2013

LUCRARE PENTRU ATESTAREA COMPETENȚELOR PROFESIONALE

Tetris

Elev: Nicolae Berendea

Profesor coordonator: Mihaela Burlacu

Colegiul Național “Octavian Goga”

Cuprins

Motivul alegerii temei

Încă de la început m-am decis că vreau să fac un joc în C/C++, prin care să văd dacă am reușit să învăț C/C++ la un nivel de care să fiu mulțumit. Nu mi-a fost greu să mă hotărăsc asupra jocului. Am facut o clonă Tetris deoarece este un joc foarte popular și în acel moment practic nu avusesem niciun contact cu vreo bibliotecă grafică și nu eram sigur dacă voi reuși.

Am avut ca opțiuni urmatoarele biblioteci grafice: Allegro, SDL și SFML. Am ajuns să aleg SDL deoarece SDL este alcătuit din mai multe module, în timp ce Allegro este construit ca un tot unitar. Am evitat SFML deoarece am apreciat că este o biblioteca prea tânără. Nu cred că a fost o alegere greșită deoarece SDL dispune de multe extensii care reușesc să îndeplinească orice sarcină. Totuși pe viitor doresc sa încerc și alte biblioteci.

Descrierea jocului

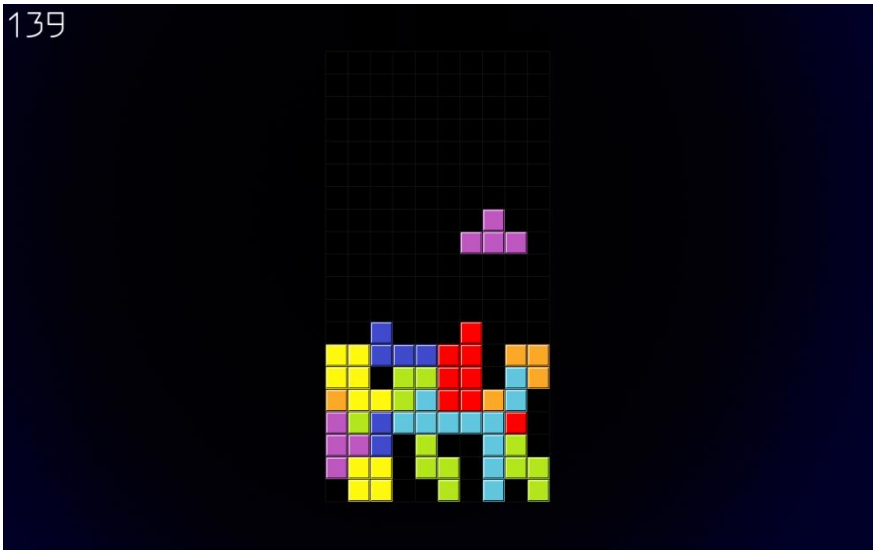
În primul rând, Tetris este un joc ce a aparut în anul 1984. Tetromino-urile sunt obiectele ce coboară în timpul jocului. Jucatorul are la dispoziție o zona de joc în care poate roti sau muta la stânga respectiv dreapta aceste tetromino-uri. Scopul jocului este de a umple cât mai multe linii.

Jocul a fost construit astfel încat să ruleze la o rezoluție standard de 640x480 sau la rezoluția la care este setat monitorul (fullscreen).

Când este rulat, apare un meniu din care se poate accesa jocul propriu zis. Am ales ca fundal un degrade radial închis la culoare (generat in Photoshop), iar logo-ul are de asemea culori închise. În meniu se poate naviga cu ajutorul săgeților sus-jos. Pentru a alege o opțiune se apasă tasta Enter.

Când se intra in jocul propriu zis apare pe ecran o grilă cu bare gri ce conține pătrate negre. În această grila vor coborî tetromino-urile. Pentru a controla tetromino-ul, se folosesc tastele stânga-dreapta (pentru a-l muta spre stânga respectiv dreapta), jos pentru a accelera coborarea acestuia sau sus pentru a roti tetromino-ul. In colțul din

stânga sus, se poate vedea scorul scris cu alb. Scorul crește cu o unitate la fiecare tetromino care ajunge jos și cu 2^nr_linii_umplute dacă, atunci când un tetromino nu mai poate coborî, o linie este umplută. Fundalul este același ca și în meniu. Se poate reveni la meniu cu una din tastele: ENTER, ESCAPE, BACK-SPACE.



Jocul crește in dificultate pe măsura ce jucătorul completează liniile. După ce jucătorul pierde, jocul revine la meniu, de unde se poate reîncepe.

Jocul dispune și de un efect audio atunci când tetromino-urile sunt rotite.

Detalii tehnice

Codul este împărțit in 7 fisiere: main.cpp și 3 perechi header - sursa cpp.

În fișierul **main.cpp** este instanțiată clasa Game după care este rulată metoda run() a acestei clase.

game.h conține definiția clasei Game, iar game.cpp conține implementarea metodelor. Aceasta clasă include metodele care schimbă fluxul de execuție, manipulează datele și modifică informațiile de pe ecran. Metoda run() constă într-o buclă care apelează metoda corespunzătoare stării care urmează (avem 2 stări: MENU și PLAY) până când utilizatorul închide jocul. Aceasta este bucla care controlează jocul (îl oprește când variabila quit devine true).

```
void Game::run(){
    while (!quit){
        switch (state){
            case MENU:
                menu();
                break;
            case PLAY:
                play();
                break;
        }
    }
}
```

Metoda menu() apelează draw_menu() care desenează fundalul din timpul stării MENU. Fluxul de execuție rămâne în starea MENU până când utilizatorul schimbă starea sau părăsește jocul. Navigarea în meniu se face cu ajutorul săgeților sus-jos. Fiecare apăsare a uneia din săgeți actualizează culoarea intrărilor de meniu.

Metoda play() are mai multe inițializări de făcut: este construită o matrice goală (bordată pe margini și în partea

inferioară) ce memorează pozițiile ocupate din grilă, sunt create tetromino-urile în funcție de datele din fișierul tetrominos.txt, este inițializată coada de tetromino-uri cu 2 elemente (în cazul în care se dorește afișarea următorului tetromino; eu am eliminat această opțiune deoarece nu mi-a plăcut cum arăta pe acest fundal), iar în final este apelată metoda draw_play() care desenează fundalul din timpul stării PLAY. În continuare, este folosită o buclă pentru a prelua evenimentele (butoanele apăsate) în vectorul keyboard, și pentru a muta tetromino-ul corespunzător.

```
while (!quit_play){
        int start=SDL_GetTicks(),end;
        SDL_Event event;

        while (SDL_PollEvent(&event)){
            switch (event.type){
                  /*aici este actualizat vectorul keyboard cu butoanele
                  apasate */
            }
        }
                  /*aici este calculata noua pozitie a tetromino-ului;
                  metoda returneaza true daca nu mai este loc ca un alt
                  tetromino sa intre in grila */
        if (Tetromino_move(fig,grid,keyboard,tetrominos,1)){
            quit_play=1;
        }
                  /*aici este desenat tetromino-ul pe ecran; el a fost deja
                  sters cand i s-a calculat noua pozitie; daca partea
                  superioara a tetromino-ului a iesit din grila in urma
                  unei rotiri a acestuia, nu va fi desenata decat partea
                  care se afla in grila */
        if (fig[0]->update){
            if (fig[0]->rotated){
                  /*deseneaza daca a fost rotit
                        ...
                  */
            }
            else{
                  /*deseneaza daca a fost mutat
                        ...
                  */
            }
        }
                  /*aceasta bucla, ca si bucla din menu() este regulata la
                  un numar fix de cadre pe secunda astfel incat jocul sa
                  ruleze lin cu un consum redus de resurse */
        end=SDL_GetTicks();
        if (end-start<frame)
            SDL_Delay(frame-end+start);
    }
```

tetromino.h definește clasele Tetromino, tetromino_list și tetromino_node, iar **tetromino.cpp** conține implementarea metodelor corespunzătoare.

Fiecare tetromino_node este un nod din lista tetromino_list. A fost necesar să memorez în fiecare nod lățimea, înălțimea, forma și reprezentarea grafică. Reprezentarea grafică are al doilea rând și a doua coloană din colțul din stânga sus de culoare albă, iar în colțul opus de culoare neagră pentru a da un aspect plăcut tetromino-urilor.

tetromino_list este o listă circulară de tetromino_node, fiecare nod reținând datele corespunzătoare unei răsuciri a unui tetromino.

Tetromino conține atât variabile necesare pentru a memora și actualiza poziția pe ecran, cât un pointer către un nod din lista circulară. Astfel, rotirea unui tetromino, va însemna avansarea la nodul următor.

Nu în ultimul rând, fișierele **functions.h** și **functions.cpp** conțin funcții necesare dar care nu sunt specifice unei clase. Aici am implementat funcția care permite jocului să ruleze la orice rezoluție:

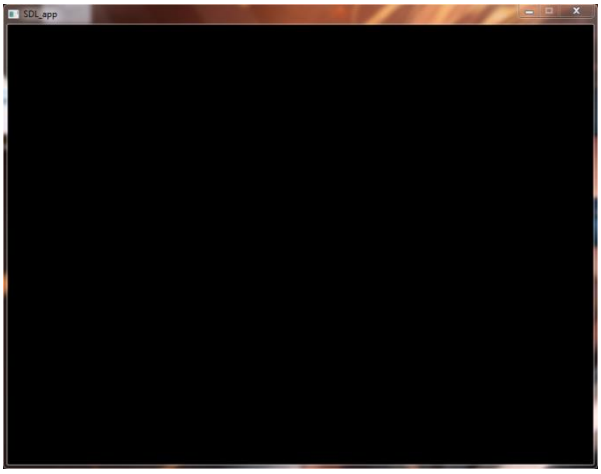
```
void resize_to(SDL_Surface *src, SDL_Surface *dst);
```

Această funție întinde sau restrânge pixelii suprafeței *src pe suprafața *dst. De asemenea, se încearcă atenuarea pixelării imaginii prin trecerea graduală de la culoarea unui pixel la culoarea pixelului următor dacă prin întinderea suprafeței apare un spațiu între cei doi pixeli.

```
void resize_to(SDL_Surface *src, SDL_Surface *dst){
    int srci,srcj,ldsti,ldstj,dsti,dstj,i,j;
            //este iterata fiecare linie
    for (srci=0,ldsti=-1;srci<src->h;++srci){
        dsti=srci*(dst->h-1)/(src->h-1);
        switch (dsti-ldsti){
            /*daca distanta dintre liniile de pixeli ai suprafetei *src
            copiati in *dst este 1, nu este necesara generarea unui degrade
            intre cele doua linii */
            case 1:
```

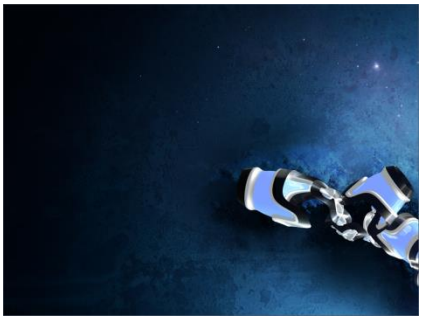
```
            for (srcj=0,ldstj=-1;srcj<src->w;++srcj){
                dstj=srcj*(dst->w-1)/(src->w-1);
                switch (dstj-ldstj){
              /*daca distanta intre coloanele de pixeli ai suprafetei
              *src copiati in *dst este 1, nu este necesara generarea
              unui degrade intre cele doua linii */
                    case 1:
                    //se copiaza pixelii
              /*daca distanta intre coloanele de pixeli ai suprafetei
              *src copiati in *dst este 0, atunci inseamna ca coloanele
              se suprapun deci noua coloana nu este copiata */
                    case 0:
                        break;
              //altfel este generat un degrade intre cele doua coloane
                    default:
                    //se genereaza degrade-ul
                }
                ldstj=dstj;
            }
        /*daca distanta dintre liniile de pixeli ai suprafetei *src
        copiati in *dst este 1, atunci inseamna ca liniile se suprapun
        deci noua linie nu este copiata */
        case 0:
            break;
        /*altfel este generat un degrade intre cele doua linii, iar pe
        fiecare linie introdusa, se aplica algoritmul aplicat in cazul
        in care dsti-ldsti=1*/
        default:
              //este generat gradientul intre linii si coloane
    }
    ldsti=dsti; /*inchidem forul*/}/*inchidem functia*/}
```

Folosind SDL, fereastra creată este neagră la început.



Pentru o scurtă demonstrație a modului de funcționare a funcției resize_to(), am copiat 2 imagini la rezoluțiile originale în această fereastră. Pe prima linie am pus capturi de ecran ale ferestrelor creeate și se poate vedea că prima imagine este tăiata, iar cea de-a doua este mai mică decât fereastra. Pe a doua linie se observă că prima imagine a fost

micșorată, iar cea de-a doua mărită astfel încat să aibă rezoluția ferestrei. De asemenea se observă că nu a fost păstrat raportul imaginii, acest lucru fiind dorit deoarece pozele au un raport apropiat de cel final (cel al imaginii care se vede pe ecran).





Tetromino-urile pot fi editate fără a recompila codul. Pentru a adăuga/elimina/edita tetromino-urile, se editează fișierul tetrominos.txt din directorul resources. In acest director găsim si logo-ul, background-ul, fontul folosit, și efectul audio care este redat cand sunt rotite tetromino-urile.

De asemenea, dacă se dorește modificarea numărului de cadre pe secunda sau a laținii respectiv înălțimii grilei, se pot modifica valorile macro-urilor definite la începutul fișierului tetromino.h.

Programul este cross-platform. Poate fi compilat și rulat pe orice pe orice sistem de operare.

Cerințe de sistem

Procesor: ---------- orice procesor de după PIII

RAM: ---------- 10MB

SO: ---------- toate SO suportate de SDL

Spațiu liber pe hard disk: ---------- 5MB

Bibliografie

Ernest Panzera, *Focus on SDL*

Site-urile:

http://www.cplusplus.com/

http://www.libsdl.org/cgi/docwiki.fcg