Universitatea ”Ștefan cel Mare” Suceava

Facultatea de Inginerie Electrică și Știința Calculatoarelor

Proiect Disciplina POO

Joc de Dame

Student:

Todosi George Vasile

Grupa 3124a

Calculatoare

Anul 2016-2017

Profesor Îndrumător:

Remus Prodan

Cuprins:

1. TEMA ȘI MOTIVAȚIA ALEGERII

TEMA

Crearea unui joc de dame pentru două persoane sub forma unei aplicații pe calculator.

MOTIVAREA ALEGERII TEMEI

Am ales această temă deoarece are un obiectiv fix, clar și concret dar care în același timp permite diferite moduri de abordare și implementare (interfață grafică sau text, cu sau fară posibilitatea de a salva/relua jocul, etc) și pentru că deși regulile jocului sunt simple implementarea detaliilor, excepțiilor și cazurilor particulare crește considerabil complexitatea codului.

2. NOȚIUNI TEORETICE

2.1 Descrierea Problemei

Jocul de dame se joacă în doi, pe o tablă de 64 de pătrățele în format de 8x8. Obiectivul jocului este simplu, trebuie să sari peste piesele adversarului pentru a i le captura iar cel care ramâne fără piese la finalul jocului pierde.

Reguli:

- Piesele pot fi mutate doar pe diagonală
- Piesele pot fi mutate doar înainte (excepție: piesele încoronate, salturile multiple)
- Distanța maximă ce pot fi mutate este de o singură pătrățică (excepții: când se sare peste piesa adversarului, piesele încoronate)
- Pentru a captura o piesă adversă pătrățica din spatele ei trebuie să fie liberă
- O piesă ce ajunge la capul celălalt devine încoronată
- Jocul se termină atunci când un jucător și-a pierdut toate piesele sau nu mai poate să le miște

2.3 Abordarea problemei

2.3.1 Grafica

Pentru creearea interfeței grafice a jocului vom folosi librăria SFML (Simple Fast Multimedia library) pentru ca jocul sa arate modern și cât mai atractiv pentru ultilizator.

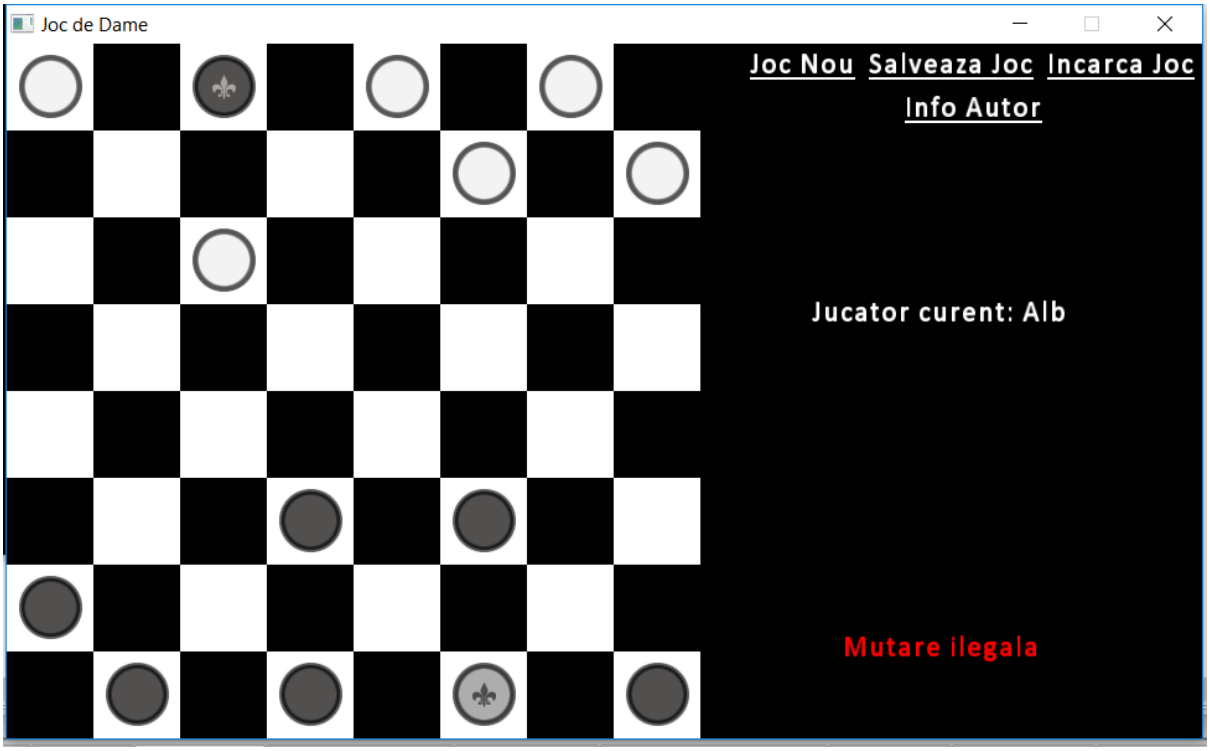
Tabla va fi formata din pătrate albe și negre și situată în partea stângă a ecranului.

Meniul va fi de culoare albă și situat în dreapta sus.

Pe partea dreaptă în mijloc se va indica jucătorul curent.

Pe partea dreaptă în partea jos vor apărea mesajele de la sistem legate de erori de mutare sau terminarea jocului.

Piesele vor fi albe și negre, cu un idicator special pentru piese încoronate precum și un idicator de selcție.



2.3.2 Modul de Operare

Pentru rezolvarea modului intern de funcționare vom folosi câte o clasă pentru fiecare element al jocului:

1. O clasă *patrat* pentru pătratele tablei – aceasta va avea ca parametri coordonate matriceale (linia și coloana), coordonate spațiale și metode pentru returnarea acestora precum și pentru desenarea pătratului în fereastra principală.
2. O clasă *tabla* formată dintr-o matrice de obiecte din clasa *patrat* – este practic clasa care formează matricea tablei și are ca metodă returnarea coordonatelor matriceale în funcție de coordonatele spațiale precum și inversul.
3. O clasă abstractă *piesa* pentru piese – are ca parametrii coordonatele matriceale și spațiale ale piesei, string-uri pentru numele texturilor (de bază, piesă selectată, încoronată și încoronată selectată), variabile pentru reținerea tipului piesei, selectării piesei și stării acesteia în cadrul jocului.
4. O clasă derivată *dame_a* pentru damele albe – este derivată din clasa *piese* și are operatorii supradefiniți pentru a efectua operații de mutare
5. O clasă derivată *dame_n* pentru damele negre – este derivată din clasa *piese* și are operatorii supradefiniți pentru a efectua operații de mutare
6. O clasă de coordonate matriceale *coorodnate* – are ca parametrii linia și coloana
7. O clasă de coordonate spațiale *c_spatiu* – are ca parametrii punctele de început și sfârșit ale unui pătrat pe axele X și Y ale ferestrei.
8. Trei clase pentru excepții *este piesa, piesa_inamic, afara_ecran* – clase utilizate pentru tratarea mutărilor ilegale.

O descriere mai detaliată a fiecărei clase, parametru și metode se poate regăsi în documentația DoxyBlocks atașată și în capitolul 3.2.

Pentru selectarea pieselor vom compara coordonatele fiecărei piese în parte cu coordonatele la care s-a dat click.

Mișcarea corectă a pieselor se va face interogând matricea de pătrate pentru fiecare tip de mutare (avans cu un pătrat dreapta/stânga, mâncare dreapta/stânga pentru piese normale și mutare la un pătrat anume pentru piese încoronate).

Meniul va funcționa, de asemenea, pe baza locației în care s-a dat click.

Funcția de salvare va salva parametrii fiecărui obiect într-un fișier iar funcția de încărcare îi va prelua din fișier.

2.4 Elemente POO

Pe lângă clase vom utiliza și următoarele elmente specifice programării orientate pe obiecte:

1. Supradefinirea operatorilor:

- Operatorul *++* : supradefinit pentru clasele *dama_n* și *dama_a* pentru a face o mutare în stânga (pre-incrementare) sau în dreapta (post-incrementare) și în clasa *coordonate* pentru incrementarea ambilor parametri (linie și coloană).
- Operatorii *+* și - : suprdefinit pentru clasele *dama_n* și *dama_a* pentru a face o mutare la un pătrat specificat pe direcție verticală (+) sau orziontală (-).
- Operatorul << : supradefinit pentru clasele *coordonate* și *coord_s* pentru afișare și scrierea în fișier.
- Operatorii < și == : supradefiniți în clasa *coordonate* pentru compararea cu un alt set de coordonate.

2. Moștenire:

- Clasele *dama_n* și *dama_n* moștenesc clasa *piesa*

3. Clase abstracte:

- Clasa *piesa* este o clasă abstractă deoarece conține mai multe funcții pur virtuale astfel ea neputând fi instanțiată.

4. Polimorfism

- Polimorfismul claselor este evidențiat în funcțiile de verificare a locului de mutare a unei piese (exemplu: *int mutare_1_dreapta(piesa *p , coordonate& mat_poz)*). Aceastea primesc în parametru un obiect de tip piesă iar pentru a verifica dacă s-a cerut o mutare cu 1 sau 2 două căsuțe înainte funcția apeleaza metoda *get_x_1()* sau *get_x_2()*. Deoarece deplasarea înainte este diferită pentru fiecare tip de damă aceste metode vor efectua operații diferite. Polimorfismul asigură că deși parametrul este de tip *piesa* metoda aplicată va fi cea a obiectului din clasa derivată precum și elminând nevoia de definire a unui set de funcții pentru fiecare tip de clasă derivată.

5. Clasa Template

- Este folisită în cazul funcțiilor *int victorie(T *t), void deselecteaza(T &p), void schimba_jucator(T &p)*. Aceasta permite utilizarea aceleiași funcții pentru diferite tipuri de variabile. În cazul nostru este o alternativă la polimorfism dar poate fi utilă în cazurile în care tipurile ce se doresc a fi folosite nu sunt derivate din aceeași clasă.

6. Referințe

- Referințele sunt utilizate în clasele XYZ pentru returnarea coordonatelor și în parametrii funcțiilor XYZ. Referințele sunt un mod ușor de a transmite variabilele prin adresă și ajută la reducerea încărcării memoriei.

7. Excepții

- Excepțiile sunt folosite în cadrul executării mutărilor pentru a putea evita mutări ce ar rezulta în suprapunerea unor piese sau mutara înafara tablei de joc. În momentul în care este detectată o astfel de mutare executarea ei se va anula și se va afișa un mesaj de eroare.

3. IMPLEMENTARE

3.1 Tehnologii Folosite

Implementarea programului a fost făcută în limbajul *C++.*

IDE-ul folosit este *CodeBlocks 16.01*.

Compilatorul folosit este *GNU GCC Compiler 4.9.2 Standard MinGW 32-bit Edition* cu extensia *-std=gnu++11*.

Libraria grafică folosită este *SFML 2.4.0*.

3.1 Descrierea Claselor

1. Clasa *patrat*

Membrii:

```
protected:
string nume_textura; ///< numele texturii
string nume_textura_baza; ///< numele texturii de baza (utilizata pentru revenire la textura initiala)
string nume_highlight; ///< numele texturii pentru luminare
string nume_textura_incoronat; ///< numele texturii piesa incoronata
string nume_highlight_incoronat; ///< numele texturii pentru luminarea unei piese incoronate
string nume_highlight_baza; ///< numele texturii pentru luminare de baza (utilizata pentru revenire la textura de luminare initiala)
coordonate xy; ///< coordonate matriceale
c_spatiu c_s; ///< coordonate spatiale
int is_highlight = 0; ///< variabila pentru luminare
int in_joc; ///< variablia de stare
int incoronat; ///< variabila pentru tipul piese
```

Metode:

```
public:
//! constructor
patrat();

//! destructor
~patrat();

//!returneaza coordonatele spatiale
c_spatiu & get_c_s();

//!returneaza coordonata spatiala de pe axa X de unde incepe patratul
int get_x_s();

//!returneaza coordonata spatiala de pe axa X unde se sfarseste patratul
int get_x_e();

//!returneaza coorodnata spatiala de pe axa Y de unde incepe patratul
int get_y_s();

//!returneaza coordonata spatiala de pe axa Y de unde se sfarseste patratul
int get_y_e();

//!deseneaza patratul in fereastra din parametru
void deseneaza(sf::RenderWindow & window /**< fereastra in care se deseneaza*/ );

//!intializeaza patratul cu coordonatele matriceale i j
/*!
\param i linia
\param j coloana
 */
void initializeaza(int i, int j);
```

2. Clasa *tabla*

Membrii:

```
1. private: patrat t[8][8]; ///<matricea de patrate ce formeaza tabla   Metode:
```

Metode:

```
1. public:
2. //! constructor
3. tabla();
4.
5. //! destructor
6. virtual~tabla();
7.
8. //! deseneaza tabla
9. /*!
10. \param window fereastra in care se deseneaza
11. */
12. void deseneaza(sf::RenderWindow & window);
13.
14. //! initializeaza fiecare patrat din matricea de patrate
15. void initializeaza();
16.
17. //!returneaza coordonatele spatiale pentru pozitia i j in matricea de patrate
18. /*!
19. \param i linia
20. \param j coloana
21. */
22. c_spatiu & get_c_s(int i, int j);
23.
24. //! returneaza coordonatele matriceale unde s-a dat click
25. /*!
26. \param localPosition coordonatele spatiale la care s-a dat click
27. */
28. coordonate coord_click(sf::Vector2i localPositon);
```

3. Clasa *piesa*

Membrii:

```
1. protected:
2. string nume_textura; ///< numele texturii
3. string nume_textura_baza; ///< numele texturii de baza (utilizata pentru revenire la textur
   a initiala)
4. string nume_highlight; ///< numele texturii pentru luminare
5. string nume_textura_incoronat; ///< numele texturii piesa incoronata
6. string nume_highlight_incoronat; ///< numele texturii pentru luminarea unei piese incoronat
   e
7. string nume_highlight_baza; ///< numele texturii pentru luminare de baza (utilizata pentru
   revenire la textura de luminare initiala)
8. coordonate xy; ///< coordonate matriceale
9. c_spatiu c_s; ///< coordonate spatiale
10. int is_highlight = 0; ///< variabila pentru luminare
11. int in_joc; ///< variablia de stare
12. int incoronat; ///< variabila pentru tipul piese
```

Metode:

```
public:
//!constructor
piesa();

//!destructor
virtual~piesa();

//!desneaza piesa in fereastra parametru
/*!
\param window fereastra in care se deseneaza
*/
virtual void deseneaza(sf::RenderWindow & window);

//!seteaza coordonatele spatiale
/*!
\param coordonate spatiale
*/
virtual void pozitie(c_spatiu & c);

//!seteaza coordonatele matriceale
/*!
\param _x linia
\param _y coloana
*/
virtual void coord(int _x, int _y);

//!pune piesa in joc
virtual void pune_in();

//!scoate piesa din joc
virtual void pune_out();

//!returneaza starea piesei
/*!
\return 1 in joc
\return 0 scoasa din joc
 */
virtual int stare();

//! returneaza linia pe care se afla piesa
virtual int get_x();

//!returneaza coloana pe care se afla piesa
virtual int get_y();

//!lumineaza piesa
virtual void highlight_on();

 //! opreste luminarea piesei
virtual void highlight_off();

 //! incoroneaza piesa
virtual void incoroneaza();

//!returneaza tipul piesei
/*!
\return 1 piesa incoronata
\return 0 piesa normala
*/
virtual int tip();

//! returneaza coordonatele matriceale
virtual coordonate & get_coord();
```

```
64.
65. //!returneaza coordonatele spatiale
66. virtual c_spatiu & get_c_spatiu();
67.
68. //!decoroneaza o piesa
69. virtual void decoroneaza();
70.
71. //!returneaza daca s-a dat click pe acea piesa
72. /*!
73. \param localPosition coordonate spatiale la care s-a dat click
74. \return 1 daca pe piesa s-a dat click
75. \return 0 daca pe piesa nu s-a dat click
76. */
77. virtual int is_clicked(sf::Vector2i localPosition);
78.
79. //!returneaza linia in cazul unui avans de 1 patrat pe directia de baza a piesei
80. virtual int get_x_1() = 0;
81.
82. //!returneaza linia in cazul unui avans de 2 patrate pe directia de baza a piesei
83. virtual int get_x_2() = 0;
```

4. Clasa *dame_a*

Membrii: moșteniți din *piesa*

Metode:

```
1.  public:
2.  //!constructor
3.  dama_a();
4.
5.  //! destructor
6.  virtual~dama_a();
7.
8.  //! returneaza linia in cazul unui avans de 1 patrat inainte
9.  int get_x_1();
10.
11. //! returneaza linia in cazul unui avans de 2 patrate inainte
12. int get_x_2();
13.
14. //!muta piesa un patrat la dreapta
15. void operator++(int);
16.
17. //!muta piesa un patrat la stanga
18. void operator++();
19.
20. //!muta piesa la coordonatele matriceale din parametru, manancand toate piesele peste care
    sare, in directia jos
21. /*!
22. \param mat_poz coordonate matriceale
23. */
24. void operator + (coordonate & mat_poz);
25.
26. //!muta piesa la coordonatele matriceala din parametru, manancand toate piesele peste care
    sare, in directia sus
27. /*!
28. \param mat_poz coordonate matriceale
29. */
30. void operator - (coordonate & mat_poz);
```

5. Clasa *dame_n*

Membrii: moșteniți din *piesa*

Metode:

```
1.  public:
2.  //!constructor
3.  dama_n();
4.
5.  //! destructor
6.  virtual~dama_n();
7.
8.  //! returneaza linia in cazul unui avans de 1 patrat inainte
9.  int get_x_1();
10.
11. //! returneaza linia in cazul unui avans de 2 patrate inainte
12. int get_x_2();
13.
14. //!muta piesa un patrat la dreapta
15. void operator++(int);
16.
17. //!muta piesa un patrat la stanga
18. void operator++();
19.
20. ///!muta piesa la coordonatele matriceale din parametru, manancand toate piesele peste care
     sare, in directia jos
21. /*!
22. \param mat_poz coordonate matriceale
23. */
24. void operator + (coordonate & mat_poz);
25.
26. //!muta piesa la coordonatele matriceala din parametru, manancand toate piesele peste care
    sare, in directia sus
27. /*!
28. \param mat_poz coordonate matriceale
29. */
30. void operator - (coordonate & mat_poz);
```

6. Clasa *coorodnate*

Membrii:

```
1.  public:
2.  int x; ///<  coordonata liniei
3.  int y; ///<  coordonata coloanei
```

Metode:

```
1.  public:
1.  //!constructor
2.  coordonate();
3.
```

```
4.  //!destructor
5.  virtual~coordonate();
6.
7.  //!returneaza daca ambele coordonate sunt mai mici decat cele din parametru
8.  /*!
9.  \param ob obiectul cu care se compara
10. \return 1 adevarat
11. \return 2 fals
12. */
13. int operator < (coordonate & ob);
14.
15. //!incremeanteaza ambele coordonate
16. void operator++(int);
17.
18.
19. //!verifica egalitatea
20. /*!
21. \return 1 adevarat
22. \return 0 fals
23. */
24. int operator == (coordonate & ob2);
25.
26. //! operator redefinit pentru scrierea in fisier
27. friend ostream & operator << (ostream & c, coordonate & ob);
```

7. Clasa *c_spatiu*

Membrii:

```
1.  public:
2.  int x_start; ///< coordonata spatiala de inceput pe axa X
3.  int x_end; ///< coordonata spatiala de sfarsit pe axa X
4.  int y_start; ///< coordonata spatiala de inceput pe axa Y
5.  int y_end; ///< coordonata spatiala de sfarsit pe axa Y
```

Metode:

```
1.  public:
1.  //!constructor
2.  c_spatiu();
3.
4.  //!destructor
5.  virtual~c_spatiu();
6.
7.  //!constructor implicit
8.  /*!
9.  \param _x_s coordonata spatiala de inceput pe axa X
10. \param _x_e coordonata spatiala de sfarsit pe axa X
11. \param _y_s coordonata spatiala de inceput pe axa Y
12. \param _y_e coordonata spatiala de sfarsit pe axa Y
13. */
14. c_spatiu(int _x_s, int _x_e, int _y_s, int _y_e);
15.
16. //!seteaza coordonatele
17. /*!
18. \param _x_s coordonata spatiala de inceput pe axa X
19. \param _x_e coordonata spatiala de sfarsit pe axa X
20. \param _y_s coordonata spatiala de inceput pe axa Y
21. \param _y_e coordonata spatiala de sfarsit pe axa Y
22.  */
23. void set_c(int _x_s, int _x_e, int _y_s, int _y_e);
24.
```

```
25. //! operator redefinit pentru scrierea in fisier
26. friend ostream & operator << (ostream & c, c_spatiu & ob);
```

8. Clasele *este piesa, piesa_inamic, afara_ecran*

Membrii: moștenește clasa exception din C++

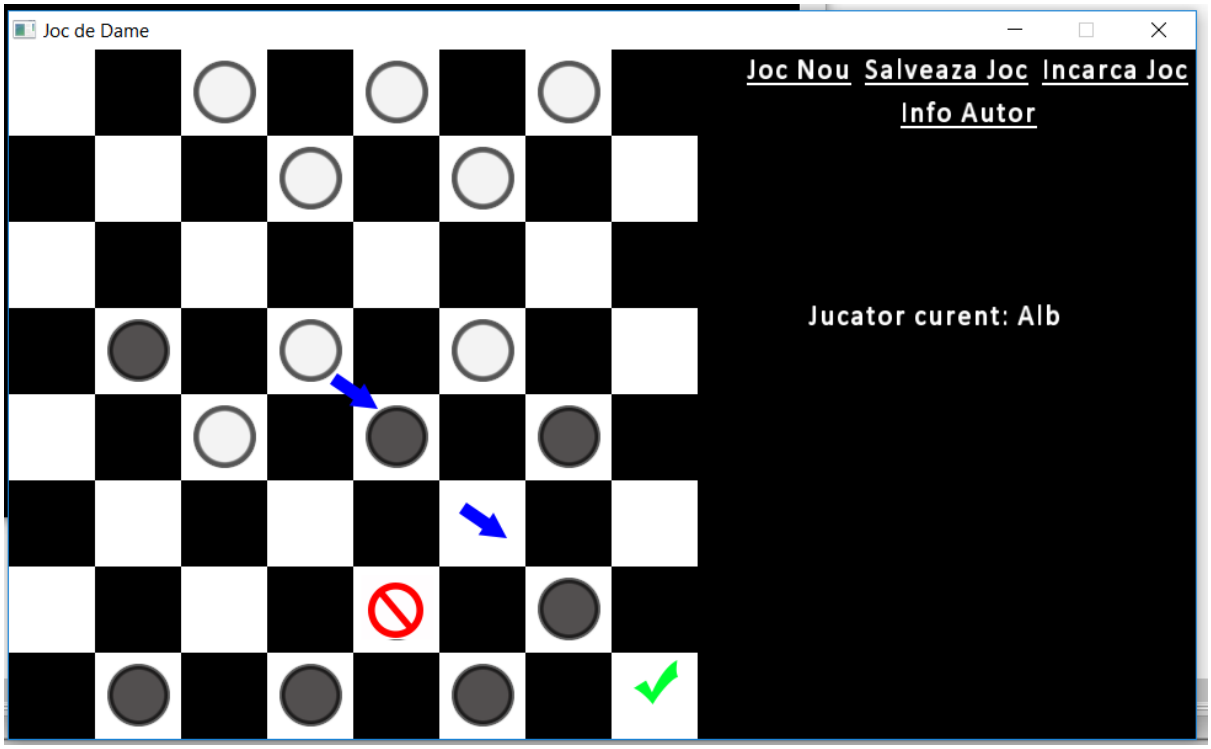
Metode:

```
1.  public: virtual
2.  const char * what() const  throw () /// returneaza un mesaj despre existenta unei piese
3.  {
4.      return "mesaj de eroare";
5.  }
```

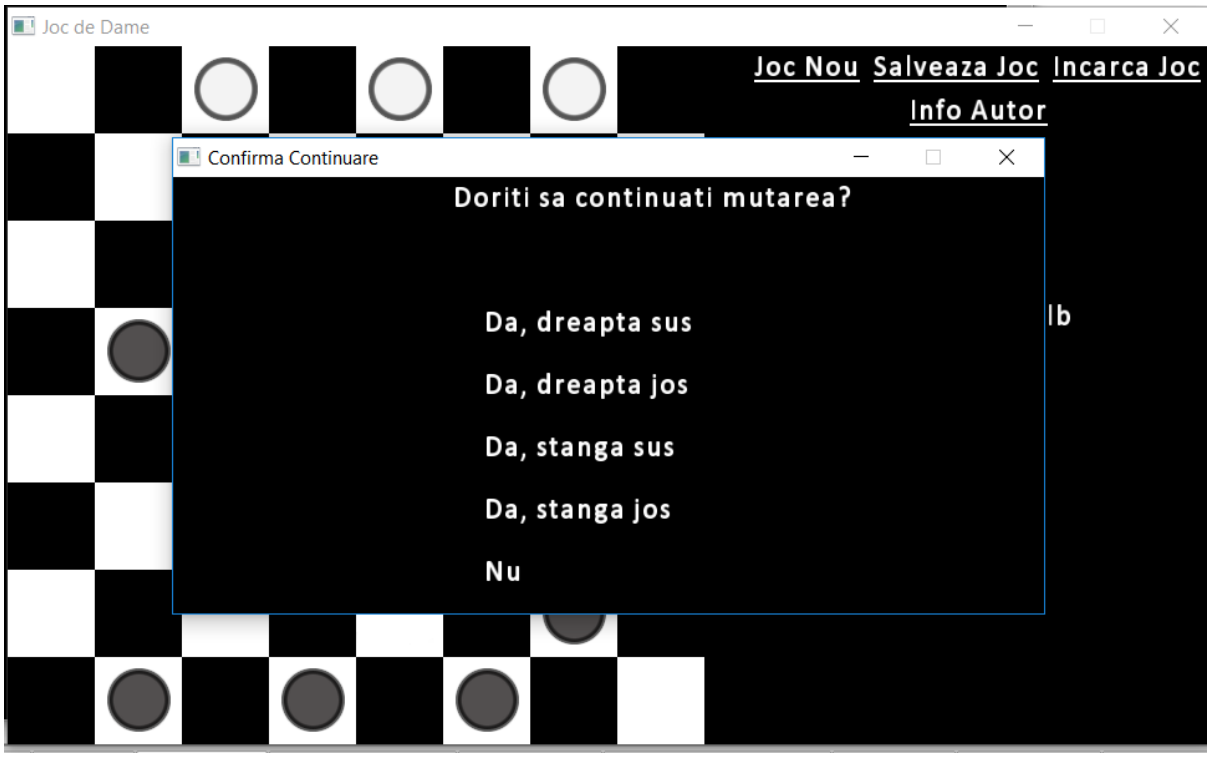
4. ANALIZA SOLUȚIEI IMPLEMENTATE

Studiu de caz: Salturi multiple

Jocul opereaza pe principiul că fiecare jucător va muta câte o piesă rând pe rând dând fiecărui jucător câte o mutare. În cazul în care saltul peste o piesă inamică permite un alt salt dacă îi dăm jucătorului respectiv o mutare în plus aceasta ar putea fi folosită în mod illegal prin executarea altei mutări în loc de continuare salturilor.

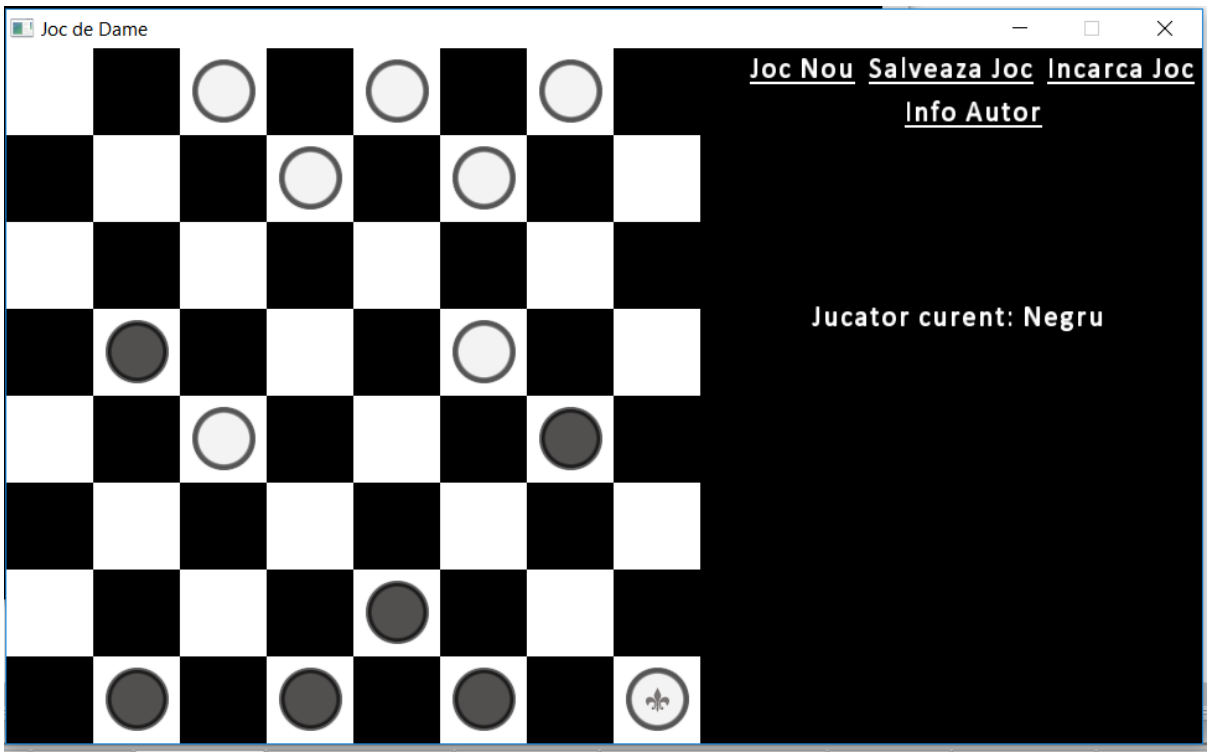


Pentru a rezolva acest caz special am fost nevoiți să implementăm o fereastră nouă în care se cere jucătorului confirmarea continuării mutării, inclusive direcția pe care dorește să mute pentru a elimia posibilitatea de a executa mutări ilegale.



Încercarea de a muta într-o direcție nepermisă va rezulta într-un mesaj de eroare.

Astfel jucătorul este forțat să continue mutarea într-un mod corect sau să renunțe la ea.



5. MANUAL DE UTILIZARE

Jocul are un design user-friendly și utilizarea este foarte ușoară și intuitivă.

4.1 Caracteristicele Jocului

1. Jocul este doar pentru doi jucători umani.
2. Nu nu are funcție de inteligență artificială.
3. Se desfășoară local pe același calculator.
4. Nu se poate juca prin LAN/Internet.
5. Are funcție de salvare joc/încărcare joc precedent.
6. Nu beneficiază de sistem cloud pentru protecția salvărilor.
7. Rezoluția jocului este fixate la 965x560.
8. Nu are sunete.
9. Nu are opțiune de schimbare sau personalizare a regulilor jocului.
10. Controlul se face prin intermediul mouse-ului.

Imediat după deschidere programului jocul poate fi început direct prin mutarea pieselor.

4.2 Regulile jocului

- Piesele pot fi mutate doar pe diagonală
- Piesele pot fi mutate doar înainte (excepție: piesele încoronate, salturile multiple)
- Distanța maximă ce pot fi mutate este de o singură pătrățică (excepții: când se sare peste piesa adversarului, piesele încoronate)
- Pentru a captura o piesă adversă pătrățica din spatele ei trebuie să fie liberă
- O piesă ce ajunge la capul celălalt devine încoronată
- Jocul se termină atunci când un jucător și-a pierdut toate piesele sau nu mai poate să le miște

4.3 Mutarea Pieselor

Pentru a muta o piesă trebuie mai întâi selectată dând click stânga pe ea. O dată selectată aceasta își va schimba aspectul. Pentru a o muta dați click unde vreți să o duceți iar aceasta își va schimba poziția. În cazul unei mutări greșite veți primi un mesaj de eroare.

Dacă mutarea este una de salt ce permite mai multe salturi după veți fi nevoiți să confirmați continuarea mutării pe o direcție. **ATENȚIE**: fereastra principală nu se actualizează pană la închiderea ferestrei de confirmare deci trebuie să rețineți ordinea alegerilor pentru a ști poziția în care sunteți.

4.4 Meniul

Meniul este accesat cu mouse-ul dând click pe unul din butoanele:

- *Joc Nou* va începe un joc nou. **ATENȚIE**: jocul precedent va fi pierdut dacă nu a fost salvat.
- *Salvaeaza Joc* va salva jocul. Dați click pe unul din slot-uri pentru a salva jocul în el. **ATENȚIE**: există doar 5 sloturi în care jocurile pot salvate. Selectarea unui slot unde există deja o salvare o va înlocui pe cea veche **FĂRĂ** confirmare.
- *Încarca Joc* va încărca un joc precedent care a fost salvat. **ATENȚIE**: încărcarea unui joc precedent se face **FĂRĂ** confirmare iar jocul curent va fi pierdut.
- *Info Autor* prezintă date despre autorul programului

Pentru a închide jocul faceți click pe X din colțul ferestrei.

6. CONCLUZII

Ca orice program modul de implementare ales are avantaje și dezavantaje.

Avantajul implementării curente constă în posibilitatea de extindere a programului pentru alte jocuri ce se pot desfășura pe o tablă 8x8 cum ar fi șahul prin simpla adaugare a unei clase noi cu metode de mutare pentru fiecare piesă și adăugarea unei opțiuni de schimbare a modului în meniu.

Dezavantajul principal al acestui mod de implementare este dificultatea determinării concrete a unei stări de remiză. Deși s-ar putea implementa o funcție de verificare a cazului în care nici un jucător nu mai are mutări legale acesta este doar unu din cazuri și relativ rar. Pentru cazurile în care jucătorii rămân pe tablă cu piese care nu se vor ataca niciodată (exemplu: fiecare jucător rămâne cu o singură damă încoronată la final) și au practic o infinitate de mutări legale rezultatul jocului trebuie stabilit înafara mediului digital. Alt dezavantaj dar mai minor ar fi neactualizarea în timp real a ferestrei principale în cazul deschiderii unei ferestre de confirmare ceea ce duce la necesitatea de a reține alegerile pentru a determina poziția curentă în cazul unor salturi multiple precum și faptul că poziția ferestrei de confirmare este relatviă rezoluției desktop-ului și nu poziției ferestrei principale.

7. BIBLIOGRAFIE

1. The C++ Programming Language (4th Edition) Addison-Wesley ISBN 978-0321563842. May 2013
2. www.cplusplus.com
3. http://www.sfml-dev.org/documentation/2.4.0/
4. http://apollo.eed.usv.ro/~remus/
5. https://www.tutorialspoint.com/cplusplus/
6. http://stackoverflow.com/