Computer Science. Tom 1. 1999 519.713:519.673:681.3.05

*Rafał Wcisło**

ANIMACJA KOMPUTEROWA OPARTA NA SYMULACJI METODĄ CZĄSTEK

1. Wstęp

W artykule zaprezentowano program służący do animacji układu wzajemnie oddziałujących elastycznych obiektów makroskopowych oraz ośrodka ciekłego. Trójwymiarowa animacja przeprowadzana jest w oparciu o symulację zjawisk fizycznych. Jej głównym zadaniem jest wierne oddanie dynamiki obiektów występujących w animacji. Wizualizacja stanowi istotny element programu animacyjnego, jednak została ona znacznie uproszczona poprzez przekazanie podstawowych jej zadań (zasłanianie i cieniowanie) do bibliotek graficznych. Przedstawiono zastosowane techniki symulacji (metoda cząstek i automatów komórkowych), wykrywania i obsługi kolizji obiektów. W celu uzyskania szybszych animacji program został zaimplementowany także w wersji rozproszonej na sieci UNIX-owych stacji roboczych.

2. Animacja komputerowa

2.1. Powstawanie filmu animowanego

Tworzenie filmu animowanego jest procesem wieloetapowym, czasochłonnym i często wymagającym zaangażowania znacznych ośrodków (ludzkich i finansowych). Z tego powodu możliwości zautomatyzowania przynajmniej części procesu produkcyjnego (poprzez użycie komputerów) jest wielce kusząca.

Proces tworzenia filmu animowanego można zawrzeć w następujących etapach (patrz rys. 1):

Etap A – Przed przystąpieniem do realizacji filmu następuje przygotowanie jego scenariusza. Scenariusz nie opisuje z reguły szczegółowo filmu, lecz stanowi jedynie opis jego fabuły. Tworzy się zatem inny dokument, scenopis, zawierający znacznie większą ilości szczegółów. Bardzo często jest on rozbudowywany i poprawiany na bieżąco w trakcie realizacji filmu.

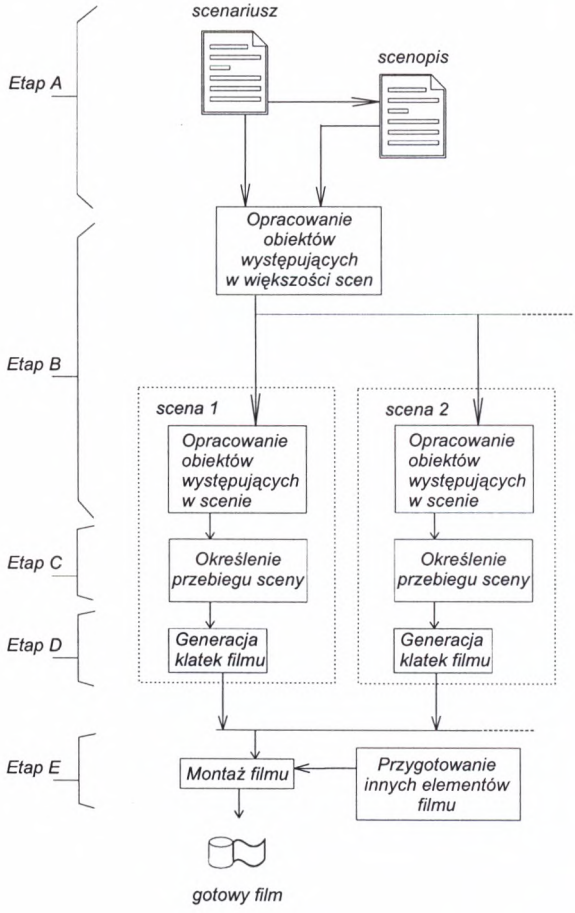
* Katedra Informatyki, Akademia Górniczo-Hutnicza, Kraków

Etap B – Szczegółowe opisanie obiektów (postaci, przedmiotów, wnętrz, krajobrazów) występujących w filmie. Ponieważ część obiektów znajduje się tylko w niektórych scenach, inne zaś w większej ich ilości, stąd rozbicie tego etapu na „globalny" i przypisany każdej scenie.

Etap C – Przed przystąpieniem do generowania pojedynczej sceny, na podstawie scenopisu konkretyzuje się jej przebieg: położenie i ruchy obiektów i kamery, czas trwania sceny itp.

Etap D – Kiedy znany jest przebieg sceny następuje żmudny etap generowania (rysowania) poszczególnych jej klatek. Jest to często proces wieloetapowy: szkicowanie, rysowanie konturów, wypełnianie kolorami, itp.

Etap E – Po zrealizowaniu wszystkich scen oraz zgromadzeniu dodatkowych materiałów przystępuje się do montażu w celu uzyskania końcowego produktu.



Rys. 1. Ogólny schemat produkcji filmu animowanego

Oczywiście przedstawiony schemat działania jest dość powierzchowny i z pewnością nie obejmuje wszystkich aspektów tworzenia filmów animowanych.

Warto zwrócić uwagę na fakt, że łączenie scen (montaż filmu) nie jest procesem znajdującym odzwierciedlenie w rzeczywistości – zmiana sceny polega bardzo często na natychmiastowej zmianie miejsca oraz czasu akcji. Ponieważ w świecie rzeczywistym takie zjawisko nie występuje, nie da się go także odtworzyć w programie metodą symulacji. Z drugiej strony poszczególne sceny filmu są „zwarte" (zachowują ciągłość czasu i miejsca), można zatem je wygenerować metodą symulacji zjawisk fizycznych. Niniejszy artykuł omawia metody cząstek i automatów komórkowych jako możliwe do zastosowania w komputerowej symulacji animowanej sceny.

2.2. Komputerowa generacja animowanej sceny

Komputerowa generacja animowanej sceny polega na maksymalnym zautomatyzowaniu etapów B – D (rys. 1) produkcji filmu animowanego. W zależności od celów i założeń animacji stosuje się rozmaite metody generacji scen. Można je podzielić na dwie klasy:

1) **metody niesymulacyjne** – uzyskanie ruchu obiektów wymaga znacznego udziału człowieka-animatora, który arbitralnie określa zasady rządzące tworzonym światem. Do metod niesymulacyjnych można zaliczyć: metodę ramek kluczowych, *morphing*, *wrapping*, metodę podpatrywania ruchu itp.
2) **metody symulacyjne** – ruch obiektów każdej sceny odbywa się z zachowaniem zadanych ogólnych praw (np. fizycznych praw dynamiki i kinematyki) i jest z nich wyprowadzony na zasadzie symulacji [1, 2, 3, 4, 5].

Praca programu symulacyjnego służącego do automatycznej generacji animowanej może zostać rozbita na trzy główne etapy:

1) **etap modelowania** – przed przystąpieniem do symulacji tworzone są modele występujących w scenie obiektów, inicjowane są także wszystkie zmienne pomocnicze programu,
2) **etap symulacji** – obliczane są zmiany zachodzące w trakcie trwania sceny, np. przemieszczenia obiektów i ich deformacja; symulacja może obejmować nie tylko obiekty występujące w scenie, ale także ruchy „kamery" obserwującej scenę,
3) **etap wizualizacji** – dane wygenerowane w trakcie symulacji pozwalają dokonać wizualizacji poszczególnych klatek sceny.

Jeżeli symulacja zachodzi z odpowiednią szybkością oraz algorytm wizualizacji jest wystarczająco efektywny możliwe jest wykonywanie jednocześnie symulacji i wizualizacji. Mamy wówczas do czynienia z animacją czasu rzeczywistego. Takie rozwiązanie ma dwie dość istotne zalety:

1) nie ma konieczności przechowywania rezultatów symulacji, są one bowiem na bieżąco wykorzystywane przez moduł wizualizacyjny,
2) ponieważ symulacja odbywa się równolegle z wizualizacją, możliwe jest oddanie w ręce użytkownika programu kontroli np. nad ruchami kamery.

2.3. Animowana scena

Stworzenie programu do animacji dowolnej sceny (składającej się z dowolnych obiektów) jest zadaniem bardzo skomplikowanym. Z reguły każdy program animacyjny posiada ściśle

zdefiniowaną klasę scen, które może animować. Program animacyjny omówiony w dalszej części artykułu służy do animacji scen złożonych z:

- określonej ilości ciał stałych cechujących się: masą, elastycznością, położeniem początkowym (rozumianym zarówno jako położenie środka masy, jak i orientacja kątowa w przestrzeni) i prędkością początkową,
- ośrodka ciekłego o zadanej objętości i gęstości,
- jednorodnego, stałego pola grawitacyjnego.

Definicja sceny, która ma podlegać animowaniu zapisywana jest w tekstowym pliku konfiguracyjnym, w języku specjalnie stworzonym dla potrzeb programu.

3. Metoda cząstek – animacja elastycznych ciał stałych

Program animacyjny dokonuje symulacji obiektów stałych metodą cząstek. Metoda ta, przy swoich stosunkowo prostych założeniach, pozwala na symulację ruchu postępowego i obrotowego obiektów, ich deformacji oraz oddziaływań pomiędzy obiektami [6].

3.1. Założenia metody cząstek

Metoda cząstek zastosowana w programie animacyjnym została opracowana na drodze modyfikacji metody dynamiki molekularnej (MD) stosowanej do zjawisk zachodzących w płynach (np. zjawiska turbulencji, konwekcji, mieszania). Metoda ta dokonuje symulacji w mikroskali (na poziomie cząstek lub atomów) korzystając z wiedzy fizyków na temat właściwości oddziaływań pomiędzy drobinami cieczy lub gazu.

3.1.1. Ogólne założenia dynamiki molekularnej

Dokładny opis technik MD można znaleźć np. w książce [7], w niniejszym artykule przedstawione zostaną jedynie główne założenia MD:

- MD służy do symulacji układów cząstek (atomów, cząsteczek chemicznych, cząstek elementarnych, itp.). Cząstki rozpatrywane są jako obiekty punktowe o określonej masie, położeniu i pędzie. Dla cząsteczek chemicznych i innych obiektów bardziej złożonych trzeba również uwzględnić takie własności jak orientacja w przestrzeni, moment bezwładności, stopnie swobody itp.
- przyjmuje się, iż cząstki oddziaływują między sobą zgodnie z zadanym potencjałem oddziaływania wynikającym z założeń fizycznych symulowanego zjawiska;
- proces symulacji odbywa się zazwyczaj według algorytmu pokazanego na rys. 2:
  - symulacja wykonywana jest metodą obliczania stanu układu w kolejnych krokach czasowych oddalonych od siebie o ΔT (krok 6. algorytmu z rys. 2),
  - obliczanie oddziaływań międzycząsteczkowych (krok 3) można na ogół przeprowadzać w taki sposób, aby nie każda para cząstek musiała być rozpatrywana – stosuje się tu listy sąsiadów, geometryczny podział przestrzeni na komórki (np. metoda komórek Hockney'a) itp.,

- krok 5 algorytmu polega na rozwiązaniu równań Newtona:

$$\frac{d\mathbf{x}}{dt} = \mathbf{v} \tag{3.1}$$

$$\boldsymbol{F} = m\frac{\mathrm{d}\,\mathbf{v}}{\mathrm{d}\,t} \tag{3.2}$$

dla każdej cząsteczki,

- okresowo interesujące eksperymentatora dane globalne (całkowita energia, rozkład temperatury, ciśnienia itp.) są obliczane co określoną liczbę kroków symulacji (krok 7).

Rys. 2. Algorytm symulacji metodą MD

3.1.2. Modyfikacja metod MD

Dynamiki molekularnej nie można zastosować wprost, bez modyfikacji do celów animacji scen zawierających obiekty makroskopowe [8, 9, 10]. Wymaga ona pewnych zmian, powodujących powstanie następujących różnic:

- W technice MD pojęciu „cząstki" odpowiada faktyczna drobina materii (atom, cząsteczka chemiczna, cząstka elementarna itp.). W metodzie zastosowanej do animacji mamy do czynienia ze sztucznym podziałem obiektu na domeny (o arbitralnych kształtach i rozmiarach), których dynamika jest następnie symulowana za pomocą ich konwersji na cząstki.
- W MD istotną rolę odgrywa zasięg oddziaływań pomiędzy cząsteczkami. Dla oddziaływań krótkozasięgowych (np. Lennarda – Jonesa) definiuje się tzw. promień odcięcia, który wyznacza zbiór cząstek oddziaływujących z daną cząstką. Zawiera on zwykle od kilku do kilkudziesięciu cząstek i jest zmienny w czasie symulacji. Oddziaływania o większym zasięgu (np. grawitacyjne i elektromagnetyczne) sprawiają tu

znacznie większy problem. W przypadku animacji ciała stałego dla każdej cząstki zbiór sąsiadów jest ustalony przed symulacją i nie zmienia się w czasie jej trwania (zbiór sąsiadów obejmować będzie z reguły tylko najbliższe otoczenie danej cząstki).

– rodzaj oddziaływania w MD zależał od rodzaju symulowanego zjawiska (zakres temperatur, rodzaj cząstek itp.) i był bezpośrednio zaczerpnięty z wiedzy, jaką fizyka dostarcza o naturze oddziaływań między cząstkami. W programie animacyjnym cząstki są jedynie odbiciem sztucznego podziału obiektu makroskopowego i odpowiedni potencjał oddziaływania między nimi należy dobrać tak, aby zachowane zostały własności obiektu (jego elastyczność).

3.2. Potencjał i równania ruchu

W programie animacyjnym rozwiązanie równania ruchu (3.1) i (3.2) odbywa się z zastosowaniem schematu *leap – frog*, czyli:

$$\mathbf{v}_i^{n+\frac{1}{2}} = \mathbf{v}_i^{n-\frac{1}{2}} + \frac{\Delta t}{m_1}\mathbf{F}_i^n$$

$$\mathbf{r}_i^{n+1} = \mathbf{r}_i^n + \Delta t \mathbf{v}_i^{n+\frac{1}{2}} \tag{3.3}$$

gdzie:

$\mathbf{v}_i^n$ – prędkość cząstki i w chwili n,
$\mathbf{r}_i^n$ – położenie cząstki i w chwili n,
m_i – masa cząstki i,
Δt – krok czasowy,
$\mathbf{F}_i^n$ – wypadkowa siła działająca na cząstkę i w chwili n.

W rozpatrywanych przypadkach siłę działającą na cząstkę można rozłożyć na trzy składniki

$$\mathbf{F}_i^n = \mathbf{F}_{p;i}^n + \mathbf{F}_{g,i}^n + \mathbf{F}_{t;i}^n \tag{3.4}$$

gdzie:

$\mathbf{F}_{p;i}^n$ – wypadkowa siła oddziaływań cząstki i z cząstkami sąsiednimi w chwili n,
$\mathbf{F}_{g,i}^n$ – siła grawitacji działająca na cząstkę i w chwili n,
$\mathbf{F}_{t;i}^n$ – siła tarcia (oporu powietrza lub cieczy) działająca na cząstkę i w chwili n.

Siły te obliczane są według następujących wzorów:

– Siła oddziaływania międzycząsteczkowego

$$\mathbf{F}_{p;i}^n = \sum_{j=1}^{N} \mathbf{F}_{p;i,j}^n \tag{3.5}$$

gdzie:

N – liczba cząstek, z którymi oddziałuje cząstka i,
$\mathbf{F}_{p;i,j}^n$ – siła oddziaływania cząstki i z sąsiadującą cząstką j.

Obliczanie siły oddziaływania pomiędzy cząstkami i i j odbywa się poprzez zdefiniowanie potencjału tego oddziaływania

$$V_{i,j}(r_{i,j}) = \frac{1}{2} k_{i,j} (r_{i,j} - r_{0;i,j})^2 \tag{3.6}$$

gdzie:

$k_{i,j}$ – współczynnik określający „twardość'' oddziaływania,
$r_{i,j}$ – odległość pomiędzy cząstkami i i j ($|r_{i,j}| = |r_i - r_j|$),
$r_{0;i,j}$ – stała określająca odległość, w której zanika oddziaływanie pomiędzy cząstkami i i j.

Siła oddziaływania obliczana jest jako

$$\mathbf{F}_{0;i,j}(r_{i,j}) = V'_{i,j}(r_{i,j}) \frac{\mathbf{r}_{i,j}}{r_{i,j}} \tag{3.7}$$

– Siła grawitacji

$$\mathbf{F}^n_{g,i} = m_i \boldsymbol{g} \tag{3.8}$$

gdzie $\boldsymbol{g}$ – wektor przyspieszenia pola grawitacyjnego.

– Siła tarcia

$$\mathbf{F}^n_{t,i} = -\lambda \mathbf{v}^n_i \tag{3.9}$$

gdzie λ – współczynnik tarcia.

Ponieważ w schemacie *leap-frog* prędkości wyliczane są w chwilach „połówkowych", należy przyjąć przybliżenie

$$\mathbf{v}^n_i = \frac{\mathbf{v}_i^{n-\frac{1}{2}} + \mathbf{v}_i^{n+\frac{1}{2}}}{2} \tag{3.10}$$

zatem równanie (3.9) przyjmuje postać

$$\mathrm{F}^n_{t,i} = -\lambda \frac{\mathrm{v}_i^{n-\frac{1}{2}} + \mathrm{v}_i^{n+\frac{1}{2}}}{2} \tag{3.11}$$

Po podstawieniu wzorów (3.5) i (3.11) do (3.3) schemat *leap-frog* wygląda następująco:

$$\mathbf{v}_i^{n+\frac{1}{2}} = \frac{1-\phi_i}{1+\phi_i} \mathbf{v}_i^{n-\frac{1}{2}} + \frac{1}{1+\phi_i} \frac{\Delta t}{m_1} (\mathbf{F}^n_{g,i} + \mathbf{F}^n_{p;i})$$

$$\mathbf{r}_i^{n+1} = \mathbf{r}_i^n + \Delta t \, \mathbf{v}_i^{n+\frac{1}{2}} \tag{3.12}$$

gdzie

$$\phi_i = \lambda \frac{\Delta t}{2 m_i} \tag{3.13}$$

3.3. Animacja układu obiektów

Do symulacji pojedynczego obiektu potrzebne są dwa zbiory:

1) zbiór cząstek wchodzących w skład obiektu,
2) zbiór par indeksów cząstek definiujący relację sąsiedztwa pomiędzy cząstkami.

Oba zbiory posiadają tę własność, iż nie zmieniają się w trakcie symulacji (sytuacja ta może się zmienić, jeśli dopuścimy np. możliwość rozrywania obiektu). Algorytm symulacyjny zwolniony jest zatem z konieczności wykrywania oddziaływań pomiędzy cząsteczkami.

Jeżeli w animowanej scenie występuje więcej niż jeden obiekt symulacja musi uwzględniać także możliwość oddziaływania pomiędzy nimi, a więc pomiędzy cząstkami nie zdefiniowanymi a priori jako sąsiedzi. Realizacja oddziaływania pomiędzy obiektami opiera się zatem o dwa inne algorytmy: wykrywania kolizji oraz odbicia obiektów.

3.3.1. Wykrywanie kolizji

Problem wykrywania kolizji jest jednym z podstawowych zagadnień w programach symulujących układy obiektów [11, 12], w których oddziaływania pomiędzy obiektami mają charakter incydentalny. W prezentowanym programie zastosowano algorytm oparty na porównywaniu ograniczeń obiektów. Mimo że jest to metoda mało dokładna (powoduje ona wiele „fałszywych alarmów" – obiekty nie zderzają się, lecz przechodzą blisko siebie), za jej zastosowaniem przemawia prostota oraz szybkość.

Ograniczenie obiektu A definiowane jest jako:

$$S(A)=[\min_x(A),\max_x(A)]\times[\min_Y(A),\max_Y(A)]\times[\min_Z(A),\max_Z(A)] \quad (3.14)$$

gdzie:

$\min_\alpha(A)$ – minimum ze zbioru współrzędnych o kierunku $\alpha\in\{X, Y, Z\}$ cząstek wchodzących w skład modelu obiektu A.

Algorytm wykrywa kolizję dwóch obiektów A i B, jeśli spełniony jest warunek

$$S(A)\cap S(B)\neq\varnothing \quad (3.15)$$

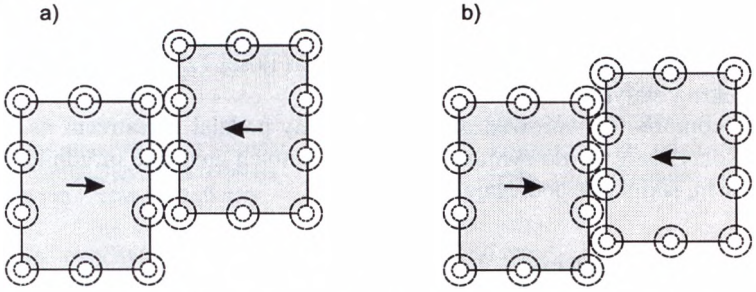
Ustalenie listy obiektów, które mogą ze sobą oddziaływać odbywa się poprzez porównanie ich ograniczeń, nie wymaga porównywania wszystkich cząstek wchodzących w ich skład. Złożoność obliczeniowa algorytmu jest zatem wprost proporcjonalna do kwadratu ilości obiektów, nie zaś od kwadratu liczby cząstek.

3.3.2. Realizacja odbić obiektów

W przypadku gdy algorytm wykrywający kolizje znajdzie obiekty, które mogą ze sobą oddziaływać, konieczne jest przeprowadzenie symulacji odbicia obiektów (jeśli faktycznie ono nastąpi). Obliczenie oddziaływania międzyobiektowego zostało zrealizowane również na zasadzie oddziaływania pomiędzy cząstkami. Zachodzi jednak pytanie, jakie cząstki oddziałujących obiektów A i B powinny być brane pod uwagę. Nasuwające się rozwiązanie,

polegające na rozpatrywaniu oddziaływań pomiędzy zewnętrznymi (brzegowymi) cząstkami obiektów stwarza trudność w określeniu zasięgu takiego oddziaływania:

- powinien on być krótki, gdyż obiekty oddziaływują ze sobą tylko podczas ich fizycznego zetknięcia,
- z drugiej strony krótkozasięgowość oddziaływania powodowałaby uzależnienie wyniku odbicia obiektów od wzajemnego przesunięcia układu ich cząstek brzegowych (patrz rys. 3).

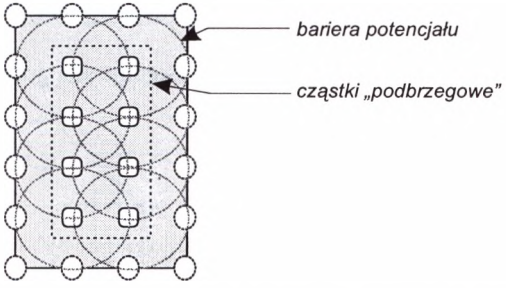


Rys. 3. Różne ułożenie cząstek brzegowych dwóch obiektów w czasie oddziaływania: a) cząstki brzegowe „wpadają" na siebie, b) cząstki brzegowe „mijają się"

Przyjęte rozwiązanie polega na wyodrębnieniu w obiekcie tzw. cząstek „podbrzegowych". Są one zdefiniowane jako te cząstki, które spełniają dwa warunki (patrz rys. 4):

1) nie są cząstkami brzegowymi,
2) posiadają przynajmniej jedną sąsiednią cząstkę brzegową.

Oddziaływanie między obiektami *A* oraz *B* (prowadzące do odbicia tych obiektów) zostało zrealizowane w oparciu o oddziaływanie pomiędzy cząstkami brzegowymi obiektu *A* i cząstkami podbrzegowymi obiektu *B* oraz viceversa. Mimo że bariera potencjału wytwarzana przez cząstki podbrzegowe obiektów (patrz rys. 4) oddaje jedynie w przybliżeniu kształt zewnętrzny obiektu, to jednak uzyskane efekty odbić są realistyczne i zapobiegają zjawisku wzajemnej penetracji obiektów.



Rys. 4. Bariera potencjału obiektu, pozwalająca na realizację odbić pomiędzy obiektami

4. Animacja cieczy

W celu animacji efektów związanych z zachowaniem się obiektów makroskopowych w ośrodku ciekłym należało znaleźć aparat obliczeniowy pozwalający na szybką symulację tych zjawisk. Ponieważ głównym celem pracy było wierne oddanie ruchu obiektów makroskopowych, a nie ośrodka ciekłego, zastosowano do obliczeń cieczy schemat trójwymiarowego automatu komórkowego [13]. Pozwala on, po zintegrowaniu z modelem MMD, na szybkie ob-

liczanie zmian stanu ośrodka ciekłego oraz na symulację dwóch zjawisk związanych z zanurzeniem (całkowitym lub częściowym) ciała stałego w cieczy:

- siły wyporu,
- lepkości cieczy.

Zastosowanie w jednym programie animacyjnym dwóch schematów symulacji (metody cząstek i automatu komórkowego) wynika z dwóch faktów:

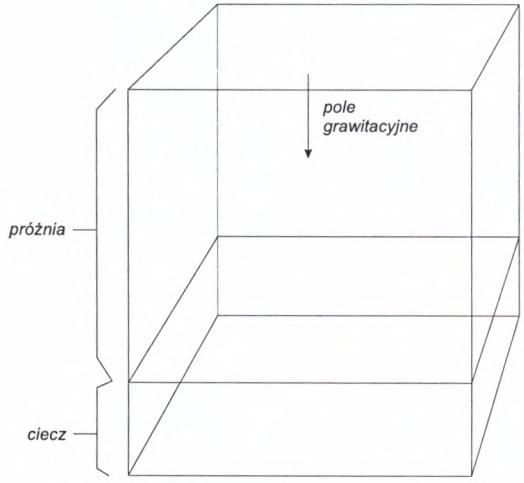
1) nie jest możliwe zastosowanie metody cząstek do symulacji cieczy bez znaczącego spowolnienia szybkości – ilość cząstek potrzebna do stworzenia modelu cieczy dla większości scen byłaby wielokrotnie większa od ilości cząstek niezbędnych w modelach obiektów stałych,
2) automat komórkowy wprowadza z góry ustalony podział przestrzeni na komórki, zatem wykorzystanie go do symulacji obiektów stałych znacznie utrudniłoby symulację odkształceń, jakim one podlegają.

4.1. Automat komórkowy

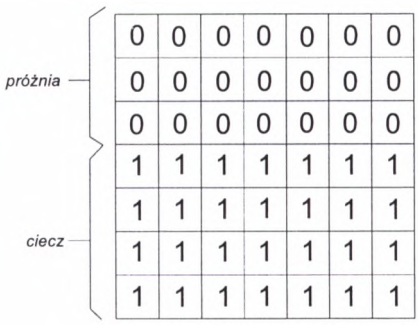
Automat komórkowy został skonstruowany tak, aby zachowywać całkowitą objętość cieczy oraz dostarczać informacji na temat rozkładu jej ciśnienia. Pominięte zostały natomiast zjawiska związane z dynamiką cieczy (komórki – elementy cieczy nie zachowują pędu), nie można zatem przedstawionym automatem uzyskać takich zjawisk jak: falowanie powierzchni, przepływ cieczy itp. Uwzględnienie tych zjawisk jest obecnie przedmiotem intensywnych badań.

Animacje zawierające ośrodek ciekły muszą spełniać następujące założenia:

- ciecz znajduje się w basenie znajdującym się w dolnej części animowanej sceny (patrz rys. 5),
- w animowanej scenie występuje siła grawitacji skierowana pionowo w dół.



Rys. 5. Trójwymiarowy automat komórkowy symulujący ośrodek ciekły



Rys. 6. Przekrój pionowy przez automat komórkowy symulujący ciecz w pudle obliczeniowym – stan początkowy

Program animacyjny dzieli „pudło obliczeniowe" na sześcienne komórki o rozmiarze równym średniej odległości pomiędzy cząstkami obiektów stałych. Z każdą komórką skojarzona jest liczba całkowita (patrz rys. 6), która może przyjmować następujące wartości:

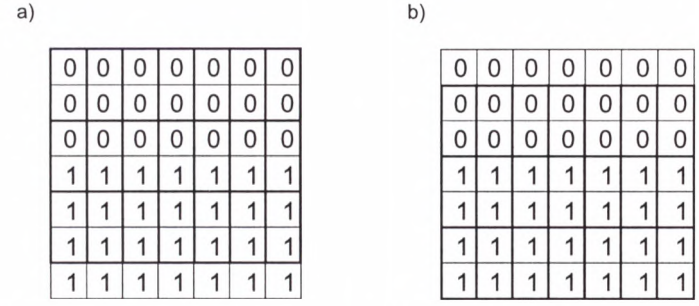
0 – oznacza brak cieczy w danej komórce,
1 – oznacza obecność cieczy w danej komórce w stanie „stabilnym",
2, 3, 4... – oznacza obecność cieczy w danej komórce pod zwiększonym (dwu-, trzy-, czterokrotnie, ...) ciśnieniem.

Automat komórkowy symulujący ciecz oparty jest o dwie reguły:

1) pionową – pozwalającą automatowi komórkowemu reagować na pole grawitacyjne oraz zachować całkowitą objętość cieczy,
2) poziomą – odpowiedzialną za zachowanie przez ciecz reguły naczyń połączonych.

4.1.1. Reguła pionowa

Reguła pionowa oparta jest o bardzo proste sąsiedztwo „pod – nad", polegające na podziale wszystkich komórek automatu na pary komórek, z których jedna leży na drugiej (patrz rys. 7). Reguła taka musi być dwukrokowa [14], aby objąć wszystkie pary komórek. Tak więc w kroku pierwszym (rys. 7a) sąsiedztwa przesunięte są o jedną komórkę w stosunku do kroku drugiego (rys. 7b).



Rys. 7. Dwa kroki reguły pionowej automatu komórkowego (przekrój pionowy przez automat)

W celu opisu reguł pozwalających na wyznaczenie następnego stanu automatu w obrębie sąsiadujących komórek, oznaczmy poprzez L wartość komórki „dolnej", a poprzez U wartość komórk i „górnej" (warto zwrócić uwagę, że komórki górne z kroku pierwszego stają się dolnymi w kroku drugim i odwrotnie). Poprzez L' i U' oznaczmy wartości wynikowe odpowiednio dla komórki dolnej i górnej.

Reguła pionowa wyczulona jest na trzy przypadki:

$$\begin{aligned} U>0 \text{ i } L=0 &\Rightarrow U'=U-1 \text{ i } L'=1 \\ U<0 \text{ i } L>1 &\Rightarrow U'=U+1 \text{ i } L'=L-1 \\ \text{w pozostałych przypadkach} &\Rightarrow L'=L \text{ i } U'=U \end{aligned} \tag{4.16}$$

4.1.2. Reguła pozioma

Reguła pozioma automatu oparta jest o sąsiedztwo czterech komórek przylegających do siebie, leżących na tej samej „głębokości". Reguła pozioma również jest dwukrokowa (patrz rys. 8).

a)

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

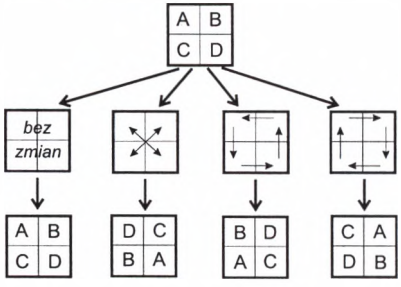
b)

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

Rys. 8. Podział na sąsiedztwa dla dwóch kroków reguły poziomej (przekrój poziomy przez automat)

W każdym kroku reguły następuje losowe przetasowanie z jednakowym prawdopodobieństwem komórek zawartych w każdym sąsiedztwie według jednego z czterech schematów:

1. komórki nie są zamieniane (rys. 9a),
2. komórki zamieniane są po przekątnej (rys. 9b),
3. komórki rotowane są w lewo (rys. 9c),
4. komórki rotowane są w prawo (rys. 9d).



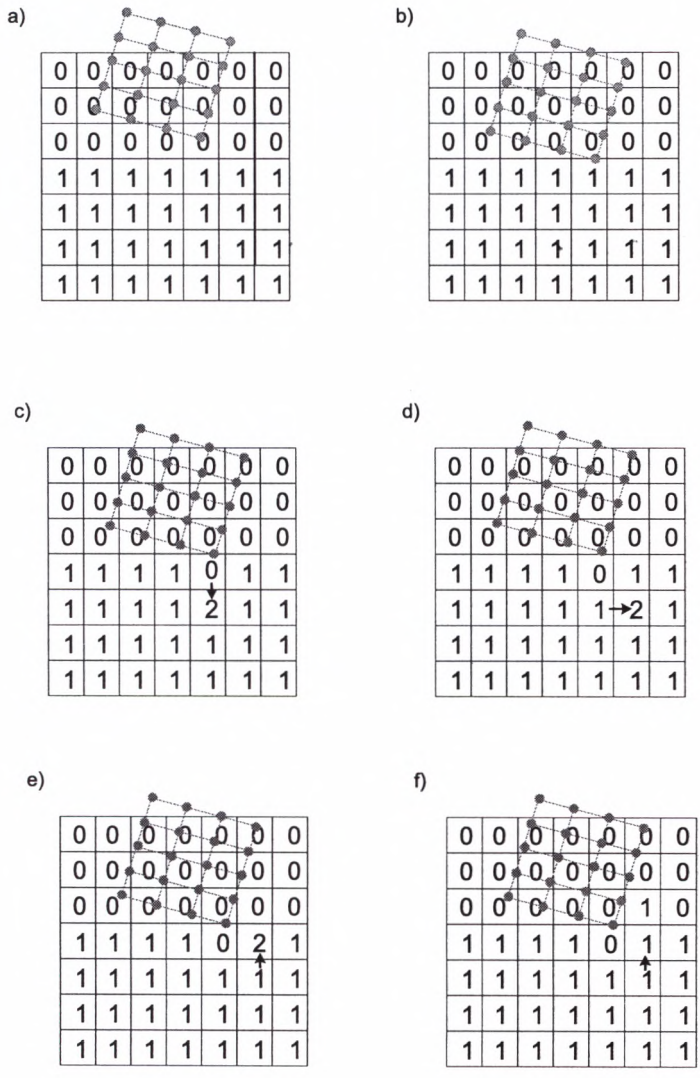
Rys. 9. Metody przetasowania komórek w obrębie sąsiedztwa poziomego

4.1.3. Oddziaływania pomiędzy obiektami stałymi i cieczą

Realizacja oddziaływań pomiędzy obiektami stałymi i cieczą wymaga, aby:

- wykrywana była obecność ciała stałego zanurzonego w cieczy,
- wykrywana była obecność cieczy w otoczeniu cząstek tworzących obiekty stałe.

W przypadku wykrycia obecności cząstki w komórce zawierającej ciecz (której zawartość jest różna od zera), komórka jest zerowana, zaś jej dotychczasowa zawartość przejmowana jest przez losowo wybraną sąsiadującą z nią komórkę (patrz rys. 10). Takie działanie programu pozwala unikać sytuacji, gdy ta sama przestrzeń zajmowana jest zarówno przez ciecz, jak i przez ciało stałe. Zaburzenie wywołane usunięciem cieczy z komórki jest propagowane przez reguły automatu ku powierzchni cieczy powodując w konsekwencji jej zaburzenie.



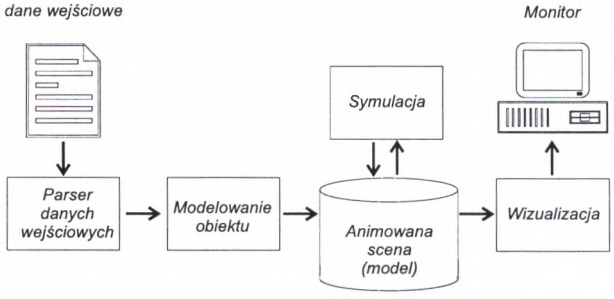
Rys. 10. Kilka kroków prezentujących zanurzanie się obiektu stałego w cieczy: a) obiekt zbliża się do powierzchni cieczy, b) wejście cząstek w komórki automatu zajmowane przez ciecz, c) przejęcie cieczy wypartej z zajętej komórki przez komórkę sąsiednią, d) zadziałanie reguły poziomej, e, f) zadziałanie reguły pionowej

Jednocześnie dla każdej cząstki obiektu stałego sprawdzane są otaczające komórki automatu komórkowego. W przypadku wykrycia w nich cieczy modyfikowane są równania definiujące siły działające na cząstkę:

- modyfikowany jest wektor pola grawitacyjnego (na podstawie różnicy gęstości cieczy i obiektu stałego) w celu uwzględnienia siły wyporu – możliwe jest nawet odwrócenie jego kierunku, co prowadzi do uzyskania zjawiska unoszenia się obiektu na powierzchni cieczy (por. (3.8)),
- modyfikowany jest współczynnik tarcia λ (por. (3.11)), aby uwzględnić tarcie pomiędzy cieczą a obiektem stałym.

5. Realizacja sekwencyjna programu

Schemat działania programu animacyjnego w wersji sekwencyjnej przedstawiony jest na rys. 11.



Rys. 11. Schemat programu animacyjnego

Praca programu animacyjnego przebiega w następujący sposób:

- opis sceny jest wczytywany z pliku konfiguracyjnego i następuje utworzenie modeli występujących w scenie obiektów (etap modelowania),
- czasie animacji wykonywana jest symulacja zmian kształtu i położenia obiektów,
- dane wygenerowane przez symulację podlegają (co określoną liczbę kroków czasowych) wizualizacji, tworząc kolejne klatki filmu.

5.1. Modelowanie

Program rozpoczyna pracę od wczytania pliku konfiguracyjnego opisującego animowaną scenę. Plik zawiera następujące informacje:

- opis obiektów stałych – ich kształt, położenie, masa, prędkość początkowa, elastyczność,
- opis ośrodka ciekłego – jego położenie, gęstość, objętość,
- wektor przyspieszenia grawitacyjnego,
- dodatkowe zmienne sterujące.

Tekstowy opis sceny konwertowany jest na wewnętrzną reprezentację programu, sprawdzana jest także poprawność wczytanych parametrów (np. dodatniość wszystkich wymiarów, mas i gęstości).

5.1.1. Modelowanie obiektów stałych

W trakcie modelowania obiektu stałego tworzony jest układ cząstek i ich oddziaływań wykorzystywany następnie w procesie symulacji. Dodatkowo generowane są pomocnicze listy (np. trójkątów definiujących zewnętrzny kształt obiektu) wykorzystywane przez moduł wizualizacyjny do rysowania obiektu.

Przed przystąpieniem do generacji modeli obiektów program analizuje kształty wszystkich obiektów (ich złożoność, dysproporcje w wielkości), znajdując optymalną gęstość siatki cząstek – w konsekwencji jest ona zbliżona we wszystkich obiektach, co umożliwia prawidłową realizację odbić obiektów za pomocą algorytmu opisanego w rozdziale 3.3.2.

Generowanie zbioru cząstek odbywa się z zachowaniem następujących reguł:

- układ cząstek leżących na powierzchni obiektu powinien odwzorowywać kształt obiektu (a zatem cząstki powinny znajdować się na krawędziach i w wierzchołkach),
- rozkład cząstek wewnątrz obiektu powinien być maksymalnie równomierny.

Po wygenerowaniu zbioru cząstek znajdowane są pary cząstek sąsiadujących za sobą. Tworzą one zbiór oddziaływań międzycząsteczkowych danego obiektu.

Końcowym rezultatem modelowania obiektu są następujące zbiory:

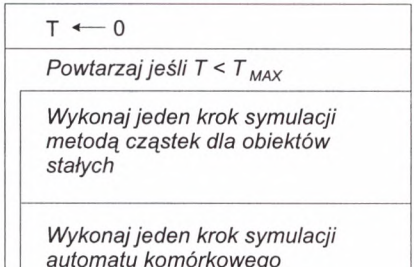
- zbiór cząstek,
- zbiór oddziaływań (par cząstek sąsiadujących),
- zbiór cząstek brzegowych,
- zbiór cząstek podbrzegowych,
- zbiór trójkątów definiujących powierzchnię obiektu.

5.1.2. Modelowanie cieczy

Modelowanie cieczy polega na utworzeniu trójwymiarowej tablicy sześciennych komórek pokrywającej przestrzeń, którą zajmuje (lub może zająć w trakcie animacji) ciecz. Rozmiar każdej komórki równy jest średniej odległości pomiędzy sąsiadującymi cząstkami w obiektach stałych. Stan każdej komórki opisywany jest pojedynczą liczbą całkowitą. Komórki zawierające ciecz inicjowane są liczbą 1, pozostałe zerami (patrz rys. 6).

5.2. Symulacja

Symulacja zmian w animowanej scenie polega na naprzemiennym wykonywaniu obliczeń metodą cząstek (dla ciał stałych) oraz automatu komórkowego (dla cieczy). Po każdym cyklu czas symulacji T zwiększany jest o stały krok czasowy ΔT (patrz rys. 12). Wielkość kroku czasowego ograniczona jest stabilnością numeryczną metody cząstek.



Rys. 12. Symulacja animowanej sceny

5.3. Wizualizacja

Moduł symulacyjny dostarcza danych wystarczających do przeprowadzenia prostej wizualizacji. Dostarczane są współrzędne cząstek brzegowych każdego obiektu oraz lista trójkątów definiujących ich powierzchnie (wierzchołki każdego z tych trójkątów są cząstkami brzegowymi). Powierzchnia cieczy wyznaczana jest siatką uzyskaną poprzez znalezienie najwyżej położonych zajętych (niezerowych) komórek automatu.

Algorytm wizualizacji odpowiedzialny jest za:

- zrzutowanie trójwymiarowej sceny na płaszczyznę dwuwymiarową z zachowaniem zasad perspektywy,
- usunięcie niewidocznych elementów sceny (zasłanianie), obliczenie oświetlenia każdego elementu,
- wygładzenie powierzchni obiektów (usunięcie efektu dyskretyzacji powierzchni trójkątami).

Większość tych zadań może zostać wykonana za pośrednictwem specjalizowanej biblioteki graficznej (np. OpenGL) lub odrębnego pakietu wizualizacyjnego (np. AVS).

6. Realizacja rozproszona programu

W programie animacyjnym zastosowano różne techniki optymalizacyjne, mające na celu maksymalne przyspieszenie obliczeń numerycznych (wprowadzenie dodatkowych zmiennych, denormalizacja wartości fizycznych itp.). Dalszego przyspieszenia działania programu należało szukać w zastosowaniu algorytmów rozproszonych [15, 16].

Program animacyjny został rozbity na szereg procesów różniących się między sobą funkcjonalnie z zastosowaniem techniki dekompozycji algorytmicznej. Program rozproszony składa się z następujących, odrębnych procesów:

- procesów wykonawczych realizujących symulację oddziaływań i ruchów obiektów,
- procesu wizualizacji realizującego algorytmy niezbędne do wizualizacji animacji na monitorze jednej ze stacji roboczych,
- procesu nadzorującego przebieg animacji.

6.1. Założenia rozproszenia obiektów

Działanie programu rozproszonego bazuje na równoległej symulacji ruchów N obiektów przez K procesów wykonawczych. Do realizacji wizualizacji animacji i ewentualnego sterowania symulacją wykorzystywane są odrębne procesy. Przypisanie obiektów do procesów wykonawczych odbywa się z zachowaniem następujących reguł:

- obciążenie procesów wykonawczych powinno być maksymalnie równomierne – należy tu wziąć pod uwagę zarówno liczbę obiektów symulowanych przez poszczególne obiekty, jak i różną złożoność obiektów,
- obiekty, które w danym momencie oddziaływują ze sobą (lub są dostatecznie blisko siebie, aby być „podejrzanymi" o nadchodzące oddziaływanie) muszą być symulowane przez ten sam proces wykonawczy; warunek ten jest silniejszy od poprzedniego, za-

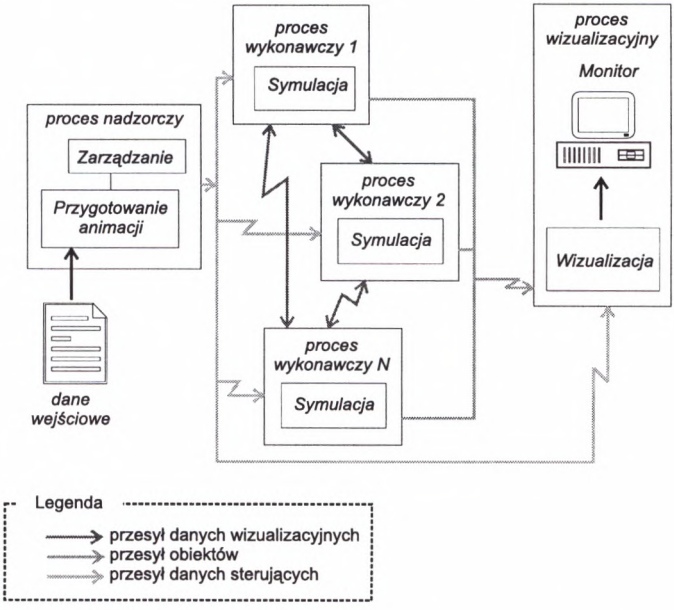
pewniającego równowagę obciążenia, co jest istotne w sytuacjach, gdy warunki 1 i 2 nie mogą być spełnione jednocześnie).
- konieczność synchronizacji procesów wykonawczych powinna zostać ograniczona do minimum, pozwalając im działać maksymalnie niezależnie – z punktu widzenia funkcjonalności programu jedynym momentem, kiedy procesy powinny być synchronizowane jest moment wizualizacji animowanej sceny.

6.2. Sterowanie rozproszoną animacją

Realizacja postulatów 1 i 2 wymaga przesyłania obiektów pomiędzy procesami wykonawczymi. Nadzór nad przesyłaniem (detekcja konieczności przesłania obiektów i wysłanie odpowiednich komunikatów inicjujących transfer) może zostać zrealizowana poprzez oddzielny proces nadzorczy (sterowanie centralne animacją) lub zostać rozproszony pomiędzy wszystkie procesy wykonawcze (sterowanie zdecentralizowane).

6.2.1. Sterowanie centralne

Schemat działania programu rozproszonego działającego w oparciu o sterowanie centralne przedstawia rys. 13.



Rys. 13. Schemat programu ze sterowaniem centralnym

Program oparty jest o wzajemną współpracę trzech typów procesów (nadzorczego, wizualizacyjnego i wykonawczych) mających za zadanie przeprowadzanie animacji zgodnie z założeniami przedstawionymi w rozdz. 6.1.

Proces nadzorczy odpowiedzialny jest za:

- przygotowanie całości animacji, a w szczególności:
  - wczytanie pliku konfiguracyjnego opisującego początkową scenę animacji,
  - przygotowanie modeli obiektów występujących w animacji,
  - wczytanie konfiguracji rozproszenia programu (ilości stacji roboczych, ich przeznaczenia),
  - uruchomienie pozostałych procesów uczestniczących w animacji,
  - przesłanie uruchomionym procesom początkowych danych dotyczących animacji,
- nadzorowanie przebiegu animacji, czyli:
  - wykrywanie kolizji obiektów,
  - wymuszanie przesłań obiektów pomiędzy procesami wykonawczymi jako rezultat wykrytej kolizji lub równoważenia obciążenia.

Działanie procesu wizualizacyjnego sprowadza się do cyklicznego:

- gromadzenia danych wizualizacyjnych przysyłanych przez procesy wykonawcze,
- wizualizacji animowanej sceny w danej chwili czasowej,
- synchronizacji procesów wykonawczych,
- przesyłania do procesu nadzorczego danych dotyczących obciążenia poszczególnych procesów wykonawczych,
- reagowania na ewentualne działania użytkownika-obserwatora animacji, polegające na zmianie parametrów wizualizacji (ustawienia kamery), żądaniu przerwania animacji itp.

Działanie każdego procesu wykonawczego polega na:

- symulacji kolejnych faz ruchu przypisanych do niego obiektów (ciał stałych lub cieczy),
- wysyłaniu informacji o zmianach ograniczenia przypisanych do niego obiektów,
- wysyłaniu i przyjmowaniu obiektów w wyniku wiadomości otrzymywanych od procesu nadzorczego,
- wysyłaniu do procesu wizualizacyjnego danych wizualizacyjnych co określoną liczbę kroków czasowych.

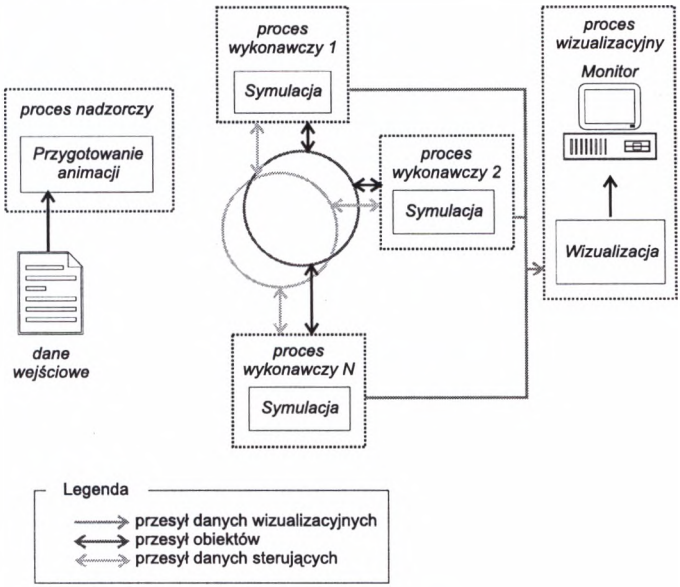
6.2.2. Sterowanie zdecentralizowane

Podobnie jak w realizacji ze sterowaniem centralnym algorytmu, w sterowaniu rozproszonym (rys. 14) występują trzy rodzaje procesów: proces nadzorczy, procesy wykonawcze i proces wizualizacyjny. Ich rola jest jednak zmieniona.

Rozproszenie sterowania animacją (wykrywania kolizji, równoważenia obciążenia) miało na celu uniknięcie sytuacji, w której przeciążenie pojedynczego procesu zaangażowanego w animację (nadzorczego) spowoduje zwolnienie jej przebiegu. Przeniesienie zadań kontrolnych na procesy wykonawcze odbywa się kosztem niewielkiego zwiększenia ich obciążenia, związanego głównie z większą ilością danych odczytywanych przez nie poprzez sieć. Wynika to z faktu, iż każda wiadomość istotna dla decyzji sterujących rozsyłana jest techniką *broadcastu* do wszystkich procesów wykonawczych.

Proces nadzorczy odpowiedzialny jest tylko za początkową fazę animacji:

- przygotowanie całości animacji, a w szczególności:
  - wczytanie pliku konfiguracyjnego opisującego początkową scenę animacji,
  - przygotowanie modeli obiektów występujących w animacji,
  - wczytanie konfiguracji rozproszenia programu (ilości stacji roboczych, ich przeznaczenia),
  - uruchomienie pozostałych procesów uczestniczących w animacji,
  - przesłanie uruchomionym procesom początkowych danych dotyczących animacji,
- czekanie na zakończenie animacji.



Rys. 14. Schemat programu ze sterowaniem rozproszonym

Działanie procesu wizualizacyjnego sprowadza się do cyklicznego:

- gromadzenia danych wizualizacyjnych przysyłanych przez procesy wykonawcze,
- wizualizacji animowanej sceny w danej chwili czasowej,
- synchronizacji procesów wykonawczych,
- przesyłania do procesów wykonawczych danych dotyczących obciążenia poszczególnych procesów,
- reagowania na ewentualne działania użytkownika-obserwatora animacji, polegające na zmianie parametrów wizualizacji (ustawienia kamery), żądaniu przerwania animacji itp.

Działanie każdego procesu wykonawczego polega na:

- symulacji kolejnych faz ruchu przypisanych do niego obiektów,
- wysyłaniu informacji o zmianach położenia przestrzennego ograniczenia przypisanych do niego obiektów, wykrywaniu kolizji pomiędzy obiektami i zarządzaniu odpowiednich przesłań obiektów z uwzględnieniem równoważenia obciążenia,

- wysyłaniu i przyjmowaniu obiektów w wyniku wiadomości otrzymywanych od innych procesów wykonawczych lub w wyniku samodzielnych decyzji,
- wysyłaniu do procesu wizualizacyjnego danych wizualizacyjnych co określoną liczbę kroków czasowych.

6.3. Wykrywanie i realizacja kolizji w programie rozproszonym

Wykrywanie kolizji pomiędzy obiektami stałymi wymaga porównania ich ograniczeń (patrz rozdz. 3.3.1). W sytuacji gdy poszczególne obiekty animowane są przez różne procesy, żaden z procesów nie posiada informacji na temat wszystkich ograniczeń. Sytuacja taka prowadzi do błędnego działania programu, który nie wykrywa prawidłowo kolizji. Aby rozwiązać ten problem procesy wykonawcze wysyłają informacje o zmianie ograniczeń podległych im obiektów do procesu sterującego (w przypadku centralnego sterowania) lub do wszystkich pozostałych procesów wykonawczych (w przypadku sterowania zdecentralizowanego). Porównanie ograniczeń wszystkich obiektów może doprowadzić do znalezienia obiektów, które ze sobą kolidują, jednak nie są symulowane przez ten sam proces wykonawczy. Następuje wówczas wysłanie komunikatów wymuszających transfer obiektów. Obiekty przesyłane są w taki sposób, aby:

- wszystkie kolidujące obiekty znalazły się pod kontrolą tego samego procesu wykonawczego,
- obiekty trafiły do najmniej obciążonego procesu,
- nastąpiła jak najmniejsza liczba przesłań.

Jeżeli jednoczesne spełnienie powyższych postulatów nie jest możliwe, zaspokajane są one zgodnie z priorytetem: 1, 2, 3. W wersji sekwencyjnej programu realizacja odbicia obiektów następowała bezpośrednio po wykryciu kolizji. W wersji rozproszonej pomiędzy tymi zdarzeniami może upłynąć trochę czasu (związanego z koniecznością przesłania kilku komunikatów, zanim kolidujące obiekty znajdą się w obrębie tego samego procesu wykonawczego), stąd konieczność rozszerzenia ograniczenia obiektu o odległość d (zależną m.in. od szybkości przesyłania komunikatów)

$$S(A) = [\min_x(A)-d, \max_x(A)+d] \times [\min_Y(A)-d, \max_Y(A)+d] \times [\min_Z(A)-d, \max_Z(A)+d] \quad (6.17)$$

Rozszerzone ograniczenia pozwalają na odpowiednio wczesne wykrycie kolizji.

Realizacja odbicia obiektów odbywa się zawsze w ramach jednego procesu i przebiega w taki sam sposób, jak w wersji sekwencyjnej programu.

6.4. Równoważenie obciążenia

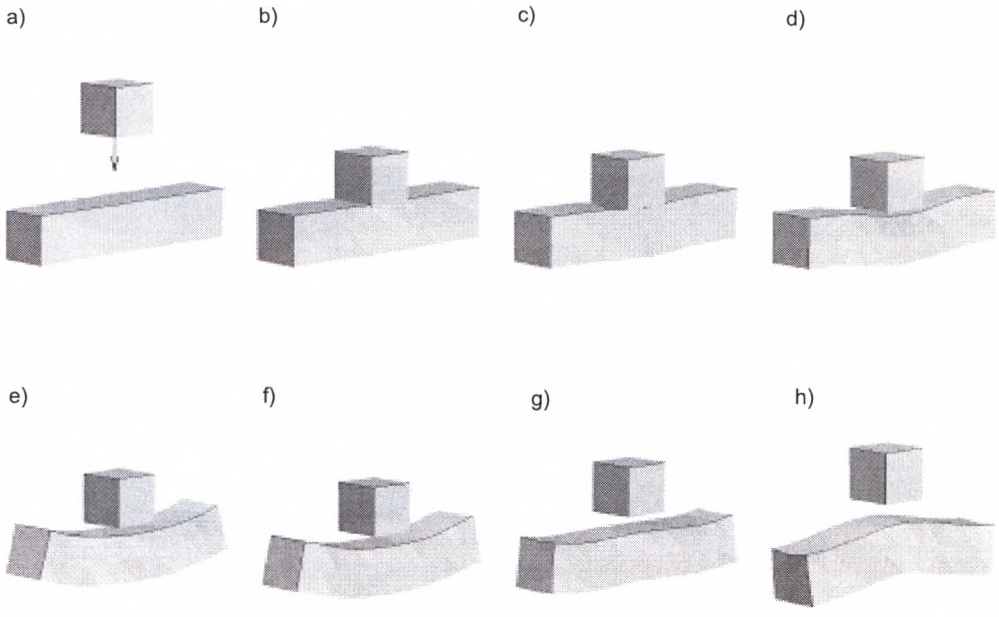
Algorytm równoważenia obciążenia działa w opozycji do algorytmu wykrywania kolizji. Algorytm wykrywania kolizji prowadzi do kumulowania obiektów w procesach (w idealnej dla niego sytuacji wszystkie obiekty są symulowane przez jeden proces). Takie zachowanie jest sprzeczne z postulatem równego obciążenia wszystkich procesów. Algorytm równoważenia obciążenia, mając informacje na temat aktualnego obciążenia procesów i dystrybucji obiektów między nimi, stara się znaleźć takie przesunięcia obiektów pomiędzy procesami, aby nie naruszyć postulatu 2. z rozdz. 6.1.

6.5. Synchronizacja czasu obiektów

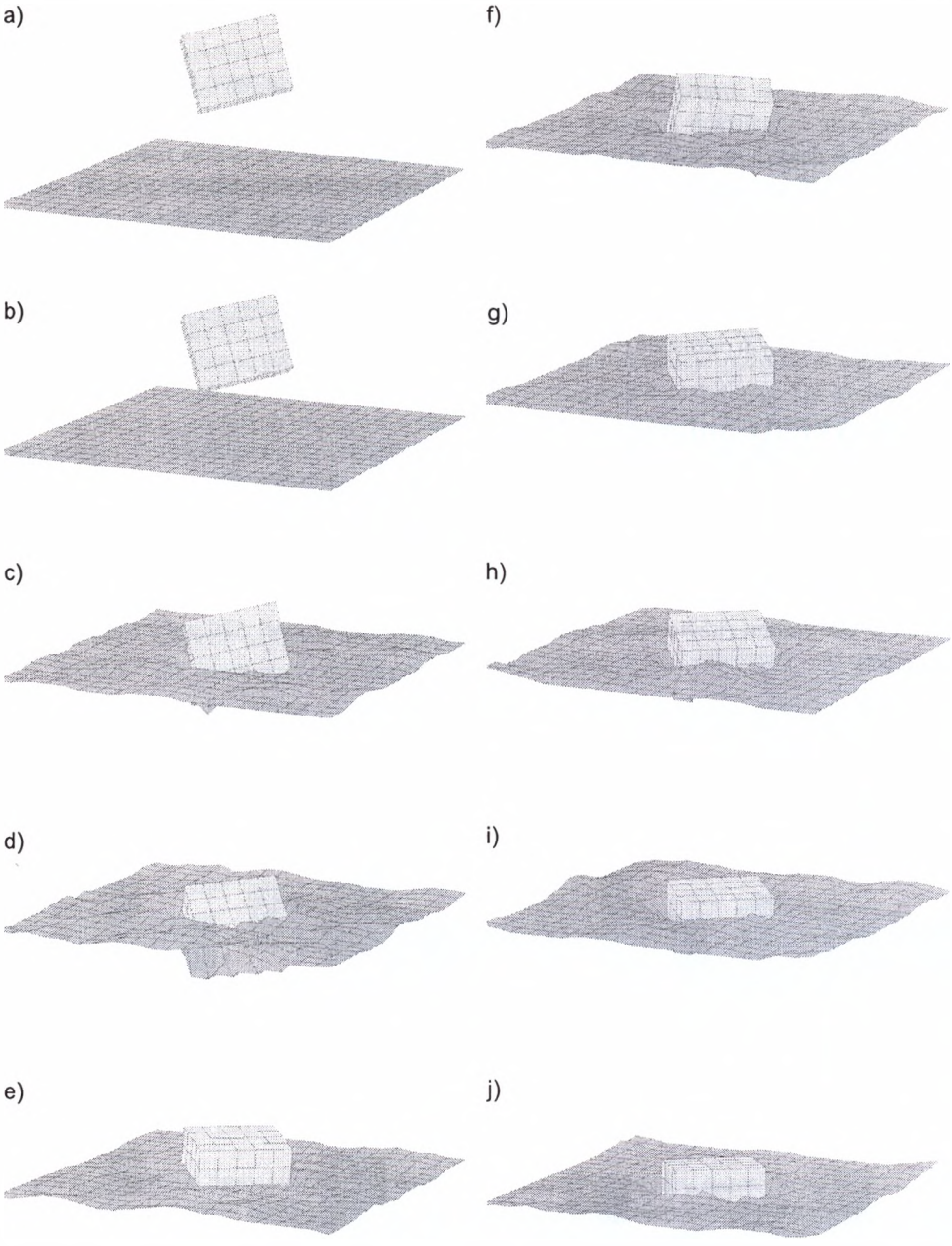
W czasie trwania rozproszonej symulacji synchronizacja procesów odbywa się jedynie w chwilach wizualizacji – proces wizualizacji wstrzymuje procesy do chwili uzyskania wszystkich danych na temat aktualnie generowanej klatki. Generacja taka następuje co kilkaset kroków symulacyjnych, co prowadzi do sytuacji, w której bardzo często czasy symulacji poszczególnych procesów nie są ze sobą zsynchronizowane. Przesłanie obiektu pomiędzy procesami (w przypadku wykrycia kolizji lub zadziałania algorytmu równoważenia obciążenia) prowadzi do obecności w jednym procesie wykonawczym obiektów o różnym czasie symulacji. Aby rozwiązać ten problem proces „zamraża" symulację obiektów o czasie wyprzedzającym pozostałe do chwili jego wyrównania.

7. Przykłady animacji

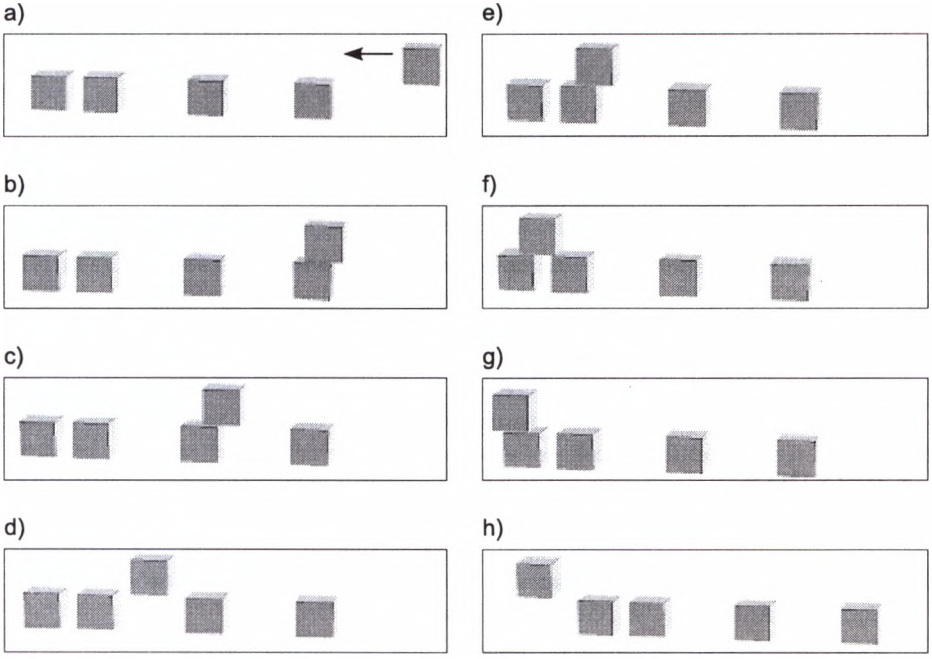
Rysunki 15, 16, 17 przedstawiają przykładowe animacje. Program sekwencyjny i rozproszony generują identyczne animacje przy takich samych danych wejściowych. Jedyną dostrzegalną różnicą jest szybkość animacji obserwowanej na ekranie. Rys. 18 przedstawia wykres zależności szybkości animacji w zależności od ilości stacji roboczych zaangażowanych w programie (użyto stacji roboczych IBM RS/6000 model 320 połączonych siecią Ethernet).



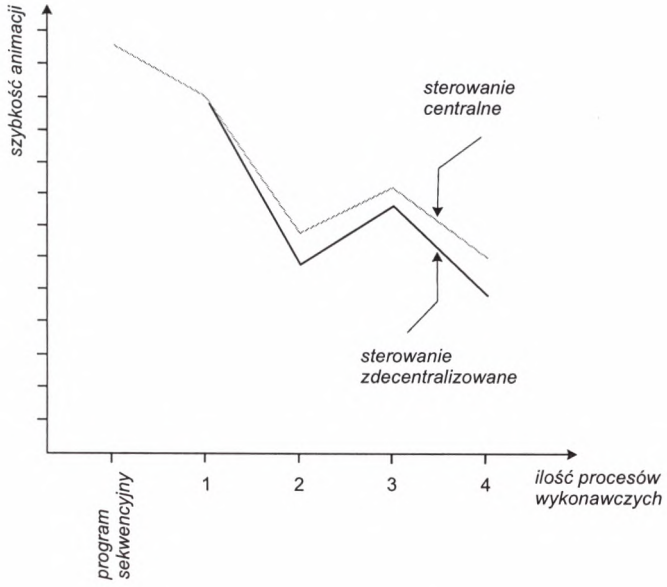
Rys. 15. Sześcian uderzający w elastyczną belkę



Rys. 16. Prostopadłościan spadajacy na powierzchnię cieczy



Rys. 17. Przykładowa scena testowa do pomiarów wydajności rozproszenia programu



Rys. 18. Pomiary czasu dla animacji z rys. 17

8. Podsumowanie

Zaprezentowany program bazując na technice symulacji pozwala na generację realistycznych animacji komputerowych. Zjawiska odkształceń obiektów, ich zderzeń oraz oddziaływania z cieczą są przedstawiane zgodne z oczekiwaniami obserwatora. Zastosowane algorytmy obliczania dynamiki sceny pozwalają, po niewielkich modyfikacjach na implementację programu w wersji rozproszonej przyspieszającej pracę systemu.

Dalsze prace koncentrować będą się głównie na poprawie jakości symulacji ośrodka ciągłego poprzez opracowanie nowych reguł automatu komórkowego, które umożliwią animację takich zjawisk jak np. falowanie cieczy.

Literatura

[1] O'Brien J.F., Hodgins J.K.: *Dynamic Symulation of Splashing Fluids*. Proceedings of Computer Animation '95, Genewa Switzerland, 1995

[2] Kass M., Miller G.: *Rapid, Stable Fluid Dynamics for Computer Grapfics*. SIGGRAPH'90 Conference Proceedings, 1990, 49–50

[3] Volino P., Courchesne M., Thalmann N.M.: *Efficient Techniques for Simulating Cloth and Other Deformable Objects*. SIGGRAPH'95 Conference Proceedings, 1995, 137–144

[4] Travis L., Hilton T.L., Egbert P.K., Parris K.: *Vector Fields: an Interactive Tool for Animation, Modelling and Simulation with Physically Based 3D Particle Systems and Soft Objects*. Computer Graphics Forum, 13 (3), 1994, 329–338

[5] Wcisło R., Dzwinel W., Kitowski J., Mościński J.: *Real-time Animation Using Molecular Dynamics Methods*. Computer Graphics and Vision, 1994

[6] Alda W., Dzwinel W., Kitowski J., Mościński J., Yuen D.A.: *Penetration Mechanics via Molecular Dynamics*.Supercomputing Institute Research Raport, UMSI 93/95, 1993

[7] Hockney R.W., Eastwood J.W.: *Computer Simulation Using Particles*. Mac Graw Hill, New York 1981

[8] Wcisło R., Dzwinel W., Kitowski J., Mościński J.: *Molecular Dynamics for Real Time Macro-World Phenomena Animation*. CCP5 inf. Q. Computer Simulation of Condensed Phases, 38, 1993, p. 25

[9] Dzwinel W., Alda W., Kitowski J., Mościński J., Wcisło R., Yuen D.A.: *Opportunities and Limitatios of the Molecular Dynynamics Methods Approach for Simulations of Plastic Deformation in Macro-Scale*. University of Minnesota Supercomputer Institute Research Raport, 1994

[10] Dzwinel W., Alda W., Kitowski J., Mościński J., Wcisło R.: *Macro-scale Simulations Using Molecular Dynamics Method. Molecular Simulation*, 15, 1995, 343–360

[11] Choryda W.: *Problemy detekcji i określenia reakcji obiektów w modelowaniu, symulacji i animacji komputerowej*. Praca dyplomowa. Politechnika Warszawska, 1994

[12] Egbert P.K., Parris K., Wingler S.H., Scott H.: *Collision-Free Object Movement Using Vector Fields.* Computer Graphics and Applications, 16(4): 18–24, 1996

[13] Wcisło R., Kitowski J., Mościński J.: *Cellular Automaton as a Fast Tool for Animation of Liquid in Multi-object Scenes.* Proceedings of the Sixth International Conference in Central Europe on Computer Graphics and Visualization WSCG'98. Plzen, Czech Republic, 1996, 417–423

[14] Toffoli T., Margolus N.: *Cellular Automata Machines – A New Enviroment for Modeling.* The MIT Prees 1987

[15] Wcisło R., Kitowski J., Mościński J.: *Distributed Simulation of a Set of ElasticMacro Objects.* Applied Parallel Computing – Computations in Phisics, Chemistry and Engineering Science – Proceedings of Second International Workshop, PARA'95, Lyngby, Denmark, August 1995, 534–539

[16] Wcisło R., Kitowski J., Mościński J.: *Pararellization of a Code for Animation of Multi-object System.* Applied Parallel Computing – Computations in Physics, Chemistry and Engineering Science – Proceedings of Second International Workshop, PARA'95, Lyngby, Denmark, 1996

Recenzent
prof. dr hab. inż. Jacek Mościński