Tetris

23 noiembrie 2010

Jocul Tetris

Probabil cunoscut de toată lumea
Vom încerca o implementare în C

Pentru partea de grafică vom folosi biblioteca *SDL*

- Simple Directmedia Layer
- `http://www.libsdl.org/`

Vom aplica

- operatori pe biți
- lucrul cu matrici

Biblioteca SDL

Bibliotecă multimedia
Oferă acces la resursele video și audio ale sistemului
Conține funcții pentru lucrul cu tastura și mouse-ul

Pe scurt: utilă la construcția de jocuri

Tutorial pentru lucrul cu SDL din Code::Blocks

- `http://wiki.codeblocks.org/index.php?title=Using_SDL_with_Code::Blocks`

Primul program SDL

Afișează un ecran negru timp de 5 secunde
Evidențiază operațiile de bază necesare pentru a utiliza SDL

Operații necesare

- Inițializare SDL
  - Funcția *SDL_Init*
- Configurare mod video
  - Rezoluție și adâncime de culoare
  - Funcția *SDL_SetVideoMode*
- Bucla principală de program
  - În cazul nostru pauză de 5 secunde
  - Funcția *SDL_Delay*
- Eliberare resurse folosite
  - Funcția *SDL_Quit*

```c
#include <SDL/SDL.h>
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char **argv) {
  /* initializare SDL */
  if (SDL_Init(SDL_INIT_VIDEO) == -1) {
    printf("Failed to initialize SDL: %s.\n", SDL_GetError());
    exit(EXIT_FAILURE);
  }
  atexit(SDL_Quit);
  /* configurare mod video */
  SDL_Surface *screen =
      SDL_SetVideoMode(640, 480, 8, SDL_SWSURFACE);
  if (screen == NULL) {
    printf("Failed to set video mode: %s.\n", SDL_GetError());
    exit(EXIT_FAILURE);
  }
  /* bucla principala de program (pauza de 5 secunde) */
  SDL_Delay(5000);
  /* eliberare resurse folosite */
  SDL_Quit();
  exit(EXIT_SUCCESS);
}
```

Un program ceva mai interesant

Desenarea unor pătrate colorate pe ecran

- La coordonate alese aleator
- De culori alese aleator

În SDL desenarea se face pe așa numite *suprafețe grafice*

- zone de memorie în care se poate *desena*
- la cerere pot fi afișate pe ecran

O suprafață grafică este caracterizată prin

- rezoluție (numărul de pixeli disponibili)
- adâncime de culoare (numărul de culori ce pot fi afișate)

La configurarea modului video se creează o astfel de suprafață grafică

```
SDL_Surface *screen =
    SDL_SetVideoMode(640, 480, 8, SDL_SWSURFACE);
```

Un program ceva mai interesant (2)

Desenarea unui pătrat

- Funcția *SDL_FillRect*
- Trebuie să specificăm
  - Suprafața grafică pe care desenăm
  - Coordonatele colțului din stânga-sus
  - Dimensiunile laturilor (cazul general e un dreptunghi)
  - Culoarea de desenare

Culoarea de desenare

- Se construiește din trei componente de culoare
  - Roșu
  - Verde
  - Albastru
- Se numește *cod RGB* (red, green blue)
- Se specifică intensitatea fiecărei componente de culoare
  - Uzual se specifică în hexa
  - Intensitate 0x00: componenta nu apare
  - Intensitate 0xFF: componenta e la valoare maximă

Un program ceva mai interesant (3)

Exemple de culori

- Culoarea *roșu*: 0xFF 0x00 0x00
- Culoarea *albastru*: 0x00 0x00 0xFF
- Culoarea *galben*: 0xFF 0xFF 0x00
- Culoarea *portocaliu*: 0xFF 0x80 0x40

Fiecare componentă de culoare are intensitatea între 0x00 și 0xFF

- Înseamnă 256 de valori posibile

Avem trei componente de culoare (roșu, verde și albastru)

- Înseamnă 256*256*256=16777216 valori posibile

Este posibil ca suprafața grafică folosită să nu poată afișa atâtea culori

Se face o *mapare* de la codul RGB la o culoare suportată de suprafața grafică

- Funcția *SDL_MapRGB*

```
SDL_Rect r;
int i;
/* dimensiunea fiecarui patrat: 32x32 pixeli */
r.w = r.h = 32;
for (i = 0; i < 2000; i++) {
  /* plasam patratul curent la coordonate aleatoare */
  r.x = rand() % (SCREEN_WIDTH - r.w);
  r.y = rand() % (SCREEN_HEIGHT - r.h);
  /* alegem aleator valori pentru componentele de
     culoare: rosu, verde si albastru */
  Uint8 rosu = rand() % 256;
  Uint8 verde = rand() % 256;
  Uint8 albastru = rand() % 256;
  /* mapam componentele rosu, verde si albastru pe
     adancimea de culoare a ecranului */
  Uint32 culoare =
      SDL_MapRGB(screen->format, rosu, verde, albastru);
  /* desenam patratul curent */
  SDL_FillRect(screen, &r, culoare);
  /* fortam afisarea lui in fereastra grafica */
  SDL_Flip(screen);
}
SDL_Delay(5000);
```

Interacțiunea cu utilizatorul

În orice joc utilizatorul apasă pe taste, folosește mouse-ul
Programul trebuie să reacționeze la toate acestea

În SDL există conceptul de *evenimente*

- Pot fi generate de: tastatură, mouse, joystick, etc.

Funcția *SDL_PollEvent* verifică dacă asemenea evenimente au fost declanșate

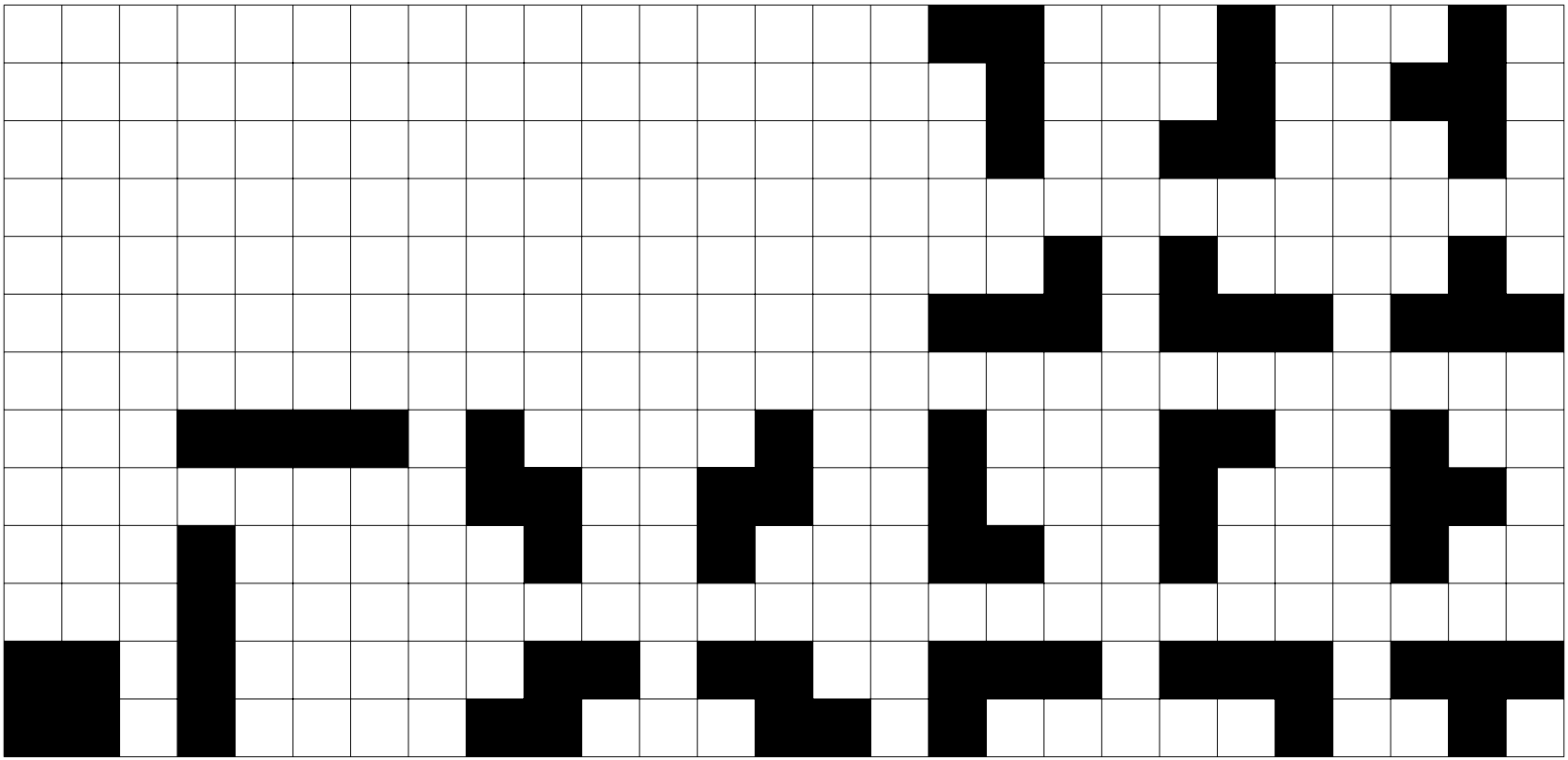
Pe noi ne interesează două tipuri de evenimente

- Apăsarea unei taste (*SDL_KEYDOWN*)
  - Dacă s-a apăsat ESCAPE ieșim din program
- Comandă de oprire a programului (*SDL_QUIT*)
  - Ieșim din program

```
SDL_Event event;
int gata = 0;
while (!gata) {
  /* desenam pe suprafata grafica din memorie */
  deseneaza_patrat(screen);
  /* afisam suprafata grafica pe ecran */
  SDL_Flip(screen);
  /* daca si cat timp utilizatorul apasa taste,
     reactionam la ele */
  while (SDL_PollEvent(&event)) {
    switch (event.type) {
      /* daca e apasare de tasta */
    case SDL_KEYDOWN:
      /* daca e tasta ESCAPE vom iesi din program */
      if (event.key.keysym.sym == SDLK_ESCAPE)
        gata = 1;
      break;
      /* daca se inchide fereastra vom iesi din program */
    case SDL_QUIT:
      gata = 1;
      break;
    }
  }
}
```

Jocul Tetris: Tipuri de piese

În jocul Tetris avem următoarele tipuri de piese



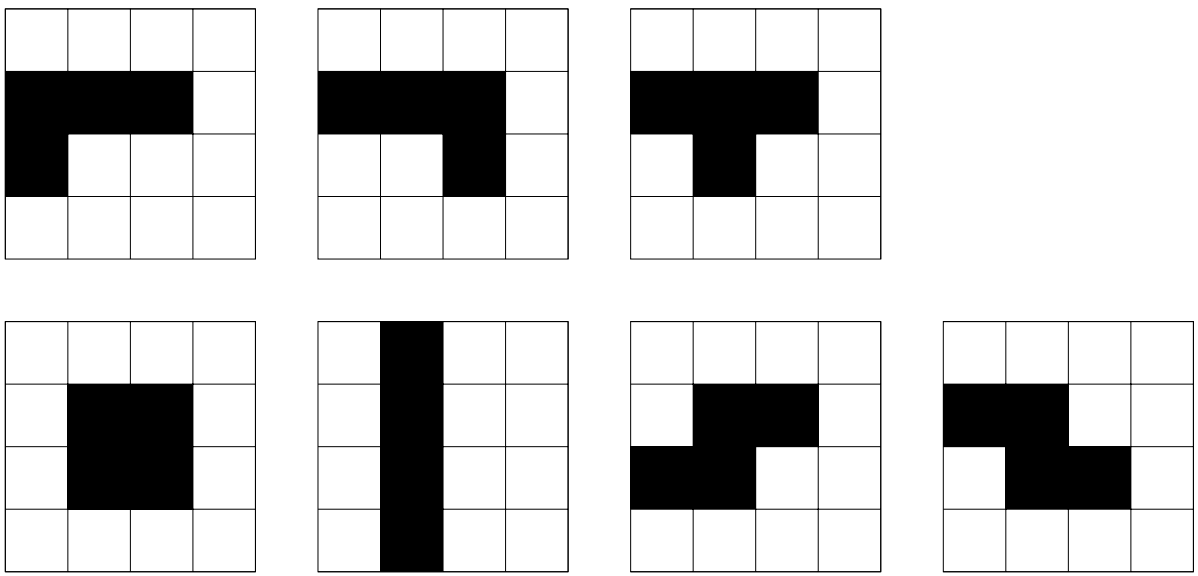
Cum le modelăm în program?

Jocul Tetris: Tipuri de piese (2)

Putem încadra orice piesă într-un pătrat de dimensiune 4x4

În plus o parte din piese se pot obține din altele prin rotire

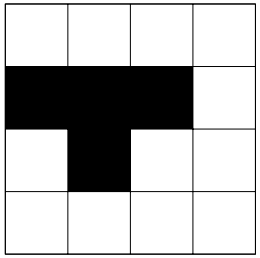
- Păstrăm doar piesele de bază
- Vom trata rotirea pe urmă



Aceeași întrebare: Cum le modelăm în program?

Jocul Tetris: Tipuri de piese (3)

Codificare pe biți



0	0	0	0
1	1	1	0
0	1	0	0
0	0	0	0

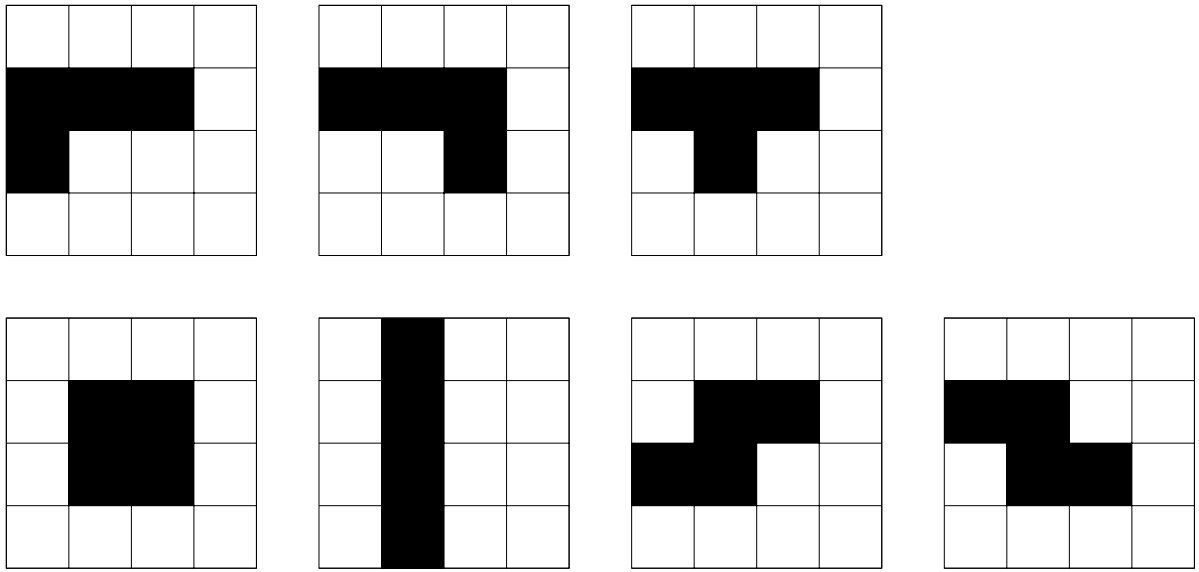
Avem 4x4=16 biți

- Încap exact pe un **unsigned short**

0	0	0	0	1	1	1	0	0	1	0	0	0	0	0	0

Valoarea 0x0E40 (în hexa)

Jocul Tetris: Tipuri de piese (4)



0x0E80 0x0E20 0x0E40
0x0660 0x4444 0x06C0 0x0C60

În cod:

```
unsigned short piese[] = { 0x0E80, 0x0E20, 0x0E40,
  0x0660, 0x4444, 0x06C0, 0x0C60
};
```

Desenarea unei piese

Parcurgere cu o mască de biți

- Pentru a vedea unde avem biți de 1
- Acolo unde găsim biți de 1, desenăm pătrate colorate
- Unde sunt biți de 0 desenăm pătrate negre

```
cod piesa:  0000111001000000
masca:      1000000000000000
            0100000000000000
            0010000000000000
            0001000000000000
                   ...
            0000000000000010
            0000000000000001
```

Operatorul ȘI pe biți între codul piesei și mască ne spune dacă avem 1 sau 0 pe poziția curentă

Desenarea unei piese (2)

Pentru a reface structura bidimensională (caroiaj de 4x4) facem parcurgere în două bucle **for**

În pseudocod:

```
unsigned short masca = 1 << (8*sizeof(unsigned short)-1);

for (i=0; i<4; i++) {
   for (j=0; j<4; j++) {
      if (cod & masca) deseneaza_colorat(i, j);
      else deseneaza_negru(i, j);
      masca >>= 1;
   }
}
```

```
void deseneaza_piesa(SDL_Surface * screen, int idx) {
  SDL_Rect r;
  Uint32 culoare;
  int i, j;
  unsigned short masca =
      1 << (8 * sizeof(unsigned short) - 1);
  r.w = r.h = 32;
  r.y = 100;
  for (i = 0; i < DIM_PIESA; i++) {
    r.x = 100;
    for (j = 0; j < DIM_PIESA; j++) {
      if (piese[idx] & masca)
        culoare =
            SDL_MapRGB(screen->format, 0xFF, 0x80, 0x40);
      else
        culoare =
            SDL_MapRGB(screen->format, 0x00, 0x00, 0x00);
      SDL_FillRect(screen, &r, culoare);
      r.x += r.w;
      masca >>= 1;
    }
    r.y += r.h;
  }
}
```

Rotirea pieselor

Lucrăm cu reprezentarea pe biți
Rotire în sensul acelor de ceas

a	b	c	d
e	f	g	h
i	j	k	l
m	n	o	p

se transformă în

m	i	e	a
n	j	f	b
o	k	g	c
p	l	h	d

Rotirea pieselor (2)

Altfel spus

a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p

se transformă în

m	i	e	a	n	j	f	b	o	k	g	c	p	l	h	d

Vom folosi operatorii de deplasare pe biți

Bitul *d* ajunge de pe poziția 12 pe poziția 0

- $(cod \;\&\; 1 << 12) >> 12$

Bitul *h* ajunge de pe poziția 8 pe poziția 1

- $(cod \;\&\; 1 << 8) >> 7$

...

Bitul *m* ajunge de pe poziția 3 pe poziția 15

- $(cod \;\&\; 1 << 3) << 12$

```
unsigned short roteste(unsigned short cod) {
  unsigned short cod_nou;
  cod_nou = (cod & 1 << 12) >> 12 |
      (cod & 1 << 8) >> 7 |
      (cod & 1 << 4) >> 2 |
      (cod & 1 << 3) |
      (cod & 1 << 13) >> 9 |
      (cod & 1 << 9) >> 4 |
      (cod & 1 << 5) << 1 |
      (cod & 1 << 1) << 6 |
      (cod & 1 << 14) >> 6 |
      (cod & 1 << 10) >> 1 |
      (cod & 1 << 6) << 4 |
      (cod & 1 << 2) << 9 |
      (cod & 1 << 15) >> 3 |
      (cod & 1 << 11) << 2 |
      (cod & 1 << 7) << 7 | (cod & 1 << 3) << 12;
  return cod_nou;
}
```

Continuarea

Va urma...