Gra w życie - sprawozdanie

Wojciech Musiatowicz i Michał Weiss

1 Wprowadzenie

Ogólne reguły gry

Gra toczy się na planszy podzielonej na kwadratowe komórki. Każda komórka ma ośmiu „sąsiadów" czyli komórki przylegające do niej bokami i rogami. Każda komórka może znajdować się w jednym z dwóch stanów: może być albo „żywa" (włączona), albo „martwa" (wyłączona). Stany komórek zmieniają się w pewnych jednostkach czasu. W kolejnych generacjach wszystkie komórki zmieniają swój stan dokładnie w tym samym momencie. Stan danej komórki zależy od liczby jej żywych sąsiadów.

Martwa komórka, która ma dokładnie 3 żywych sąsiadów, staje się żywa w następnej jednostce czasu (rodzi się)

Żywa komórka z 2 albo 3 żywymi sąsiadami pozostaje nadal żywa; przy innej liczbie sąsiadów umiera (z „samotności" albo „zatłoczenia").

Każda komórka ma dokładnie 8 sąsiadów(lewo,prawo,góra,dół i narożniki). Przypadki szczególne tj. pola na krawędziach planszy zostały rozwiązane zgodnie z poniższymi zasadami:

Pola z górnej krawędzi sąsiadują z polami z dolnej krawędzi (i odwrotnie).Natomiast lewa krawędź sąsiaduje z prawą. Komórki na rogach dodatkowo sąsiadują ze sobą nawzajem na następujących zasadach: Pole[0][0] sąsiaduje z pole [h][0] Pole[0][w] sąsiaduje z polem [h][w] (dla planszy o wymiarach (h+1) x (w+1))

2 Struktura programu

Funkcjonalność

Program został napisany w języku C. Symuluje grę w życie Johna Connowaya. Został przewidziany do używania w systemach Unix. Może być obsługiwany zarówno z linii komend, jak i graficznego interfejsu użytkownika.

Program w zależności od podanego układu komórek i wielkości planszy generuje pliki png zawierające konkretną generację lub podaną liczbę pierwszych generacji. Na utworzonych obrazkach biały kolor reprezentuje martwe komórki, natomiast kolor żywych komórek może ustalić użytkownik.

Wywołanie programu

./Gameoflife [-rand ¡height¿ ¡width¿] [-help] [*.cfg]
-help - program wyświetla pomoc;

-rand ¡height¿ ¡width¿ - losowanie konfiguracji planszy o danej wysokości i szerekości;
.cfg - argument wskazujący na plik zawierający planszę z zerową generacją.

Dane wyjściowe

Zapisane wybrane pliki png z generacjami w folderze /result.

Działanie programu

Program pobiera parametry z pliku do odpowiednich struktur. Następnie z kolejnego pliku czytana jest zerowa generacja. Na jej podstawie moduł symulujący tworzy kolejne generacje. W zależności od konfiguracji, przekazywana jest informacja do modułu, który stworzy i zapisze plik png.

Pliki, ich funkcjonalność oraz struktura

*.cfg

Plik konfiguracyjny, który można modyfikować.Nazwę pliku, z którego ładowana będzie pierwsza plansza można opcjonalnie podać jako argument linii wiersza poleceń. Domyślna nazwa pliku to "gen.cfg". Plik zawiera układ komórek, na podstawie której symulowane są kolejne generacje. Komórki reprezentuje macierz znaków plus i minus, gdzie "-" uchodzi za martwą komórkę, a "+" żywą.

config.cfg

Plik konfiguracyjny, który można modyfikować. Zawiera następujące opcje:
numberofgen - liczba generacji, które chcemy symulować;
print - numer generacji, którego graficzną reprezentację chcemy zobaczyć. Dla wartości "0" wszystkie generacje są eksportowane do plików png;
red,green,blue - przyjmują wartości od 0 do 255, które ustawiają kolor żywych komórek na obrazkach.

main.c

Główny moduł sterujący.

read.h

Plik nagłówkowy zawierający strukturę:

```
typedef struct {
    int height; // wysokość planszy
    int width; // szerokość planszy
    char **grid; // macierz przechowująca żywe i martwe komórki
    int numberofgen;
```

```
    int print;
    int red;
    int green;
    int blue;
}option;
```
oraz prototypy funkcji
```
int read_cfg(FILE* config, option *cfg);
int read_grid(FILE* grid, option *cfg);
```

read.c

Moduł odpowiedzialny za czytanie informacji z plików config.cfg i gen.cfg. Szerokość planszy czytana jest z pierwszego rzędu w pliku. Nawet, jeśli szerokość w poszczególnych rzędach się nie zgadza, będzie ona równa liczbie komórek w pierwszym rzędzie.

gen.h

Plik nagłówkowy do modułu symulującego kolejne generacje.

gen.c

Moduł symulujący generacje. Zawiera funkcje:
*int fill (option *cfg)* - odpowiedzialna za poprawne ustawienie komórek na krawędziach planszy, by funkcja generate() mogła sprawdzić ich sąsiedztwo. Funkcja wypełnia zewnętrzną powłokę macierzy hxw, rozszerzając ją tym samym do macierzy (h+2)x(w+2). Dodatkowe pola planszy gry symbolizują sąsiadów pól znajdujących się na krawędziach zasadniczej planszy (rozmiaru hxw).
*int generate(option *cfg)* - odpowiedzialna za sprawdzenie sąsiedztwa każdej komórki i zapisanie tablicy dwuwymiarowej temp[][]. Następnie dane z tablicy nadpisywane są do grid[][].

imagen.h

Plik nagłówkowy do modułu programującego obrazki.

imagen.c

Moduł programujący obrazki png połączony z biblioteką lodepng. Zawiera funkcje:
int makeimage(option *cfg,int nrgen)
W tej funkcji odbywa się projektowanie przyszłego obrazka, zanim jego szkielet zostanie przesłany do biblioteki lodepng. Najpierw określany jest parametr k - od niego zależy, jaką szerokość i wysokość w pikselach będzie miała jedna komórka na obrazku. Domyślnie wynosi 10, natomiast zmiejsza się dla większych

wymiarów planszy, by obrazki były szybciej generowane. Dla wymiarów z przedziału (200,800) k=5, a dla większych, niż 800 k=2.
Następnie inicjowana jest tablica *unsigned char* image*, która jest szkieletem obrazka. W następującym kodzie następuje zapełnienie obrazka białymi pikselami:

```
for(y = 0 ; y ⟨< height; y + +)
    for(x = 0; x ¡ width; x++) {
       image[4 * width * y + 4 * x + 0] = 255;
       image[4 * width * y + 4 * x + 1] = 255;
       image[4 * width * y + 4 * x + 2] = 255;
       image[4 * width * y + 4 * x + 3] = 255;
    }
```

W następnym zagnieżdżeniu pętli analogicznie następuje zakodowanie pikseli odpowiadającym żywym komórkom.

void encodeOneStep(const char* filename, const unsigned char* image, unsigned width, unsigned height) - funkcja przekazująza dane do biblioteki lodepng. Otrzymuje nazwę pliku, tablicę z zaprogramowanym obrazkiem oraz wymiary obrazka w pikselach.

lodepng.h oraz lodepng.c

Moduł generujący obrazki png. Jest to zewnętrzna biblioteka udostępniona do dowolnego użytku na stronie *http://lodev.org/lodepng/*

Makefile

Plik kompilujący program.

clean.sh, install.sh

Skrypty umożliwiające korzystanie z programu bez linii komend w środowiskach typu Unix. Clean.sh usuwa zawartośc folderu result, a install.sh kompiluje pliki.