CELLULAR AUTOMATON AS A FAST TOOL FOR ANIMATION OF LIQUID IN MULTI-OBJECT SCENES

Rafał Wcisło[1], **Jacek Kitowski**[1,2] **and Jacek Mościński**[1,2]
[1] Institute of Computer Science, AGH
Al. Mickiewicza 30, Kraków, POLAND
[2] Academic Computer Centre CYFRONET-Kraków
Ul. Nawojki 11, Kraków, POLAND
e-mail: `wcislo@icsr.agh.edu.pl`

ABSTRACT

The paper presents the usage of a cellular automaton (CA) in three-dimensional computer animations. The main goal is to achieve a fast, realistic motion of a set of bodies. In order to fulfil the demand of "reality" we try to employ the simulation methods based on physical laws. However, in scenes where different kinds of bodies (rigid, elastic and liquid) are animated together it is hard to find one universal paradigm for their simulation. Instead, we propose to use CA for simulating the liquid and the molecular dynamics (MD) method for other bodies. The paper describes the CA rules and its interaction with particles used in the MD method. Finally we present the animation algorithm (composed of CA, MD and visualization) distributed on networked Unix workstations.

Keywords: computer animation, cellular automaton, molecular dynamics, parallel algorithms

1. INTRODUCTION

The paper presents a computer program developed to perform 3D animations of macroscopic objects. The main goal of the system is to animate the scenes composed of:

- macroscopic, deformable solids with defined shape, mass, elasticity, colour, initial position and velocity,
- liquid medium with defined volume and density,
- infinitive planes (walls) with infinitive mass,
- gravitational field with defined strength and direction.

The user of the system can generate the animations of objects falling on the rigid surface or into the liquid, observe their deformations and interactions.

In order to effectively use the system it should fulfil the following assumptions:

- the movements of the objects should conform to the common sense of the viewer and to the physical laws,
- computations should be fast enough to meet the demand of "real-time animation" for as many animated scenes as possible.

There are many methods used for animating the behaviour of a liquid [Reeve83a, Kass90a, Szeli92a, Desbr95a, OBrie95a, Witki94a]. Most of them are suitable for situations where the fluid is the only object of an animated scene. In this paper we concentrate on the scenes in which solids and a liquid medium are present. From the computational point of view the solid bodies are very different from the liquid ones. It is hard to find an universal simulation scheme which could be used for both types. Instead of this, the system employs two different techniques – Cellular Automaton (CA) for a liquid and modified Molecular Dynamics (MD) for solids. To implement the interactions between solids and a liquid the "computational interface" between CA and MD has been developed.

Simulation of liquid dynamics can be performed using several methods [Dzwin94a, Dzwin95a]. First, the classical Navier-Stokes equations can be adopted. It is both complicated and time-consuming. Among other choices based on the particle approach like MD, lattice gas (a special kind of CA), Boltzman lattice gas and smoothed particle hydrodynamics [Monag92a, Vesel94a], CA method has been adopted due to its simplicity and high performance. The last issue is an important factor according to real-time or near real-time animation requirements. In spite of simplicity cellular automata used in liquid animation imitate real liquid properties in sufficient way.

We use MD approach to model solids and their mutual collisions. A solid is constructed with (quasi-) particles interacting each other with a potential. In this way elasticity of a solid can be modelled, introducing more reality into the animated scene.

Another issue of the project was to implement the animation system as a distributed program on a local network of Unix machines. Parallelization of the code gives the opportunity to perform animations faster, but brings on the problems connected with the message passing mechanism and the process synchronization.

2. STRUCTURE OF THE PROGRAM

Fig. 1 shows the scheme of a classical sequential version of the animation program (distributed version is discussed in the later sections).

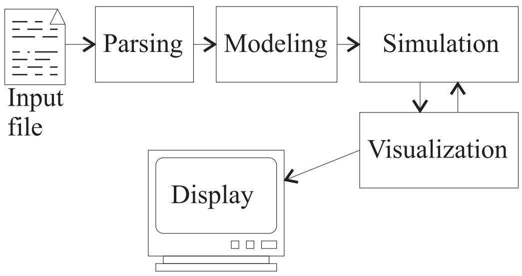

Fig. 1: The structure of the animation program – sequential version.

The composition of a scene is given in a input (text) file. The file describes the scene, objects, properties of the liquid medium, gravity and walls. It is read by the parser which fills in the internal data structures. In the modelling phase those structures are converted to data required by the simulation engines: the particles for MD and the cells for CA. The simulation module generates the position of every object in a given time step. Periodically the scene is displayed on a computer screen by the visualization process.

In the following sections we shall focus on the simulation module and computational algorithms implemented in it.

3. SIMULATION TECHNIQUES

During the simulation the MD and CA routines are called in a loop shown in Fig. 2.

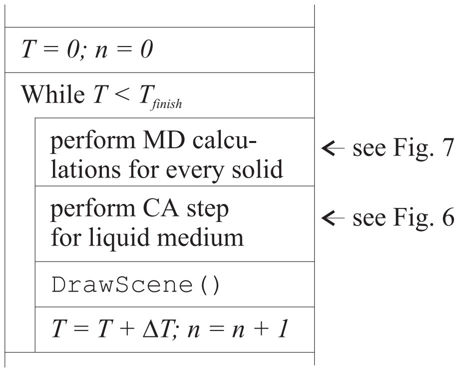

Fig. 2: The main loop of the animation module. T - time of the simulation; n - step of the simulation.

Although it would be very attractive to use the same simulation scheme for both liquid and solids (i.e. CA or MD), there are some reasons for giving up this idea:

- CA bases on the cells which uniformly divide the space of the animation scene (CA creates 3D cubic mesh) – this makes impossible to simulate deformations of solids (via CA approach) without introducing a very dense cell division which in turn would effect in time consuming computations,

- on the other hand in the animated scenes the volume of the liquid medium is much bigger than the total volume of all other objects – using MD for the liquid would dramatically slow down the animation.

3.1. Cellular Automata

To simulate the presence of the liquid medium in the animation scene the program uses 3D Cellular Automaton. There are two assumptions connected with the scene containing the liquid:

- there is the gravitational field present in the scene and the gravity vector is directed "down" (see Fig. 3.a),

- the liquid is contained in the 3D rectangular pool.

In the modelling stage of the program the pool is divided into cells (see Fig. 3.b). The size of each cell is equal to the average initial distance between interacting particles in the MD part (average $r_{0;i,j}$

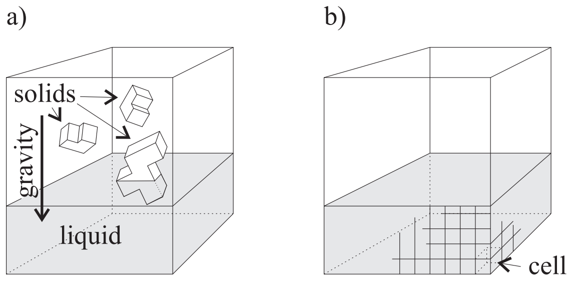

Fig. 3: Scene with the liquid: a) the scene with a liquid and solids, b) the pool divided into cells.

– see Eq. (8)). The design of CA allows to observe the following phenomena during the animation:

- friction between a solid object and the liquid,
- uplift pressure of the liquid,
- equalization of a liquid surface.

3.1.1. CA rules used in the animation program

In the beginning the cells containing the liquid are set to 1, all other cells are set to 0. To animate dynamics of the liquid with CA method the approach incorporates two substrategies (two sets of rules):

- "horizontal rule" is used to equalize the surface of the liquid,
- "vertical rule" is used to assure the constant volume of the liquid.

The former rule is used for mixing 2D CA which represent the layers of a liquid. The latter rule was added to implement the 3D nature of the animation.
At present there are no rules (but the program is still under development) which could assure the "wave effects" of the liquid.

3.1.2. Horizontal rule

The rule is performed independently for every horizontal CA layer. The role of the this rule is to distribute the liquid evenly in the layer. The rule is based on a cell-interconnection scheme called the *Margolus neighbourhood* [Toffo87a] in which every CA layer is partitioned into blocks of size 2×2 (see Fig. 4). Depending on the origin of the first block (upper-left in Fig. 4) there are four possible partitionings of the cells.
Let's denote four cells inside a block as LU, RU, LD, RD (see Fig. 4.a). The horizontal rule mixes the contents of those cells in one of four ways depending on the random generated number

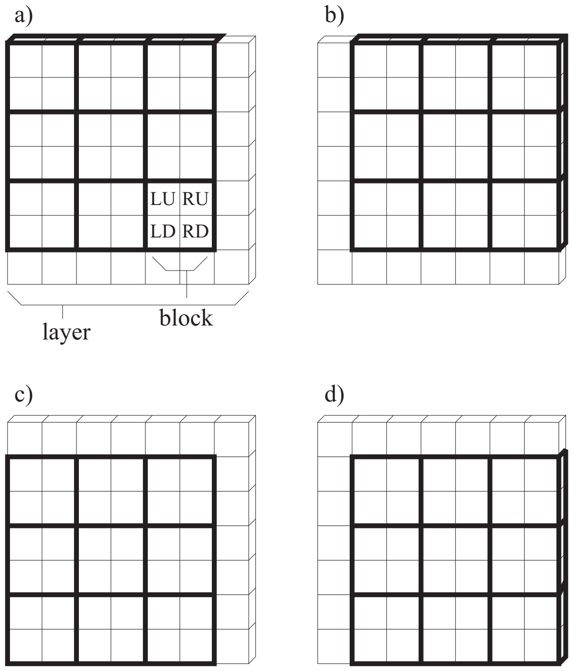

Fig. 4: Partitionings of a layer.

$n \in \{1, 2, 3, 4\}$:

$$\{LU', RU', LD', RD'\} = \tag{1}$$
$$= \begin{cases} \{LU, RU, LD, RD\} & \text{for } n = 1 \\ \{RU, LD, RD, LU\} & \text{for } n = 2 \\ \{RD, LU, RU, LD\} & \text{for } n = 3 \\ \{RD, LD, RU, LU\} & \text{for } n = 4 \end{cases}$$

where:
$\{LU', RU', LD', RD'\}$ – resulting quadruple of values for the given block.

3.1.3. Vertical rule

The vertical rule is used to ensure the constant volume of the liquid. In this rule the CA cells are divided into vertical pillars. The vertical rule bases on blocks of size 2×1 and two partitionings are possible (see Fig. 5).

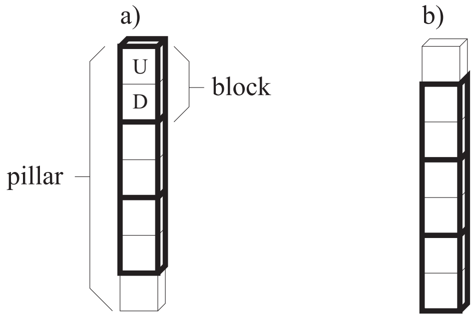

Fig. 5: Partitionings used in the vertical rule.

Let's denote the cells in a block as U and D. The vertical rule is:

$$\{U',D'\} = \begin{cases} \{U-1,1\} & \text{if } U>0, D=0 \\ \{U+1,D-1\} & \text{if } U<D, D>1 \\ \{U,D\} & \text{otherwise} \end{cases} \tag{2}$$

where:

$\{U',D'\}$ – resulting pair of values for the given block.

Fig. 6 shows the whole algorithm of CA invoked every time step of the animation.

```
for each layer of the CA                          ┐
  for each partitioning of the layer              │
    for each block of the partitioning            │ horizontal rule
      compute new values                          │
      of cells according to                       │
      Eq. (7)                                     ┘
for each pillar of the CA                         ┐
  for each partitioning of the pillar             │
    compute new values                            │ vertical rule
    of cells according to                         │
    Eq. (8)                                       ┘
```

Fig. 6: CA computations.

3.2. Molecular Dynamics

The Molecular Dynamics method [Hockn81a] is widely used for simulating the phenomena of the micro-world [Alda93a]. In order to use MD for macroscopic, deformable objects it must be appropriately adopted [Wcisl93a, Wcisl94a, Dzwin94a, Dzwin95a]:

- In the classical MD scheme the simulation is based on the particles (atoms, molecules or elementary particles). Presently the simulations with number of particles of order 10^9 (with supercomputers) can be performed, but it is still not enough to form a macroscopic object. In the modified algorithm of MD used in the animation program the term "particle" denotes the arbitrary chosen fragment of a given object.

- Modified MD limits the range of the particle-particle interactions only to the neighbouring particles. (two particles are considered neighbours if the distance between them is shorter than a predefined constant). This reduces the number of computations.

- In the classical MD scheme interactions between particles are based on the equations describing actual micro-world forces. In our approach the animation program uses the forces which approximate the right global elasticity of a solid (macroscopic) body.

- Because in our 3D pool several bodies can be animated, in order to simulate mutual collisions additional computations are performed [Wcisl98a].

The MD computations (see Fig. 7) are performed every time step ΔT. They were developed from the following typical assumptions:

- Particles obey Newton's laws of motion:

  $$\frac{d\mathbf{r}}{dT} = \mathbf{v} \tag{3}$$

  $$m\frac{d\mathbf{v}}{dT} = \mathbf{F} \tag{4}$$

  where:

  $\mathbf{r}$ – position of a particle,
  $\mathbf{v}$ – velocity of a particle,
  $\mathbf{F}$ – total force acting on a particle,
  m – mass of a particle,
  T – time.

- Equations of motion are solved with the leap-frog scheme [Hockn81a]. This scheme is widely used in MD simulations with short-range interactions due to its advantages, like performance, numerical stability and others:

  $$\mathbf{v}_{n+\frac{1}{2}} = \mathbf{v}_{n-\frac{1}{2}} + \frac{\Delta T}{m}\mathbf{F}_n \tag{5}$$

  $$\mathbf{r}_{n+1} = \mathbf{r}_n + \Delta T \mathbf{v}_{n+\frac{1}{2}} \tag{6}$$

  where:

  n – step of the calculations.

- In order to take into account the friction force (defined as $\mathbf{F}_{frict} = -\lambda\mathbf{v}$) between the particles and the liquid medium Eq. (5) is modified to the form:

  $$\begin{gathered} \mathbf{v}_{n+\frac{1}{2}} = \tfrac{1-\phi}{1+\phi}\mathbf{v}_{n-\frac{1}{2}} + \tfrac{1}{1+\phi}\tfrac{\Delta T}{m}\mathbf{F}_n \\ \phi = \lambda\tfrac{\Delta T}{2m} \end{gathered} \tag{7}$$

  where:

  λ – friction factor.

- The force of the particle-particle interaction in calculated from the equation of potential:

  $$V_{i,j}(r) = \frac{1}{2}k\left(r_{i,j} - r_{0,i,j}\right)^2 \tag{8}$$

  where:

  $V_{i,j}$ – interaction potential between particles i and j,
  $r_{i,j}$ – distance between particles i and j,
  $r_{0;i,j}$ – initial distance between particles i and j (which assures the equlibrium state of the particles),
  k – factor describing the elasticity of the interaction.

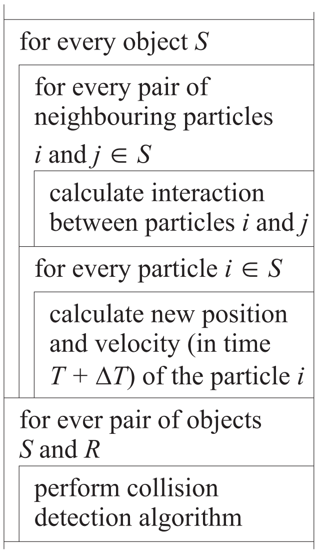

Fig. 7: MD computations.

3.3. Interactions between MD and CA structures

The interactions between particles of the MD models and CA cells base on the following assumptions:

- the liquid and the solid object cannot occupy the same space,
- only particles laying on the surface of a solid object may interact with the liquid.

When MD algorithm calculates the forces acting on a particle it looks whether the cells neighbouring to the particle are filled with a liquid. In that case if the particle lays on the surface of an object the value of λ (see Eq. (7)) is modified to reflect the solid-liquid friction. In addition the gravity force is changed (according to the density of the liquid) to ensure the proper uplift pressure of the liquid.

When a new position is given to the particle by the MD algorithm, the program calculates the coordinates of a CA cell the particle is in. If the cell in not filled with the liquid then nothing happens (see Fig. 8.a), otherwise (Fig. 8.b) the value of the cell is set to 0 (liquid is removed) and one of the randomly chosen, neighbouring cells (which must not be occupied by any particle) inherits the liquid (Fig. 8.c). The subsequent changes in the CA state (value '2' moves right and up – see Figs. 8.d-f) are caused by the horizontal and vertical rules of the CA.

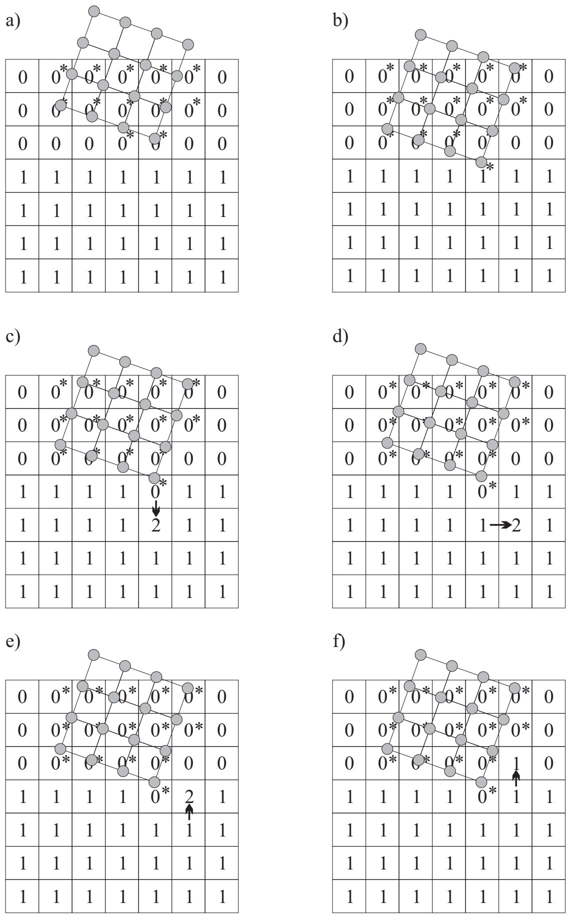

Fig. 8: The solid object interacting with the liquid. Cells occupied by particles are marked with '*'. (2D drawing is presented for simplicity – the actual computations are 3D)

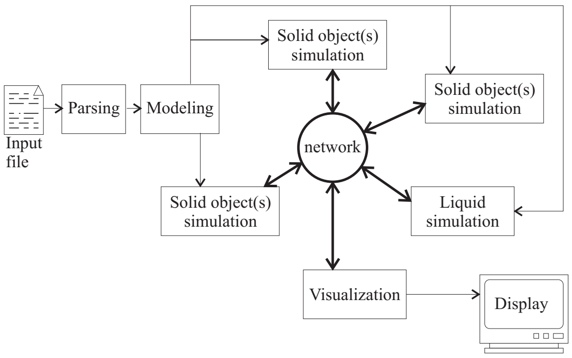

Fig. 9: Structure of the distributed version of the animation program.

4. PARALLELIZATION OF THE PROGRAM CODE

In order to speed up the animation the distributed version of the system with MD only (without CA part) was implemented [Wcisl95a, Wcisl96a]. The Fig. 9 shows the structure of the parallel program with a new module performing the CA calcula-

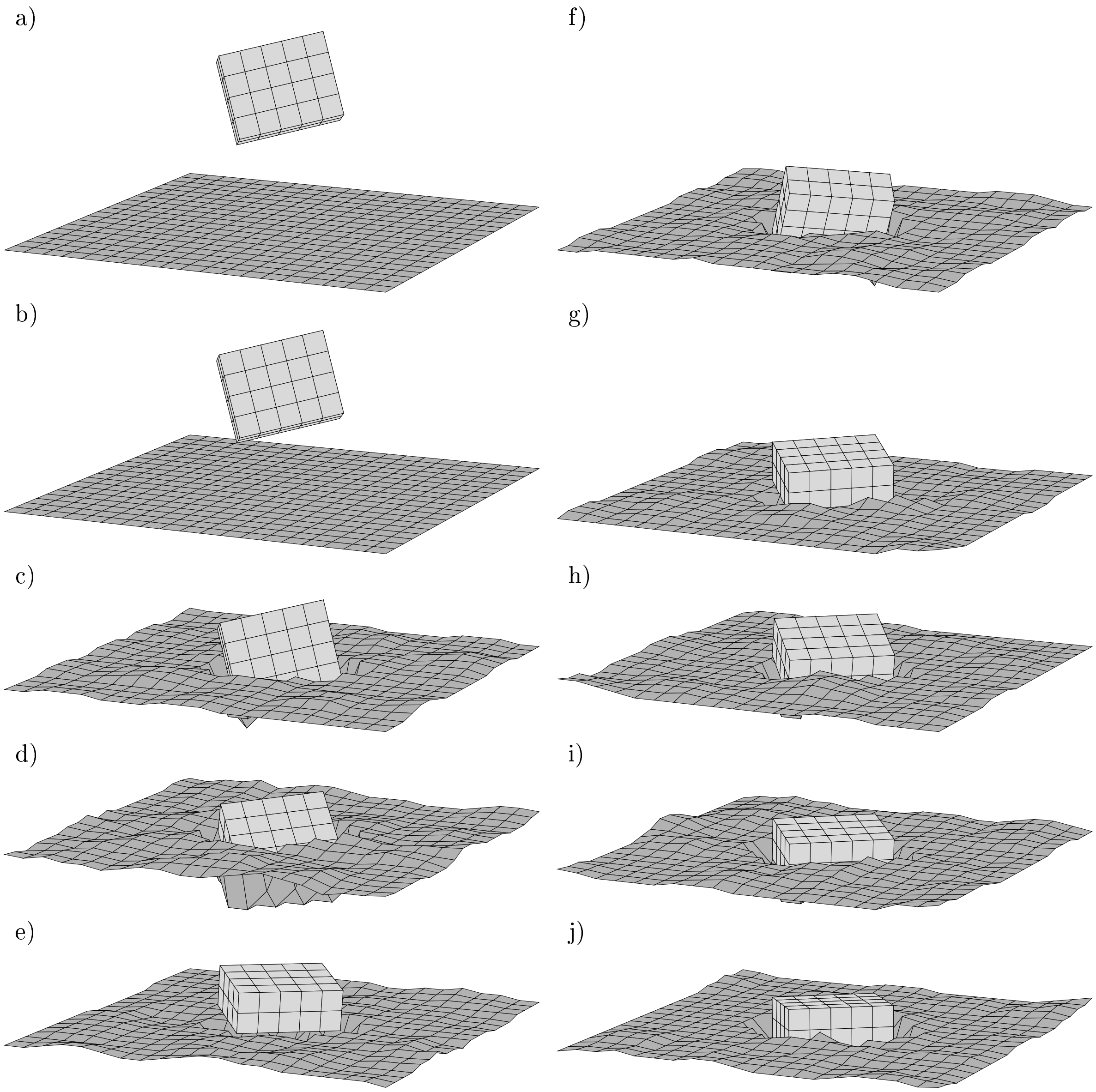

Fig. 10: Snapshots from the animation of a solid dropped into the liquid (the surface of the liquid shown only).

tions.
Comparing to the sequential code the main differences are:

- there are several simulation modules running on separated computers connected via the local network,
- the simulation modules are not universal – some of them calculate MD algorithms and simulate solid objects, one process performs CA calculations and simulates the liquid medium and one workstations is dedicated for the visualization process (typical MIMD model of computations),
- information about the components of the animated scene are passed between computers (using PVM library) via the network,
- special algorithms are periodically invoked to ensure the load-balancing between workstations and synchronization of the simulation.

5. RESULTS AND SUMMARY

Fig. 10 presents some snapshots from a sample animation of a rectangular solid dropped into a liquid. The program was ran on the HP D270/1 workstation. To generate and display the animation the sequential program needed 9.14 seconds (1000 simulation steps was computed).

The development of the presented animation system proved that:

- cooperation the MD and CA paradigms can be used for animation purposes,
- the MD and CA can be connected in one simulation of a scene composed of solids and liquid,
- the use of MD and CA allows to write animation programs which can run in parallel speeding up the animation.

The future research will mainly concentrate on enhancing the rules of CA in order to implement more phenomena connected with the liquid, e.g. waves and splashes.

Acknowledgements. The work was supported by AGH Grants 11.120.16 and 10.120.420.

REFERENCES

[Alda93a] Alda, W., Dzwinel, W., Kitowski, J., Mościński, J., Yuen, D.A. *Penetration Mechanics via Molecular Dynamics.* University of Minnesota, AHPCRC Report, Report No. 93-037, 1993

[Desbr95a] Desbrun, M., Gascuel, M.-P. *Animating Soft Substances with Implicit Surfaces.* SIGGRAPH'95 Conference Proceedings, 1995

[Dzwin94a] Dzwinel, W., Alda, W., Kitowski, J., Mościński, J., Wcisło, R., Yuen, D.A. *Applications of molecular dynamics method for simulations in macro-scale.* University of Minnesota, AHPCRC Report, Report No. 94-032, 1994

[Dzwin95a] Dzwinel, W., Alda, W., Kitowski, J., Mościński, J., Wcisło, R. *Macro-scale Simulations Using Molecular Dynamics Method.* Molecular Simulation, 15, 1995.

[Hockn81a] Hockney, R.W., Eastwood, J.W. *Computer Simulation Using Particles.* MacGraw Hill, 1981.

[Kass90a] Kass, M., Miller, G. *Rapid, Stable Fluid Dynamics for Computer Graphics.* SIGGRAPH'90 Conference Proceedings, 1990

[Monag92a] Monoghan, J.J. *Smoothed Particle Hydrodynamics.* Ann. Rev. Astron. Astrophys., 30, 1992

[OBrie95a] O'Brien, J.F., Hodgins, J.K. *Dynamic Simulation of Splashing Fluids.* Proceedings of Computer Animation '95, Geneva Switzerland, 1995

[Reeve83a] Reeves, W. *Particle Systems – a Technique for Modelling a Class of Fuzzy Objects.* ACM Transactions on Graphics, 2(2), 1983

[Toffo87a] Toffoli, T., Margolus, N. *Cellular Automata Machines – A New Environment for Modeling.* The MIT Press, 1987.

[Szeli92a] Szeliski, R., Tonnesen, D. *Surface Modelling with Oriented Particle Systems.* SIGGRAPH'92 Conference Proceedings, 1992

[Vesel94a] Vesely, F.J. *Computational Physics.* Plenum Press, New York, 1994

[Wcisl93a] 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.

[Wcisl94a] Wcisło, R., Dzwinel, W., Kitowski, J., Mościński, J. *Real-time Animation Using Molecular Dynamics Methods.* Computer Graphics and Vision, 1994.

[Wcisl95a] Wcisło, R., Kitowski, J., Mościński, J. *Distributed Simulation of a Set of Elastic Macro Objects.* Applied Parallel Computing – Computations in Physics, Chemistry and Engineering Science – Proceedings of Second International Workshop, PARA'95, Lyngby, Denmark, 1995.

[Wcisl96a] Wcisło, R., Kitowski, J., Mościński, J. *Parallelization of a Code for Animation of Multi-object System.* Applied Parallel Computing – Computations in Physics, Chemistry and Engineering Science – Proceedings of Third International Workshop, PARA'96, Lyngby, Denmark, 1996.

[Wcisl98a] Wcisło, R., Kitowski, J., Mościński, J. *Animation of Group of Elastic Objects Based on Distributed Physical Simulation.* In preparation for GKPO'98 conference in Tomaszów Mazowiecki, Poland, 1998