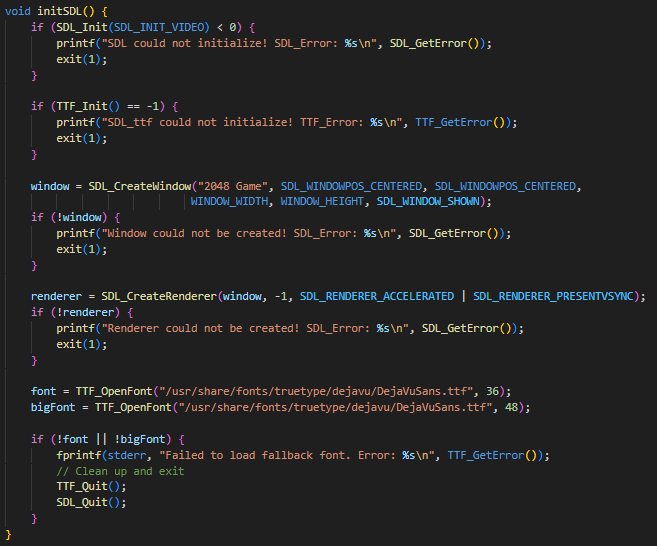
Documentatie jocul 2048

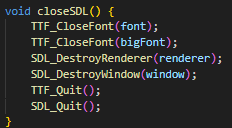
Acest proiect este o implementare a popularului joc 2048 în limbajul de programare C. Jocul implică combinarea plăcilor cu același număr pentru a crea o placă cu suma celor două. Obiectivul este să ajungi la placa 2048.

initSDL()



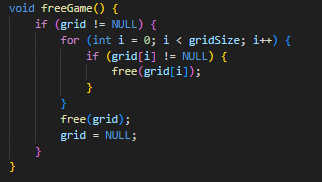
* Inițializează subsistemul video SDL
* Inițializează SDL\_ttf pentru afișarea textului
* Creează o fereastră cu titlul "2048 Game" și dimensiunile specificate
* Creează un renderer cu accelerare și VSync
* Încarcă fonturile din sistem (DejaVu Sans) în două dimensiuni diferite
* Încheie programul cu eroare dacă orice inițializare eșuează

closeSDL()



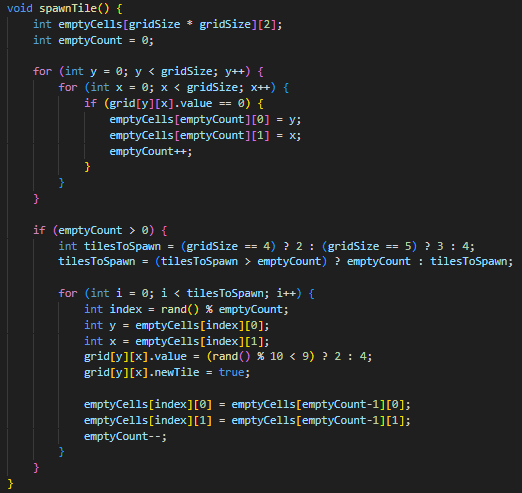
* Închide resursele pentru fonturi
* Distruge renderer-ul și fereastra
* Oprește subsistemele TTF și SDL
* Ar trebui apelat înainte de încheierea programului

freeGame()



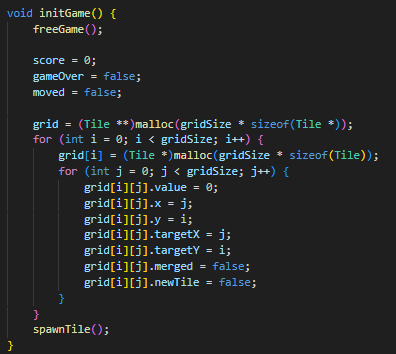
* Eliberează în siguranță memoria alocată pentru gridul de joc
* Gestionează eliberarea imbricată (mai întâi rândurile, apoi pointerul grid)
* Setează pointerul grid la NULL după eliberare
* Apelat când se schimbă dimensiunea gridului sau când se încheie jocul

spawnTile()



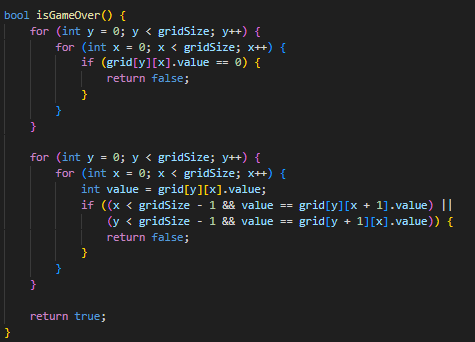
* Colectează toate coordonatele celulelor goale într-un array
* Generează 2-4 piese noi (în funcție de dimensiunea gridului) în locații aleatoare libere
* 90% șansă pentru o piesă de valoare 2, 10% șansă pentru o piesă de 4
* Marchează piesele noi cu flagul newTile
* Gestionează cazurile când rămân puține spații libere

initGame()



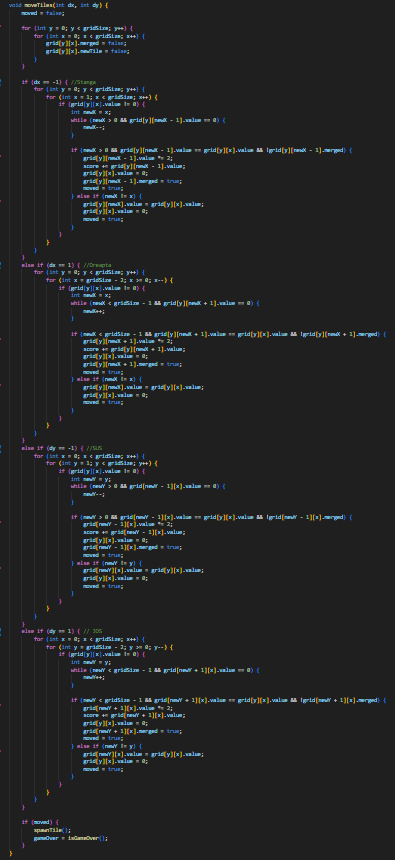
* Eliberează orice joc existent prin freeGame()
* Resetează scorul și stările de joc
* Alocă memorie pentru noul grid
* Inițializează toate celulele gridului cu valoarea 0
* Generează primele piese prin spawnTile()

isGameOver()



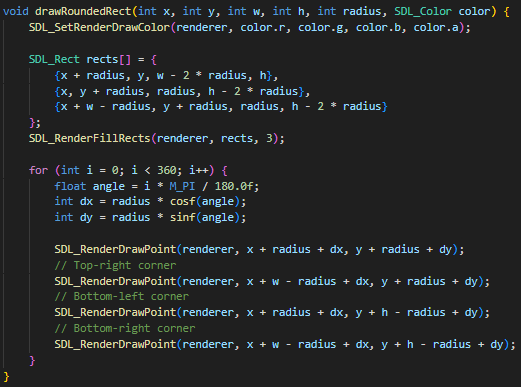
* Verifică dacă mai există celule goale în grid
* Verifică dacă mai există mutări valide (posibilități de combinare a pieselor)
* Returnează true numai dacă niciuna din condițiile de mai sus nu este îndeplinită

moveTiles()



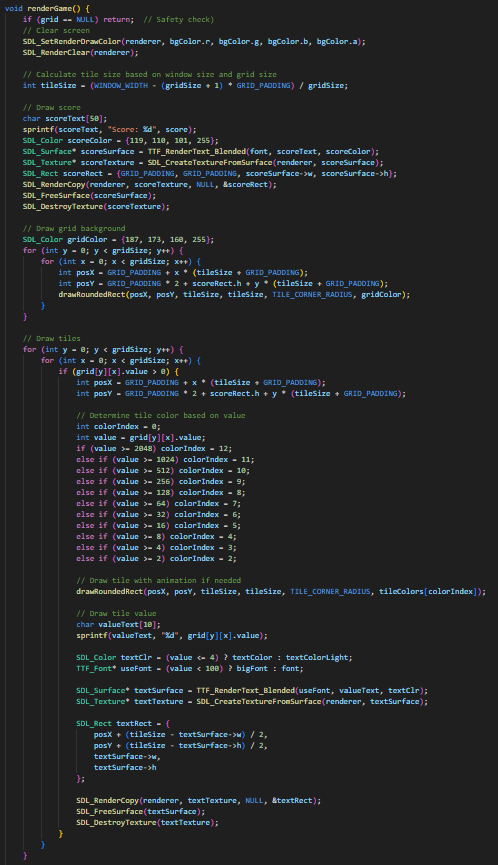
* dx: Direcția orizontală (-1 pentru stânga, 1 pentru dreapta)
* dy: Direcția verticală (-1 pentru sus, 1 pentru jos)  
  **Returnează**: Nimic (void)  
  **Descriere**:
* Resetează stările de fuziune și de piesă nouă
* Implementează logica de mișcare pentru fiecare direcție
* Combină piese cu aceeași valoare dacă sunt adiacente
* Actualizează scorul la fiecare combinare
* Generează piese noi dacă s-a produs o mișcare validă
* Verifică dacă jocul s-a terminat după mutare

drawRoundedRect()



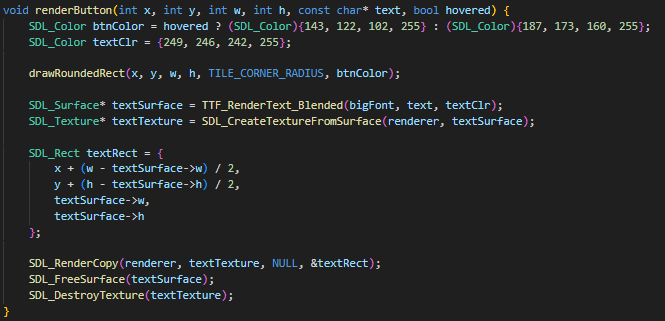
* x, y: Poziția în fereastră
* w, h: Lățimea și înălțimea
* radius: Raza rotunjirii colțurilor
* color: Culoarea de umplere  
  **Returnează**: Nimic (void)  
  **Descriere**:
* Desenează un dreptunghi compus din 3 zone: un dreptunghi central și două capete
* Completează colțurile cu arce de cerc pentru efectul rotunjit
* Folosește trigonometrie (sin/cos) pentru calculul punctelor pe cerc
* Utilizează SDL\_RenderFillRects pentru eficiență

renderGame()



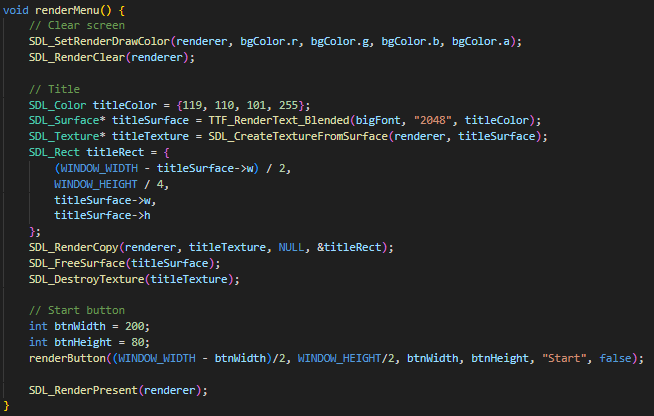
* Curăță ecranul cu culoarea de fundal
* Afișează scorul curent în partea superioară
* Desenează gridul de bază cu celule goale
* Iterează prin grid și desenează fiecare piesă:
  + Alege culoarea în funcție de valoarea piesei
  + Afișează valoarea numerică centrată în piesă
  + Folosește fonturi diferite pentru valori mici/mari
* Afișează mesaj "Game Over" transparent peste tot dacă jocul s-a terminat
* Actualizează ecranul final cu SDL\_RenderPresent

renderButton()



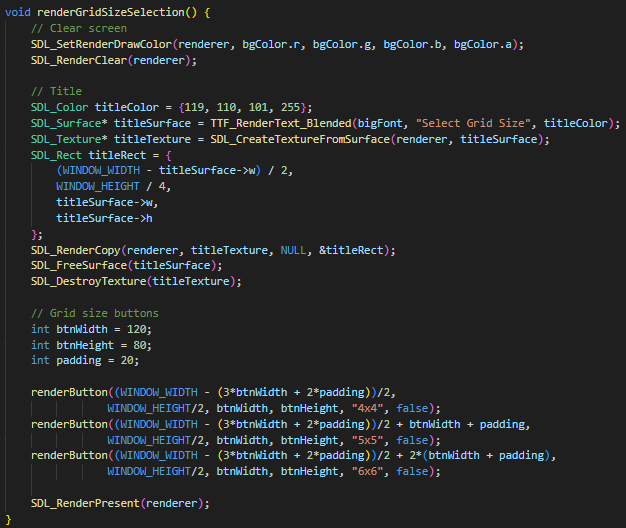
* x, y: Poziția butonului
* w, h: Dimensiunile butonului
* text: Textul de afișat
* hovered: Starea de hover (schimbă culoarea)  
  **Returnează**: Nimic (void)  
  **Descriere**:
* Desenează un dreptunghi rotunjit cu culoare diferită la hover
* Randrează textul centrat în buton
* Folosește fontul bigFont pentru text
* Culorile sunt predefinite pentru coerență vizuală

renderMenu()



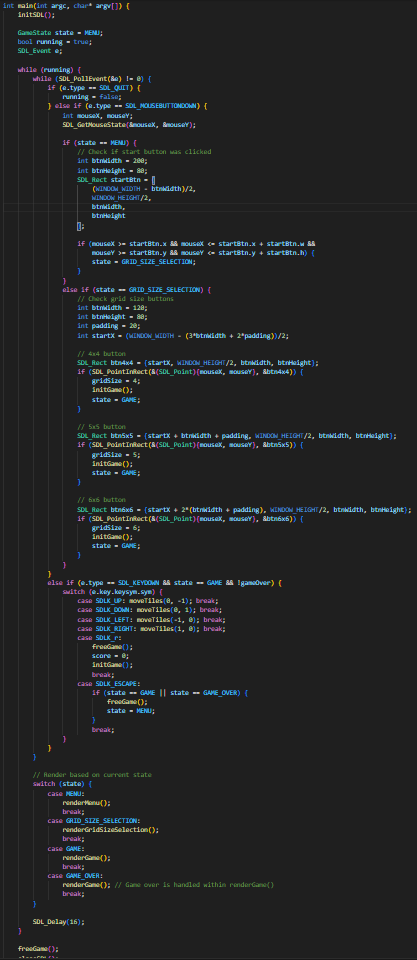
* Desenează titlul "2048" centrat în partea superioară
* Adaugă un buton mare "Start" în centrul ecranului
* Folosește culori consistente cu tema jocului
* Toate elementele sunt poziționate relativ la dimensiunile ferestrei

renderGridSizeSelection()



* Afișează titlul "Select Grid Size" în partea superioară
* Desenează trei butoane pentru dimensiuni: 4x4, 5x5 și 6x6
* Butoanele sunt aliniate orizontal cu spațiere egală
* Folosește aceeași temă vizuală ca meniul principal

Main()



* Inițializează SDL și starea jocului
* Pornește bucla principală de joc:
  + Gestionează evenimentele de intrare
  + Actualizează starea jocului
  + Randează conținutul corespunzător stării curente
* Gestionează tranzițiile între stări (meniu, joc, game over)
* Eliberează resursele la încheiere
* Implementează controalele:
  + Tastele săgeți pentru mișcare
  + R pentru restart
  + ESC pentru revenire la meniu

STRUCTURI DE DATE

**Tile:**

* value: Valoarea piesei (0 pentru celulă goală)
* x, y: Poziția curentă în grid
* targetX, targetY: Poziția țintă (pentru animații)
* merged: Flag pentru piesele recent combinate
* newTile: Flag pentru piesele nou generate\

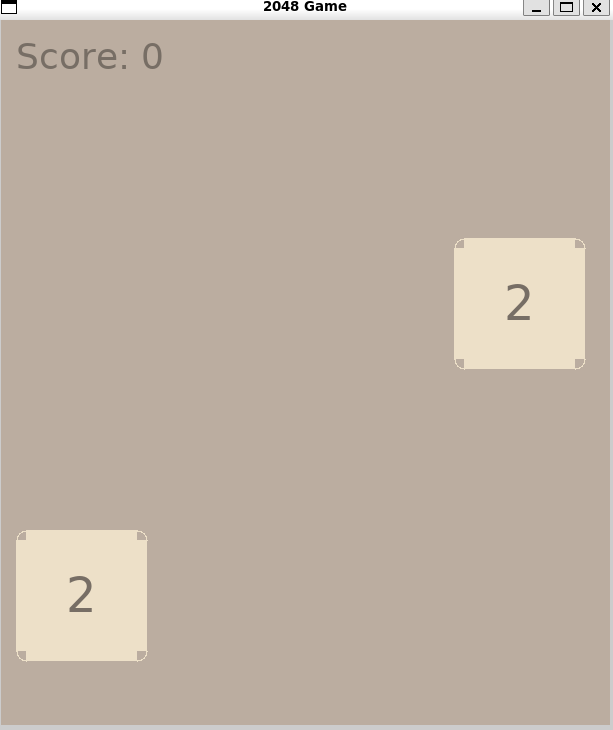
**GameState:**

* MENU: Ecranul principal
* GRID\_SIZE\_SELECTION: Selectarea dimensiunii gridului
* GAME: Jocul activ
* GAME\_OVER: Starea de final de joc

**Fluxul Jocului**

1. Inițializare SDL și resurse
2. Afișare meniu principal
3. Selectare dimensiune grid
4. Inițializare joc cu grid gol + 2-4 piese
5. Buclă principală:
   * Citire input
   * Actualizare stare joc
   * Randare
6. Detectare game over
7. Revenire la meniu sau restart

**Preview Game:**

**Instalare**

1. Clonează depozitul:

    git clone https://github.com/PatrikDav/ProiectTP.git

2. Compilează proiectul:

    gcc -Wall -o 2048 2048.c -lSDL2 -lSDL2\_ttf -lm

**Utilizare**

Rulează jocul compilat:

./2048

**Controale**

- Folosește tastele SAGETI ( ^, <, >, ˅) pentru a muta plăcile.

- Combină plăcile cu același număr pentru a crea o nouă placă cu suma lor.

- Încearcă să ajungi la placa 2048 pentru a câștiga jocul.

- Setează dimensiunea boardului la începutul jocului.