rezultat
* posibilitatea de restarta jocul
* posibilitatea de a fi jucat în rețea

### Definiții, Acronime și Abrevieri

StartPage = numele primei ferestre în care se găsește titlul jocului și butonul de start.

LoginPage = numele celei de-a doua ferestre in care i se cere utilizatorului sa isi noteze numele si sa intre in joc.

GameArea = cea de-a treia fereastra, in ceea ce priveste interfata grafica;aceasta reprezinta jocul propriu-zis in care se afla cartile, butonul de Exit, cel de Restart si scorul.

In ceea ce priveste codul, exista clasa Card, care reprezinta o carte si proprietatile acesteia:

* isFlipped= membru al clasei card, de tip bool, care reține daca cartea este sau nu intoarsa pe tabla
* cardId = un membru al clasei card, acesta este de tip int si retine mai exact numarul asociat fiecarei carti (de ex. 1,2,..16).
* frontImage = membru al clasei card, este de tip Image, serveste la atribuirea imaginii care se va afisa pe carte atunci cand aceasta este intoarsa
* backImage = membru al clasei card, de tip Image care atribuie imaginea corespunzatoare atunci cand cartea nu este intoarsa
* pictureBox = membru al clasei card, este de tip PictureBox pentru a atribui imaginea unui card pe interfata
* Flip = metoda din clasa card care actualizeaza imaginea cartii in functie de starea ei
* GetCardID = metoda din clasa card care returneaza id-ul cartilor; este folosita pentru a pastra membrul privat, lasand posibilitatea de a se putea lucra cu el si din celelalte clase
* GetCurrentImage = metoda din clasa card care verifica starea cartii si returneaza imaginea de fata sau spate a cartii
* UpdateImage = metoda din clasa card care seteaza imaginea corespunzatoare pe picturebox
* SetPictureBox = metoda din clasa card, avand ca scop asocierea fiecarui obiect card cu picturebox-ul său

A doua clasa se numeste Game si contine formalitati legate de functionalitatea jocului:

* Cards = membru din clasa game, de tipul unei liste, care stocheaza obiecte de tipul Card; mai exact este o lista folosita pentru a stoca toate cartile de pe tabla
* clickedCards = membru al clasei game, reprezentand o lista de obiecte Card care retine cartile pe care s-a facut click pentru a fi comparate
* flippedCards = membru din clasa game, este o lista de obiecte Card care retine cartile care au facut pereche si vor ramane intoarse pe tabla pana la sfarsitul jocului curent
* scoreLabel = membru al clasei game, de tip Label, utilizat pentru a afisa si actualiza scorul pe interfata grafica
* score = membru din clasa game de tipul int care stocheaza scorul jocului
* clickEnabled = membru din clasa game, de tip bool; acesta are rolul de a gestiona actiunile de click ale utilizatorului in diferite parti ale jocului
* StartNewGame = metoda din clasa game este folosita la initializarea unui nou joc
* InitializeCards = metoda din clasa game, care inilializeaza cartile cu imaginile corespunzatoare
* GetFrontImage = metoda din clasa game care returneaza din resurse imaginea corespunzatoare in functie de id-ul unei carti
* GetBackImage = metoda din clasa game care returneaza din resurse imaginea afisata pe toate cartile cand acestea nu sunt intoarse
* Shuffle= metoda din clasa game, care are scopul de a amesteca cartile de joc intr-o ordine aleatorie
* HandleCardClick = metoda din clasa game care gestioneaza actiunile atunci cand un jucator face click pe o anumita carte
* TestAndFlip = metoda din clasa game care are rolul de a testa anumite aspecte inainte de a intoarce o carte
* CompareCards = metoda din clasa game care compara cartile; daca corespund, acestea raman intoarse, scorul creste si sunt adaugate in lista de flippedCards pentru a nu mai putea fi comparate ulterior; daca nu sunt la fel, se intorc;
* AllPairsFound = metoda din clasa game, de tip bool care returneaza daca toate perechile au fost gasite
* UpdateScore= metoda din clasa game care se ocupa cu actualizarea valorii scorului și, de asemenea, de actualizarea afișajului scorului în interfața grafica.
* SetClickEnabled= metoda din clasa game, fiind utilă pentru a dezactiva sau activa posibilitatea de a face clic pe cărți în funcție de anumite condiții sau evenimente în joc.
* ShowAllCards = metoda in clasa game folosita pentru a intoarce toate cartile deodata in momentul inceperii unui joc
* HideAllCards = metoda din clasa game folosita pentru a intoarce cu fata in jos toate cartile
* GetCards = metoda din clasa game care returneaza lista de carti pentru a putea fi folosita si din alte parti ale codului

### Tehnologiile utilizate

Tehnologia utilizată în timpul realizării proiectului este Visual Studio 2022. În timpul realizării lucrării de documentație s-a folosit Microsoft Office Word.

## Cerințe specifice

* afisarea cartilor pe tabla
* amestecarea cartilor intr-o ordine aleatorie
* afisarea tuturor cartilor un timp limitat pentru memorare
* setarea corespunzatoare a imaginilor in functie de starea cartii: intoarsa/ neintoarsa
* posibilitatea de a face click doar pe doua carti
* compararea cartilor si ceea ce se intampla cu acestea in functie de rezultatul comparatiei
* sincronnizarea scorului cu labelul din interfata si actualizarea acestuia cu 10 puncte pentru fiecare pereche realizata
* posibilitatea de incepere a unui joc nou prin butonul de restart

# Prima funcționalitate

## Descriere

Testarea cartilor pe care s-a facut clic este prima functionalitate care urmeaza a fi prezentata. Aceasta este importanta deoarcere asigura functionarea corecta a jocului in orice conditii si in functie de orice metoda incorecta de joc ar incerca utilizatorul sa realizeze voit sau nevoit. Astfel, aceasta implementare asigura o buna desfasurare a jocului in conditii optime si corecte.

## Fluxul de evenimente

### Fluxul de bază

Aceasta functionalitate presupune mai multe verificari in urma carora o carte pe care s-a facut clic va reactiona sau nu acestui eveniment. Primul pas este de a verifica daca este sau nu activata interactiunea click-urilor. Aceasta este stocata intr-un membru al clasei game de tip bool, numit clickEnabled si retine daca se poate face clic pe tabla de joc in acel moment sau nu. Aceasta interactiune a fost implementata pentru a interzice jocatorului sa faca clic pe carti la inceputul jocului atunci cand acestea sunt intoarse pentru scurt timp, servind a fi memorate. Astfel, in cazul in care aceasta interactiune este dezactivata, cartea pe care s-a facut clic nu va reactiona in niciunul fel eliminand situatiile in care s-ar putea trisa.

Urmatoarea verificare este cea care impiedica ca o carte sa fie comparata cu o carte care deja a facut pereche si a ramas intoarsa pe tabla. Acest lucru este interzis deoarece cartile care au facut deja pereche sunt considerate iesite din joc. Pentru a realiza acest lucru, s-a folosit o lista de obiecte card, numita flippedCards care retine toate cartile care au ramas intoarse pe tabla in urma compararii. Astfel, se verifica daca cartea pe care s-a facut clic se gaseste sau nu in aceasta lista. In cazul in care aceasta se gaseste, cartea pe care s-a facut clic, nu va reactiona in niciun fel.

O ultima verificare este cea in care s-ar face clic pe aceeasi carte de doua ori. In acest caz exista posibilitatea ca scorul sa creasca in mod eronat deoarece cartea s-ar fi comparat cu ea insasi, id-ul ei ar coincide si s-ar considera o pereche, ceea ce este complet gresit. Pentru a evita astfel de situatii s-a folosit o lista de obiecte card numita clickedCards care contine cartile pe care s-a facut clic pentru a fi comparate. Asadar, se veridica daca cartea pe care s-a facut clic la momentul actual se regaseste deja in acea lista. In cazul in care este asa, cartea pe care s-a facut clic nu va reactiona in niciun fel, lucru care nu il va lasa pe utilizator sa compare o carte cu ea insasi.

In cazul in care s-a trecut peste toate aceste verificari cu succes, cartea pe care s-a facut clic se va intoarce si automat va fi adaugata si in lista de carti pe care s-a facut clic (clickedCards) pentru a servi compararii.

Aceasta functionalitate a fost apelata o singura data, tot in clasa game in functia HandleCardClick. Aceasta functie este apelata pe interfata in momentul in care are loc o interactiune de clic pe un picturebox. Astfel, functia aceasta apleleaza testarea cartii pe care s-a facut clic iar mai apoi compararea ei.

### Pre-condiții

Pentru a porni funcționalitatea de testare a cartilor, trebuie să parcurgi următoarele acțiuni:

* Pornirea jocului:
  + Interfața jocului este afișată corect și toate elementele sunt inițializate.
  + Cerere catre utilizator să înceapă jocul, prin butonul de start si prin accesarea butonului de Log In dupa introducerea numelui.
* Click pe o carte:
  + Utilizatorul trebuie să poată face clic pe o carte pentru a o întoarce.
  + Dupa ce a selectat una, ar trebui sa mai selecteze inca una de orice fel pentru a putea observa ca si daca alege „gresit” nu va schimba functionalitatea jocului

### Post-condiții

După rularea acestei funcționalități de joc se vor observa următoarele:

Inițierea jocului:

* Toate cărțile sunt așezate pe ecran și asociate cu imagini corespunzătoare.
* Utilizatorul ar trebui să vadă o prezentare clară a stării inițiale a jocului, inclusiv scorul curent.

Evenimentele de clic:

* Utilizatorul ar trebui să poată face clic pe două cărți diferite pentru a le compara.
* După clic, cărțile respective ar trebui să se întoarcă pentru a dezvălui imaginile lor.

Feedback vizual:

* La fiecare clic, utilizatorul ar trebui să primească feedback vizual pentru a confirma că a făcut o alegere.

# A doua funcționalitate

## 3.1 Descriere

Compararea cartilor si ceea ce se intampla cu acestea in urma acestui eveniment este cea de-a doua functionalitate care urmeaza a fi prezentata. Aceasta este cea mai importanta parte deoarece reprezinta obiectivul principal al jocului.

Jocul de memorie implică găsirea de perechi potrivite de cărți identice, iar scopul jocului este amintirea locației și imaginea fiecărei cărți pentru a le potrivi corespunzător. Astfel, compararea cărților în jocul de memorie este crucială pentru atingerea obiectivului jocului, dar și pentru dezvoltarea și îmbunătățirea diferitelor abilități cognitive ale memoriei.

## Fluxul de evenimente

### 3.2.1Fluxul de bază

In primul rand, se verfica daca lista de carduri clickedCards contine doua elemente. Aceasta lista stocheaza cartile pe care s-a facut click pentru comparare si va contine mereu un numar maxim de doua elemente deoarece jocul presupune compararea a doua carti intre ele, nu mai multe. Astfel daca aceasta lista contine numarul maxim de carti se poate trece la urmatorul pas, adica compararea celor doua.

Ulterior se verifica daca coincid primul si respetiv al doilea element al listei. Aceasta verificare se face dupa id-ul cartilor, apeland metoda din clasa card GetCardId pentru a-l obtine. Se apeleaza aceasta metoda deoarece membrul cardId din clasa card este privat, oferind o anumita securitate clasei. In cazul in care id-ul cartilor corespund, scorul se va actualiza cu 10 puncte, fiind apelata functia UpdateScore. Aceasta are datoria de a actualiza atat textul de pe interfata grafica care indica scorul cat si variabila score.

In continuare, urmeaza a fi verifcat daca s-au descoperit toate cartile. Se apeleaza functia AllPairsFound care returneaza true sau false. Aceasta functie returneaza adevarat daca numarul total al cartilor intoarse pe tabla coincide cu numarul total al cartilor jocului. In cazul in care acestea coincid reiese ca jucatorul a gasit toate perechile, finalizand jocul. Astfel, va fi afisat un mesaj cu „Felicitari! Ai gasit toate perechile de carti!”.

Pe de alta parte, in cazul in care id-ul celor doua carti nu corespunde se va urma acest scenariu. Este utillizata o scurta intarziere de timp pentru ca utilizatorul sa vada cartile pe care le-a selectat chiar daca acestea nu se potrivesc. Mai apoi, cele doua se vor intoarce la loc apeland metoda Flip din clasa card.

Un ultim eveniment necesar este acela de a goli lista de clickedCards inainte de aiesi din functie pentru a fi pregatita pentru urmatoarea comparatie de carti.

Aceasta functionalitate a fost apelata intr-un singur loc, si mai exact in functia HandleCardClick care apartine tot de clasa Game. Aceasta functie gestioneaza evenimentele care se intampla dupa ce s-a facut clic pe o carte de joc.

### Flux alternativ

In trecut aceasta functionalitate a fost rezolvata si altfel. Compararea cartilor se realiza dupa imagini ci nu dupa numere (id) ca in momentul prezent. Clasa game avea un alt membru care reprezenta un vector de imagini in care erau inscrise toate imaginile de la resurse, care ulterior prin alta functie erau dublate si amestecate, iar ulterior se verifica daca imaginile celor doua carti pe care s-a facut clic coincid. Din punctul meu de vedere, metoda actuala este mai optima si mai usoara decat cea precedenta.

### Pre-condiții

Această secțiune este destinată identificarii pașilor pe care utilizatorul trebuie să îi urmeze pentru a activa și utiliza funcționalitatea descrisă. În contextul metodei CompareCards:

Activarea și Inițierea Jocului:

* Conectarea la joc prin apasarea butonului start, iar mai apoi prin apasarea butonului de Log In, dupa introducerea numelui

Selectarea Cărților:

* În timpul jocului, utilizatorul trebuie să facă clic pe două cărți pentru a le selecta.

Feedback Vizual pentru Compararea Cărților:

* Feedback-ul include actualizarea scorului, evidențierea cărților potrivite sau întoarcerea cărților în cazul nepotrivirii.

Repetarea Procesului:

* După rezolvarea feedback-ului, jucătorul poate continua să selecteze perechi de cărți până când toate perechile sunt potrivite și jocul se încheie.

### Post-condiții

După rularea funcționalității CompareCards, utilizatorul ar trebui să observe următoarele:

Potrivirea Cărților:

* Dacă cărțile selectate sunt potrivite, acestea rămân întoarse si scorul se actualizează
* Utilizatorul ar trebui să observe o schimbare în scorul afișat pe ecran, indicând că a obținut puncte pentru potrivirea cărților.

Cărți Nepotrivite:

* În cazul în care cărțile nu se potrivesc, acestea vor fi întoarse înapoi după o scurtă întârziere.
* Utilizatorul va observa intoarcerea cărților care revin la starea lor originală.

Continuarea Jocului:

* După finalizarea comparării cărților, utilizatorul poate continua să selecteze alte perechi de cărți sau să încheie jocul în funcție de logica specifică a aplicației.

Pentru a se asigura că funcționalitatea s-a terminat corect, utilizatorul poate verifica scorul său, să observe starea cărților pe tablă, și să interacționeze în continuare cu interfața grafică în conformitate cu regulile jocului.

**4. Implementare**

## 4.1Diagrama de clase

GAME

- cards: list<Card>

- clickedCards: list<Card>

- flippedCards: list<Card>

- score: int

- clickEnabled : bool

- scoreLabel : Label

+Game(int, int, Label);

+ StartNewGame( );

- InizilizeCards( );

- GetFrontImage(int);

- GetBackImage( );

- Shuffle(List<Card>)

+ HandleCardClick(Card)

- TestAndFlip(Card)

- CompareCards( );

- AllPairsFound( );

- UpdateScore ( );

+ SetClickEnabled (bool);

+ ShowAllCards( );

+ HideAllCards( );

+GetCards( );

CARD

- cardId: int

- pictureBox: PictureBox

- frontImage: Image

- backImage: Image

- isFlipped: bool

+ Card(int, Image, Image, PictureBox);

+ Flip(bool);

+ GetCurrentImage( );

+ GetCardId( );

- UpdateImage( );

+ SetPictureBox(PictureBox)

## Descriere detaliată

**NU**

**DA**

**clickedCards.Count==2**

**DA**

**DA**

**DA**

**clickedCards.Add(card)**

**Card.Flip(true)**

**clickedCards.Contains(card)**

**!clickEnabed**

**flippedCards.Comntains(card)**

Hale

HandleCardClick

**clickEnabled = true**

**timer stop**

**StartNewGame, timer start**

**clickEnabled = false**

s

**Start**

**Stop**

**clickedCards.Clear();**

**clickedCards[0].Flip(false)**

**clickedCards[1].Flip(false)**

**“Felicitari!Ai gasit toate perechile!”**

AllPairsFound

**flippedCards.Add(clickedCards[0])**

**flippedCards.Add(clickedCards[1])**

**UpdateScore**

**NU**

**DA**

**clickedCards[0].GetCardId( )==**

**clickedCards[1].GetCardId( )**

# Bibliografie

1. <https://users.utcluj.ro/~igiosan/Resources/PC/Lab/L01.pdf> - informare scheme logice
2. <https://learn.microsoft.com/ro-ro/dotnet/fundamentals/networking/sockets/socket-services#create-a-socket-client> – informatii despre retea
3. <https://www.mooict.com/c-tutorial-create-a-superhero-memory-game/> - inspiratie pentru realizarea jocului
4. <https://tuneam.files.wordpress.com/2020/11/programare-visuala-c-prin-joc.pdf> - inspiratie pentru realizarea jocului
5. <https://www.sourcecodester.com/c/15773/simple-memory-game-c-free-source-code.html?utm_content=cmp-true> – inspiratie pentru interfata
6. Cartea de la curs a domnului Macarie Breazu
7. Laboratoarele puse la dispozitie de catre doamna profesoara Cofaru Ileana
8. Vizionarea videoclipurilor pe youtube
9. Videoclipul cu jocul Snake