Ruchome Piaski v1.0

Maciej Matyka
email: maq@panoramix.ift.uni.wroc.pl

Uniwersytet Wrocławski
Wydział Fizyki i Astronomii
Fizyka II BIS

16 lutego 2002 roku

Spis treści

Streszczenie

W dokumentacji tej przedstawiam ogólne założenia modelu użytego do symulacji piasku przy użyciu metod stosowanych w symulacjach opartych na automatach komórkowych. Oprócz tego w pracy tej znajduje się opis funkcji programu "Ruchome Piaski" napisanego na bazie przedstawionych algorytmów. W ostatniej części znajdują się przykłady gotowych symulacji stworzonych opisywanym programem.

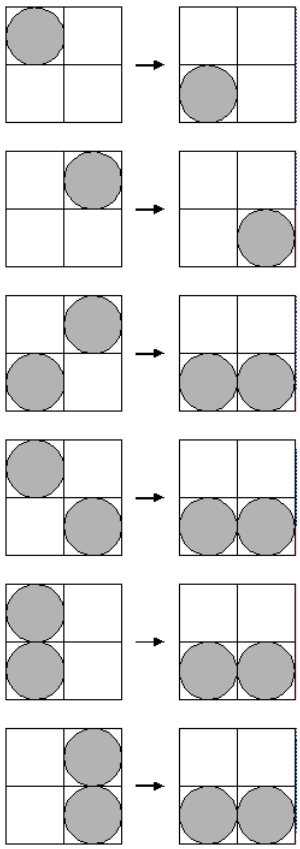
1 Teoria

1.1 Ogólne założenia

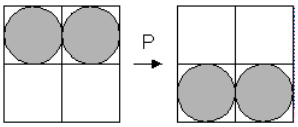
Wykonanie symulacji piasku wymaga poczynienia kilku założeń co do modelu. Po pierwsze - zakładamy, że piasek składa się z równej wielkości ziaren o jednakowym kształcie, jednakowym ciężarze. Zakładamy, że cała symulacja odbywa się w polu grawitacyjnym, jednakże nie definiujemy przyspieszenia jako takiego - cząstki opadające będą miały stałą prędkość. Już w tym miejscu widać, jak duże uproszczenia stosujemy. Definiujemy przestrzeń zdefiniowaną jako siatka komórek, które mogą przyjąć tylko wartości 0 lub 1 (0 - ziarna w komórce nie ma, 1 - ziarno jest).

1.2 Pętla czasowa

Pętla czasowa programu jest bardzo prosta i sprowadza się do implementacji ogólnych zasad usypywania się piasku. Zasady te - przedstawione na rysunku (1). Już tu widać jaka jest ogólna idea naszego rozwiązania - pracować będziemy przy pomocy narzędzia $2x2$, którego przejście przez całą tablicę spowoduje co najwyżej usypanie każdego z ziaren o jedną jednostkę w dół. Zasady te zawierają też w sobie sytuacje, gdy ziarno piasku usypuje się na boki. Dodatkowo, dla zwiększenia realizmu (pomysł zaczerpnięty z [1]) sytuację z rysunku (2) trochę komplikujemy. Otóż wprowadzamy warunek na opadanie podwójnego ziarna piasku - opadnie ono, ale z pewnym z góry określonym prawdopodobieństwem P. Takie potraktowanie podwójnych ziaren piasku zwiększa realizm symulacji. Rozsądną wydaje się wartość ok. $P = 0.5$.



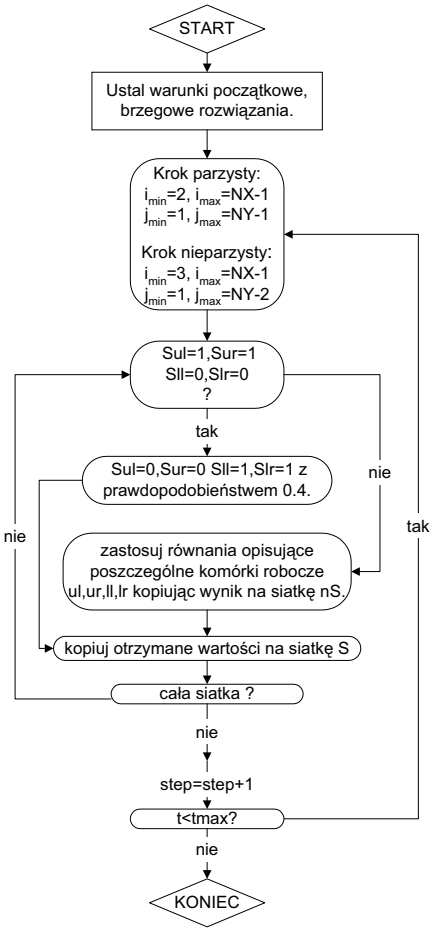
Rysunek 1: Ogólne zasady ruchu ziaren piasku.



Rysunek 2: Dwa zlepione ziarna piasku.

1.3 Algorytm działania programu "Ruchome Piaski".

Algorytm działania programu przedstawia się nadzwyczaj prosto i taki w zasadzie jest.



Rysunek 3: Algorytm programu "Ruchome Piaski".

Na rysunku (3) w formie diagramów pokazana jest struktura programu. Podział kroku czasowego na dwa oddzielne (dla kroków nieparzystych i parzystych z osobna) został wprowadzony, bo tego wymaga charakter zagadnienia. Co drugi krok czasowy należy działać komórką roboczą $2x2$ wedle zasad z rysunku (1) obniżając startowy punkt o jedną jednostkę w dół. Gdybyśmy bowiem startowali zawsze z tego samego miejsca - żadne ziarno piasku nie opadło by o więcej niż jedną jednostkę w dół!

1.3.1 Fragmenty kodu źródłowego

Zgodnie z podanym algorytmem zapiszemy teraz kompletną implementację kroku czasowego w języku c. Warto zdefiniować często używane oznaczenia komórek roboczych, by bezpośrednio w pętli czasowej używać zapisanych wcześniej równań, jak pokazujemy poniżej:

```
#define Gul G[i][j] #define Gur G[i+1][j]

#define Sul S[i][j] #define Sur S[i+1][j] #define Sll S[i][j-1]
#define Slr S[i+1][j-1]
```

Zgodnie z tą definicją krok czasowy przebiegu po całej siatce można zapisać w formie, jak poniżej:

```
// dla kroku czasowego parzystego pętla dla y(NY-1;1) i x(2;NX-1)
// dla kroku nieparzystego pętla dla y(NY-2;1) i x(3;NX-1)

    for(j=NY-1-(step%2);j>1;j-=2)
        for(i=2+(step%2);i<NX-1;i+=2)

// nie wykonujemy operacji na komórkach brzegowych
        if(!Gul && !Gur)
        {

            if(Sul==C_SND && Sur==C_SND && Sll!=C_SND && Slr!=C_SND)
            {
                if(rand()/(float)RAND_MAX>0.4)
                {
                    Sll=Sul;
                    Slr=Sur;

                    Sul=C_EMP;
                    Sur=C_EMP;
                }
            } else
            {
                nSul = Sul*(Gul + (1-Gul)*Sll*(Slr + (1-Slr) * Sur));
                nSur = Sur*(Gur + (1-Gur)*Slr*(Sll + (1-Sll) * Sul));
                nSll = Sll+(1-Sll)*(Sul*(1-Gul)+(1-Sul)*Sur*(1-Gur)*Slr);
                nSlr = Slr+(1-Slr)*(Sur*(1-Gur)+(1-Sur)*Sul*(1-Gul)*Sll);

                Sul = nSul;
                Sur = nSur;
                Sll = nSll;
                Slr = nSlr;
            }
        }

// zwiększenie indeksu kroku czasowego
        step++;
```

2 Program "Ruchome Piaski"

2.1 Wymagania sprzętowe

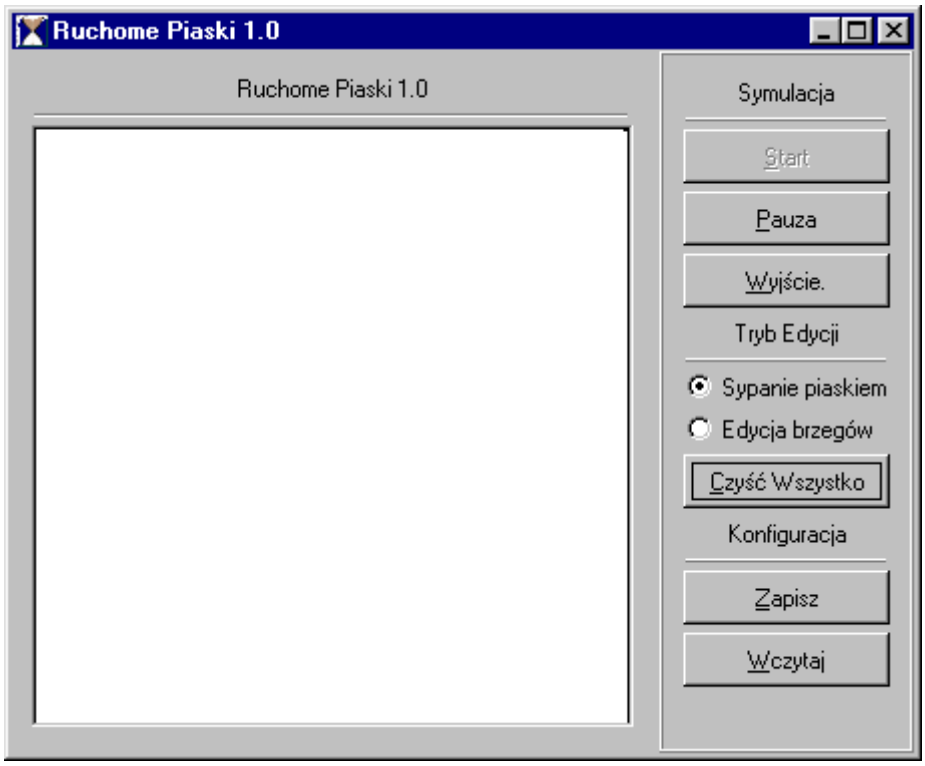
Wymagania sprzętowe programu "Ruchome Piaski" nie są duże, jednak szybki procesor bardzo dobrze wpływa na jakość symulacji. Mamy tu do czynienia z animacją w czasie rzeczywistym, co znacznie wpływa na szybkość animacji generowanej przez program. Wersja dostarczona do konkursu jest skompilowana pod system operacyjny Windows (98/2000/NT/Millenium/XP).

2.2 Kod źródłowy programu

Do programu dołączony jest pełen kod w języku C++ wraz z plikami konfiguracyjnymi Visual C++. Program można bez problemu skompilować pod systemy operacyjne Linux, bo do stworzenia interfejsu graficznego wykorzystana została biblioteka FOX. Do wizualizacji danych wykorzystany został port OpenGL zaimplementowany w bibliotece FOX. odatkowo użytych zostało kilka funkcji biblioteki GLut.

2.3 Opis programu

2.3.1 Okno główne programu



Rysunek 4: Okno główne programu "Ruchome Piaski".

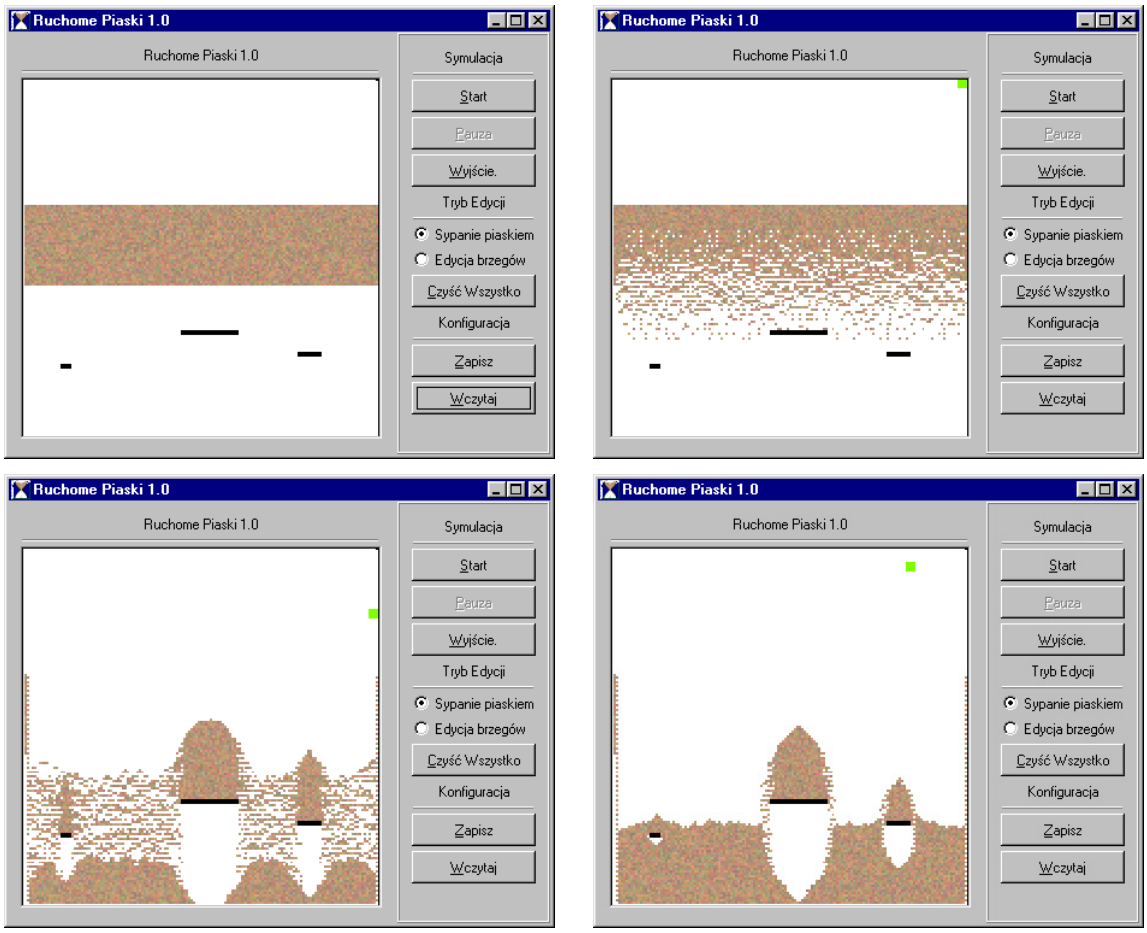
W oknie głównym programu po stronie lewej znajduje się obszar wydzielony na wizualizację danych, po stronie prawej znajduje się pasek z przyciskami i opcjami programu.

2.3.2 Obsługa programu

Obsługa jest bardzo prosta, przy pomocy myszki dokonujemy zmian w geometrii problemu w części wizualizacyjnej okna programu. W zależności od ustawienia opcji po prawej stronie (znaczniki *"Sypanie piaskiem"* i *"Edycja brzegów"*) możemy dodać piasek, lub edytować komórki brzegowe. Przycisk *"Czyść Wszystko"* powoduje wyczyszczenie geometrii problemu - wszystkie komórki w modelu stają się puste. Warto wspomnieć też o użytecznej opcji *"Pauza"*, którą warto używać przy definiowaniu nowych geometrii - sypanie piasku przy włączonej opcji *"Pauza"* nie spowoduje jego opadania, dopiero ponowne wciśnięcie *"Start"* spowoduje ruch. Stworzone konfiguracje można zapisać na dysku do późniejszego wykorzystania (opcje *"Zapisz"* i *"Wczytaj"*).

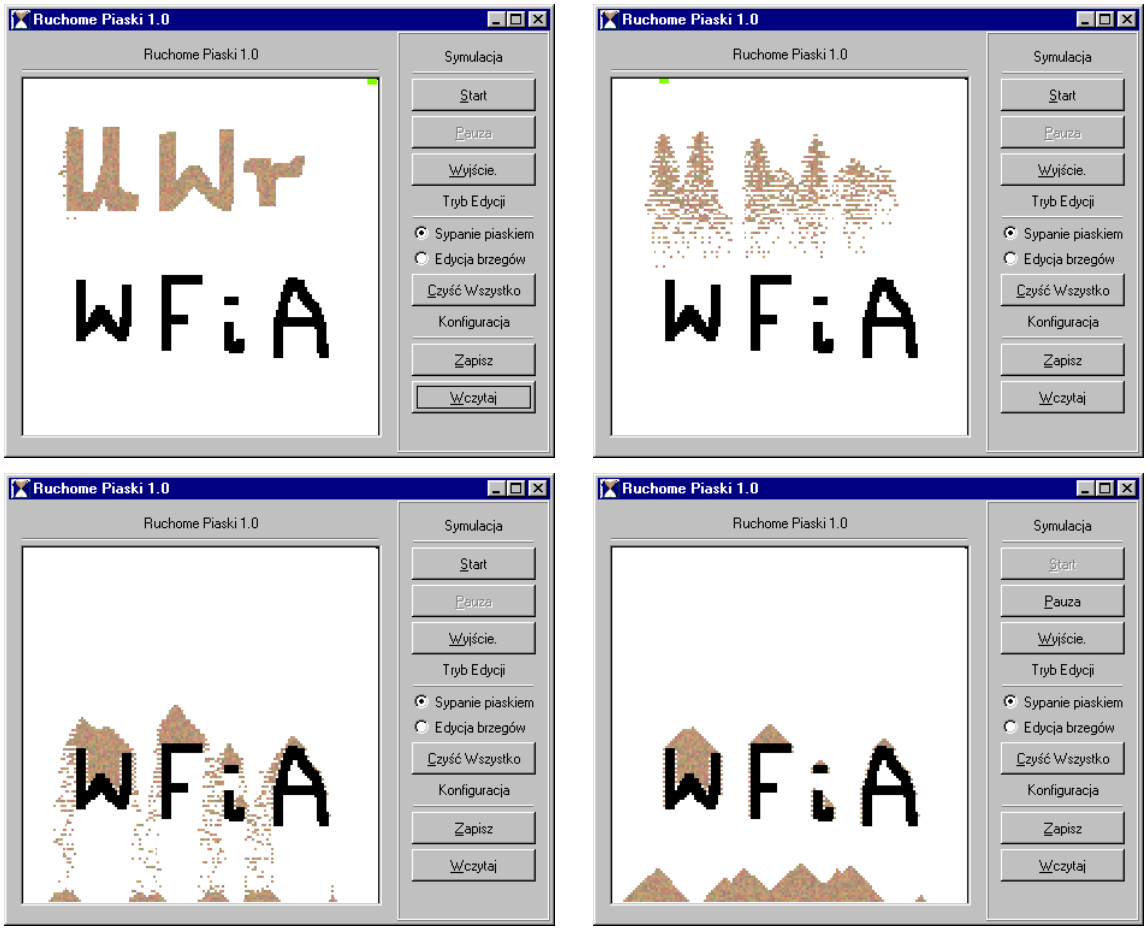
3 Efekty działania

3.1 Piach usypujący się na przeszkodach



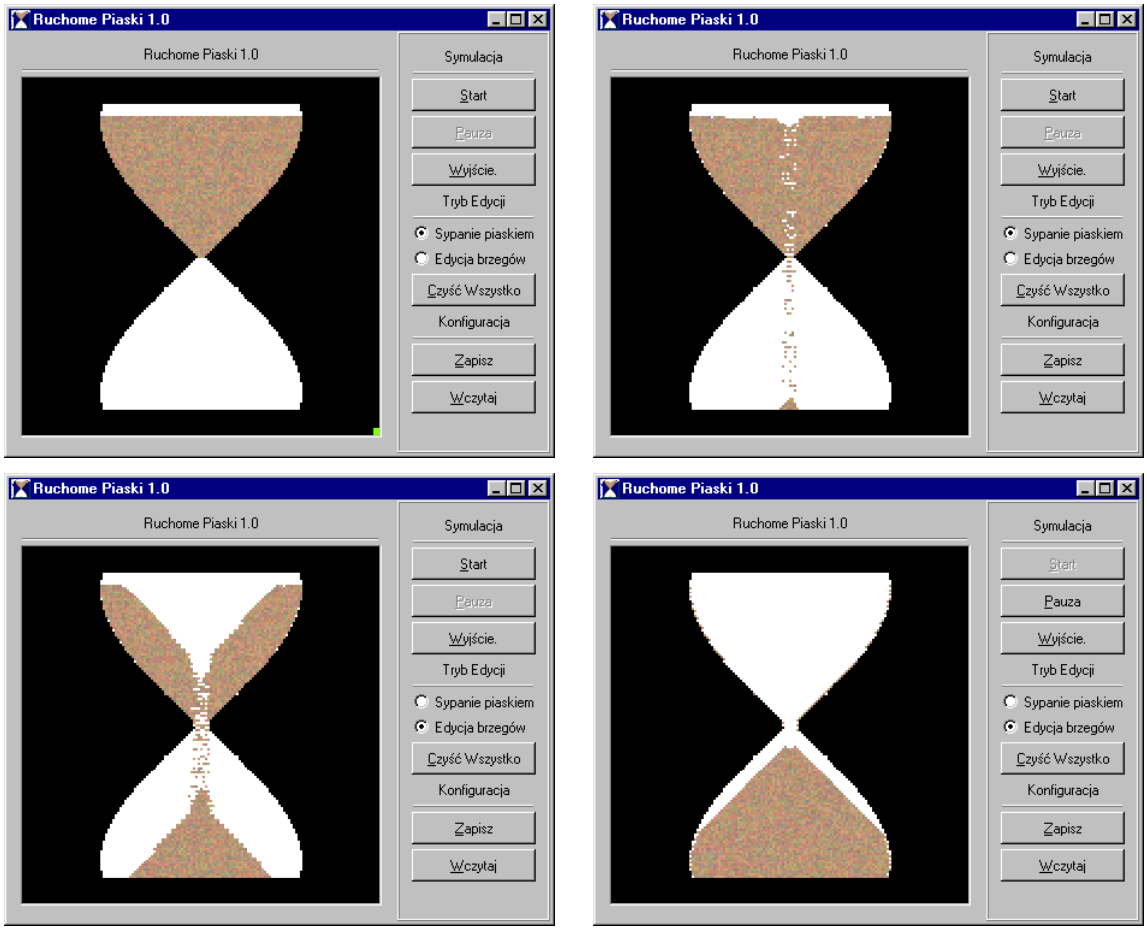
Symulacja usypywania piasku na przeszkodach. W chwili początkowej duża objętość piasku znajduje się ponad przeszkodami. Na rysunkach widać wyraźnie wewnętrzne (lokalne) zatrzymywanie się pojedynczych ziaren piasku na ziarnach podwójnych (połączonych). Efekt ten można wzmocnić lub osłabić poprzez edycję parametru P.

3.2 Dowolna konfiguracji brzegów



Ten przykład zamiszczam po to, by pokazać jak dowolne mogą być brzegi, oraz konfiguracja piasku w chwili początkowej. Tu na tym przykładzie napis "UWr" jest złożony z ziaren piasku, napis "WFiA" stanowi barierę złożoną z komórek brzegowych. Po uruchomieniu symulacji "UWr" się rozsypuje i sypie się na "WFiA". Proszę o zachowanie poczucia humoru przy analizowaniu tego przykładu ;).

3.3 Klepsydra



Symulacja dwuwymiarowej klepsydry. Usypujący się piasek z górnej komory tworzy stożek o kącie rozwarcia 45^o, co jest spowodowane użyciem siatki prostokątnej modelu.

4 Podsumowanie

Chciałbym podziękować prof. Pękalskiemu za [1], które okazało się być wspaniałym opracowaniem powoli wprowadzającym w teorie automatów komórkowych. Program ten powstał raczej na zasadzie próby wykorzystania algorytmów przedstawionych w [1], jednak zdecydowałem się go wystawić do konkursu, bo na ludziach którym go pokazałem robił pozytywne wrażenie. Można by oczywiście zająć się badaniem tak uproszczonego modelu na gruncie fizyki (zależności tworzenia się lawin od gęstości piasku itp.), ale w tym miejscu nie o to chyba chodzi.

Życzę miłej zabawy z programem!

Literatura

[1] B. Chopard and Michel Droz, University of Geneva *'Cellular Automata Modelling of Physical Systems.'*, Cambridge University Press, 1998.