Behavioural Simulation in Voxel Space

Hongwen Zhang

SAGE Systems Division
Valmet Automation
Calgary, Alberta, Canada T2W 3X6

Brian Wyvill

Department of Computer Science
University of Calgary
Calgary, Alberta, Canada T2N 1N4

Abstract

In this paper we present a framework for behavioural simulation. A uniform voxel space representation is used to implement the environment mechanism of the framework. An example environment is presented where actors with olfactory sensors are able to direct their motions according to a scent of the chemicals in the voxel space based on mass transfer theory. Objects in the enviroment are scan converted to the voxel representation to facilitate collision detection. An example of using the framework to simulate the behaviour of a group of artificial butterflies is used to demonstrate the ideas of this research.

1 Introduction

Complex images of a population of objects can be automatically generated by modeling the simple behaviour of each individual object and the interaction between the objects. This approach is termed by Craig W. Reynolds as *behavioural animation* [12]. Reynolds noticed that scripting the paths of a large number of individual objects such as a flock of birds is a very difficult and tedious task in computer animation. He demonstrated that behavioural animation is a more efficient and robust way to accomplish this task. The basic idea of behavioural animation is that the complex paths can be generated by the interaction between the individual objects. Hence only the behaviour of each individual object needs to be modeled. The paths can then be generated by simulating these models. This differs from the more conventional computer animation methods which only models the shape and physical properties of the characters. When visualized, a simulation of a behaviour model can generate a sequence of dynamic, intentional, and complex pictures. This makes behavioural animation a very attractive high level modeling method in computer animation.

Generally speaking, a behavioural animation process includes three related steps. They are :

- modeling the behaviour of each individual object,
- simulating a population of these behaviour models,
- visualizing the simulation processes.

In the later literature [5], the modeling and the simulation steps in behavioural animation are referred to as *behavioural simulation*. Compared with behavioural animation, the concept of behavioural simulation has a larger scope. It not only includes behavioural animation, but also includes those applications that use similar modeling and simulation principles to produce complex and static images, such as [4].

The key problems in behavioural simulation are :

(1) Collecting behavioural data from the observation of living systems.

(2) Behaviour representation. This concerns how to better represent the observed behaviour in a computer system.

(3) Modeling the world view and the associated sensing mechanism for each individual object.

(4) Path generation by reasoning on the behaviour representation with the sensed information.

(5) The attachment of the details of the motion with the generated path, such as the wing flip of birds and the gaits and body swing of a dancer, etc.

This paper addresses the above third problem. There are two related sub-problems. The first is how to provide a world view for each object. Any solution to this problem has to provide a mechanism to represent the neighborhood of each object. The second sub-problem concerns how an object obtains information from the outside world. In order to solve this problem, a sensing mechanism has to be implemented for all the objects.

Many previous approaches use a database to store geometric information about the objects in the system. This database provides a world view which is shared by the objects. Each object can then abstract information about its environment by querying the database and computing its relationship with all the other objects. This process has to be applied to all the objects in the system. It is an expensive operation which has a complexity of $O(n^2)$, where n is the number of objects. To reduce this complexity, Ned Greene [4] used a voxel space to provide a subdividing strategy for geometric information. A **voxel space** is a region of three dimensional space partitioned into identical cubes. Each

cube is called a **voxel**. His approach effectively reduces the above complexity to $O(n)$. However, his original method is mainly concerned with using voxel space for the purpose of generating static images of vine like plants. The plants have very simple geometric shapes. Their behaviour is mainly affected by the geometric information distribution in the neighborhood.

In this research we propose a framework for behavioural simulation. The framework not only employs vision perception but also olfaction perception which is otherwise very difficult to implement in a conventional approach. By using voxel space, the complexity of the perception process of each object is reduced to linear, within the limitation of a voxel space's resolution.

2 A FRAMEWORK FOR BEHAVIOURAL SIMULATION

2.1 THE IMPORTANT CONCEPTS

Behaviour simulation assumes that the complex phenomena of a large system is caused by the interactions of its components. Hence only these components need to be explicitly modeled. The global phenomena of the large system can be generated automatically by simulating a group of explicitly modeled components. Each model of such a component is called an **actor**.

One loosely defined concept in behavioural simulation is the **environment**. It is used here to refer to the mechanism that represents all the outside factors for each individual actor. These factors are filtered and transformed by each actor to its internal representation of its neighborhood. The mechanism that an actor uses to filter and transform a certain type of information from the environment is called the **sensor** of that type of information. In a simulation, a set of actors are coupled by their environment. The set of these actors and the environment is called a **behavioural system**.

The **behaviour** of an actor is the internal representation of our observation of the corresponding dynamic object. This internal representation is used in a simulation to map the sensed information and the current state of the actor to a path of state changes. This mapping is referred to as **path planning**. **Actuators** are used to execute these state changes by interpreting and/or interpolating the path. In behavioural simulation, these are usually programs or funtions that directly manipulate the state parameters of actors. The examples of actuators are the motor controller in [15], or the muscles in [17].

2.2 THE FRAMEWORK

Based on the above discussion, this research establishes a framework for building behavioural simulation applications. The framework is composed of two levels. The first level is the model of behavioural systems (Figure 1). It has a set of actors, an environment shared by all the actors, and a global clock for controlling the animation frequency and synchronizing each actor with the changes in the environment. The second level is the model of actors (Figure 2). An actor senses the environment to detect any signif-

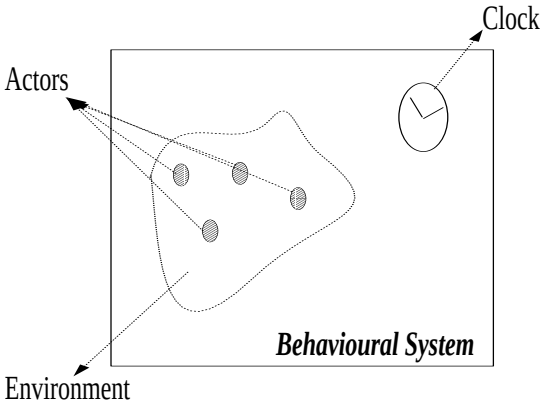

Figure 1: A behavioural system

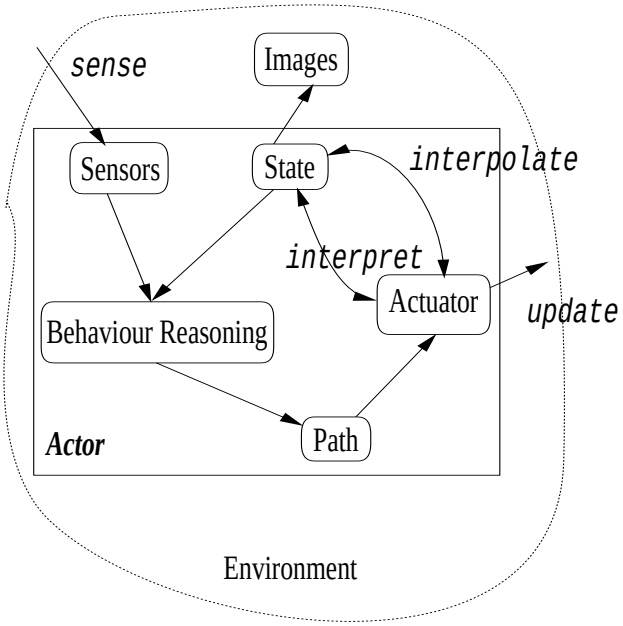

Figure 2: An actor

icant changes. The path planning mechanism plans the path according to the sensed information and the current state of the actor. The actuators interpret and interpolate the path by reading and setting the current state. Images can be generated by rendering the state.

The behaviour reasoning mechanism can be implemented by using heuristic rules [3], a network of nodes [20, 16], a genetic algorithm [10], or a hybrid approach using both a genetic algorithm and a network of nodes [14]. The path can be represented as qualitative terms or a set of parameters to drive the actuators. The implementation of the actuators can range from simply interpreting the qualitative terms, to the interpolation of the actors state with key-framing or physically-based modeling techniques [15].

The environment and the sensors are implemented in this research with voxel space representation.

3 VOXEL SPACE AS THE ENVIRONMENT

The central idea of using a voxel space as the environment is to distribute the information about each actor in the voxels. This way, an actor can get information about any changes in its neighborhood, such as an approaching obstacle, a food source, etc., by sensing the neighboring voxels. An actor can also let its activity be known to others by leaving its "footprint" in the voxel space. In our approach, each voxel can hold a list of *information parcels*. Each information parcel corresponds to a type of information. The types of information currently being modeled are geometric information and olfactory information. Each information parcel is represented as the following triplet :

$$(t, i, ids)$$

where :

t — the information type. Currently, it is either *geometry* or *olfaction*.

i — the intensity of the information. For geometric information, i is always 0 or 1, which indicates the voxel is empty or occupied. For olfactory information, i is the combined scent intensity of the actors that contribute their scent in this voxel.

ids — A set of actor identifiers. Each actor in this set has contributed to the intensity of the information in this parcel.

When sensing the neighborhood, an actor only needs to search the list of parcels associated with each voxel.

3.1 GEOMETRIC INFORMATION IN VOXEL SPACE

A set of elegant algorithms for distributing the geometric primitives such as three dimensional lines, polygons, polyhedra, circles, cylinders, etc. were devised by Arie Kaufman [7, 6]. These algorithms are used in this research to scan convert geometric primitives.

Complex shapes can be constructed hierarchically by using these primitives. The OpenInventor [19]

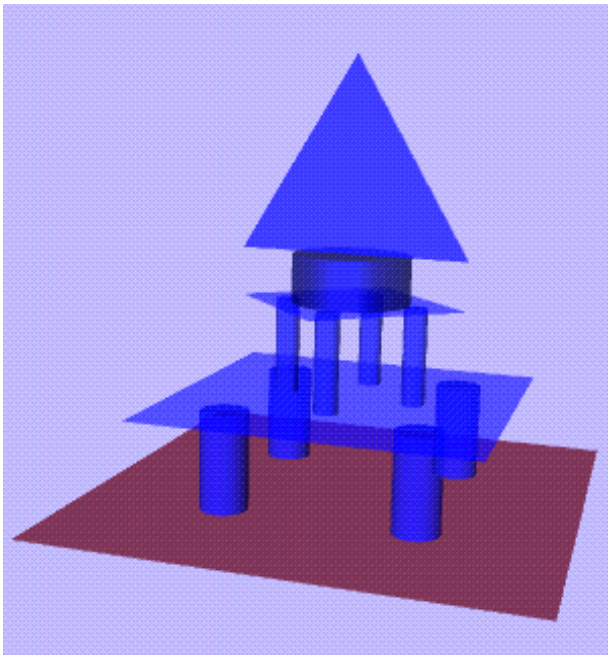

Figure 3: A glass temple defined in OpenInventor

scene graph is used as the data structure to represent a hierarchical geometric shape. A scene graph consists of one or more nodes, each of which represents a geometry, property (e.g material, etc.), transformation, or grouping object. Hierarchical scenes are created by adding nodes as children of grouping nodes, resulting in a directed acyclic graph. Figure 3 shows a glass temple constructed with an OpenInventor scene graph.

The algorithm for scan converting a scene graph in the voxel space is described as the following :

```
scanConvert(GroupNode gNode,
      Transformation aMatrix){
  for each child x of gNode{
    m = the transformation of x
          under gNode;
    //Calculate the combined
    //transformation
    m = m*aMatrix;
    switch(the node type of x){
      case GroupNode :
        scanConvert(x, m);
        break;
      case ShapePrimitive :
        x.transform(m);
        //Use Kaufman algorithms
        x.scanConvert();
        break;
      default :
        break;
    } //end switch
  } //end for loop
} //end scanConvert
```

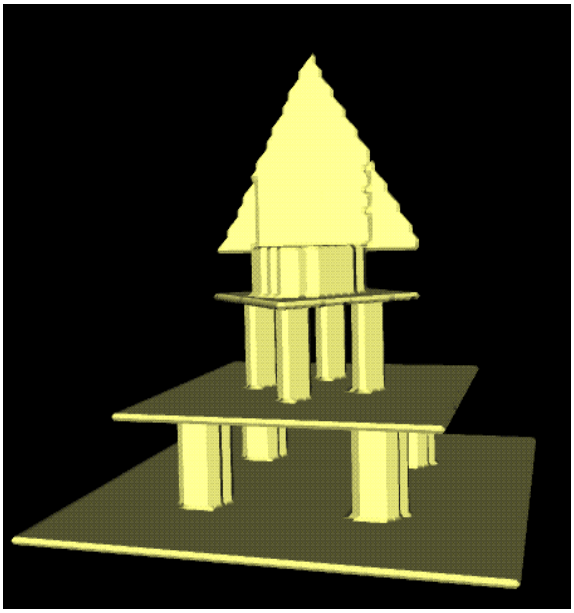

Figure 4: The glass temple in voxel space

Figure 4 is the scan converted glass temple in the voxel space with a $75 \times 75 \times 75$ resolution.

3.2 OLFACTORY INFORMATION IN VOXEL SPACE

Olfactory information associated with searching for food, finding a mate, or avoiding a predator plays an important role in the activities of many animals. The following is a vivid depiction of how the African hyaena uses olfactory information to guide itself through the darkness [1] :

> "With a sniff a hyaena can perceive not only here and now but, simultaneously, a whole series of events stretching back into the past. .... The pasted patches of grass must shine in the distance like lighthouses and the pack's trails, perfumed by their paws, must stretch ahead like lines of reflector studs down the middle of a motorway."

Previous approaches in behaviour simulation applications have not modeled the effect of olfactory information on the actors.

According to physics [13], given a chemical source, its scent intensity at any point in a three dimensional space is determined by the molar concentration of this chemical at the point. The *molar concentration* of a chemical A is defined as the number of moles of A per unit volume. We assume that the molar concentration of any chemical substance within a voxel is constant, whilst different voxels may have different molar concentration. If the air is static, i.e. no wind, the difference of the molar concentration between the neighboring voxels will suggest the direction of the chemical source. Hence, an actor can reach the source by "sniffing" a set of voxels.

Now the question is how to calculate the molar concentration in each voxel. The answer can be found in the mass transfer theory [8]. The mass balance of a chemical A in a voxel is modeled as:

$$\nabla \bullet N_A + \frac{\partial C_A}{\partial t} - R_A = 0 \qquad (1)$$

where :

- ∇ — the differential operator, $\frac{\partial}{\partial x} + \frac{\partial}{\partial y} + \frac{\partial}{\partial z}$
- N_A — the molar flux at time t. The *molar flux* of A is defined as the number of moles of A that pass through a unit area per unit time.
- C_A — the molar concentration of A in the voxel.
- R_A — the rate of reproduction of A per unit volume. It is non-zero in the voxels that contain the sources of A and zero elsewhere.

If we only consider the diffusion in the static air, then the molar flux N_A is governed by the *Fick's law of diffusion* :

$$N_A = (-D\frac{\partial C_A}{\partial x}, -D\frac{\partial C_A}{\partial y}, -D\frac{\partial C_A}{\partial z}) \qquad (2)$$

where D is the mass diffusive coefficient of A in the air. It is a constant if the temperature and pressure are all constant.

Solving equation 1 and equation 2 for C_A with the *Forward Time Center Space(FTCS)* method [11], we have :

$$C^{t+1}_{A(i,j,k)} = \Delta t R^{t+1}_{A(i,j,k)} + C^{t}_{A(i,j,k)} + \frac{\Delta t}{s} F^{t}_{A(i,j,k)}$$

where:

- $C^t_{A(i,j,k)}$ — the molar concentration of A in the voxel (i,j,k) at time t.
- $R^t_{A(i,j,k)}$ — the reproduction rate of A in the voxel (i,j,k) at time t.
- s — the size of each voxel.
- $F^t_{A(i,j,k)}$ — the net flux of A in the voxel (i,j,k).

The net flux of A can be expressed as

$$F^t_{A(i,j,k)} = \sum_{m,n,p \in \{-1,1\}} C^t_{A(i+m,j+n,k+p)} - 6C^t_{A(i,j,k)}$$

Figure 5 illustrates the scent distribution of a point chemical source in the voxel space.

In a complex scene, geometric information can interfere with olfactory information, e.g. a scent will spread a gas around an object. To approximate this effect, the net flux is modified into :

$$F^t_{A(i,j,k)} = \sum_{m,n,p \in \{-1,1\}} T^t_{(i+m,j+n,k+p)} C^t_{A(i+m,j+n,k+p)} - \sum_{m,n,p \in \{-1,1\}} T^t_{(i+m,j+n,k+p)} C^t_{A(i,j,k)}$$

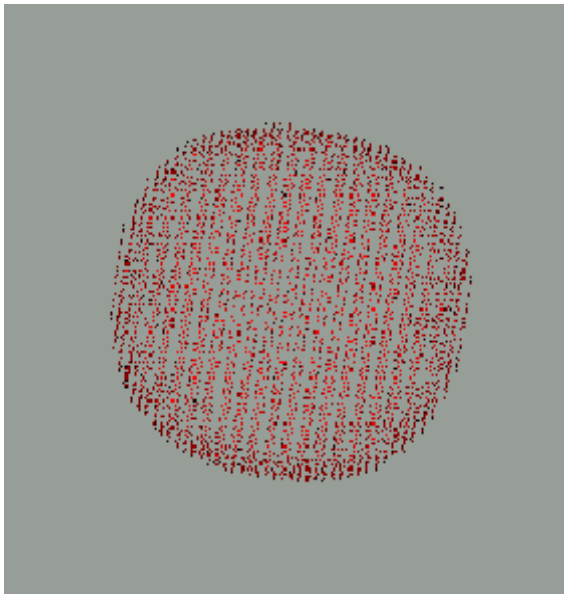

Figure 5: The scent distribution of a point source (See colour slide)

where $T^t_{i,j,k} \in \{0,1\}$. It is 1 if the voxel (i,j,k) is not occupied. Otherwise, it is 0.

Figure 6 shows the result of putting the same point source in the center of a box with an open gap on one end. Figure 7 only shows the scent distrubution of Figure 6. Note that it is shaped by the box.

4 THE SENSORS

A set of sensors are implemented. An actor can use these sensors to get stimuli from its neighboring voxels. This is achieved by scanning the information parcels in those voxels. The action of the actor can then be planned. Hence both the principles of the stimuli/response paradigm and the spatial locality are enforced.

4.1 THE VISION SENSOR

The vision sensors are used to get images of the actor's neighborhood and to extract abstract information, e.g. collision, found food, etc., from these images with some simple perceptual processes. Each vision sensor is constrained by its view angle and view range.

The image obtained by a vision sensor is a 2D latitude-longitude array. A ray is cast from the sensor center towards each element of this array. This ray casting can be accomplished by a simple 3D version DDA algorithm [2]. If the ray passes through an occupied voxel within the view range, the reference of this voxel will be recorded in the corresponding element of the latitude-longitude image. Otherwise, a null value is assigned to the element. Figure 8 illustrates a vision sensor.

The perceptual processes are task oriented. For example, if the task is to avoid collision, the associated process will scan the latitude-longitude array to look

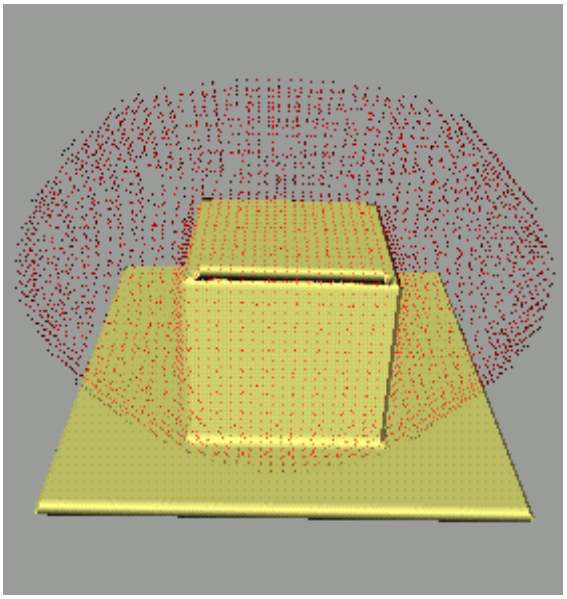

Figure 6: Interfering scent distribution with geometric information

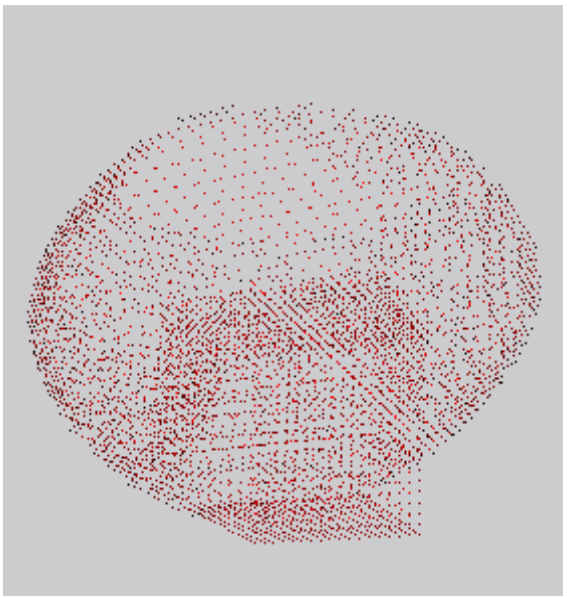

Figure 7: Olfactory information distribution is shaped by geometric information

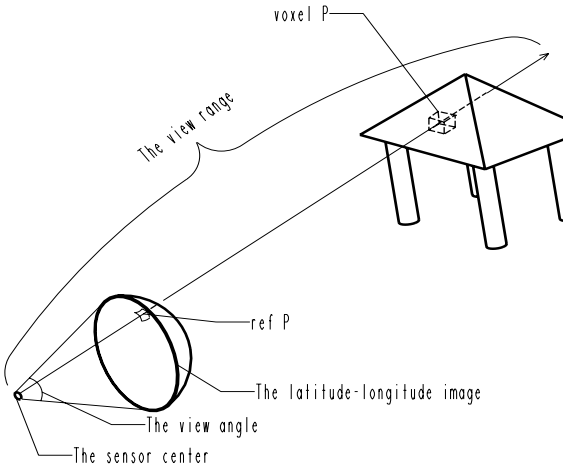

Figure 8: The vision sensor

for an area with enough null elements that will allow the whole body of the actor to pass through. This is actually a very easy task in the simulated environment of a voxel space since all the information contained in a voxel, e.g. depth, who occupies it, etc. are readily accessible through the elements in the latitude-longitude array.

4.2 THE TACTILE SENSOR

The tactile sensor simply tests if the voxel at the sensor center is occupied or not.

4.3 THE OLFACTORY SENSOR

The olfactory sensor is a simulated nose. It is composed of a set of smell-detecting cells with one cell at the sensor center. Each cell can get the scent intensity of the associated voxel. The intensity gradient can be approximated with a vector which points from the sensor center to the cell that detects the biggest intensity value. Figure 9 illustrates an olfactory sensor in a 2D voxel space. Sensors that detects the gradient of a chemical concentration can be found in the living systems. For example, *Caenorhabditis elegan*, a type of nematode, can sense the concentration gradient of the chemical released by its food and move along the gradient [18].

5 AN EXAMPLE — THE JOURNEY OF THE DIGIFLIES

As an example, a set of artificial butterflies called *digiflies* are simulated. Each digifly is equipped with a vision sensor, an olfactory sensor, and a tactile sensor. The behaviour of the digifly is defined by the following rules :

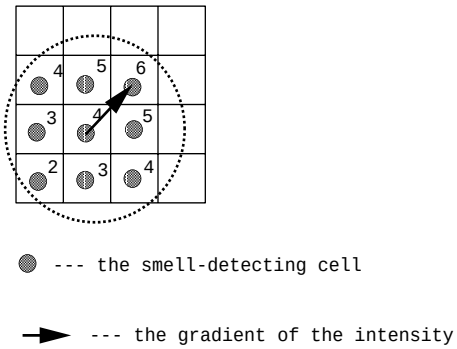

Figure 9: The olfactory sensor in a 2D voxel space

avoid collision It should avoid fly into the other digiflies or any obstacles. There is one exception to this rule, that is a hungry digifly can collide with its food, i.e. the flowers.

seek food The energy level of each digifly drops with the passage of time. When this level is below a threshold, the digifly will begin to seek for food. If it can "see" a flower, it will fly towards the flower. If it can "smell" the food, it will fly to the direction of the food.

eat food If the tactile sensor detects that the digifly touches a flower, the digifly will recharge itself, i.e. set its energy level to a maximum value. It will then fly away.

fly towards other digiflies When the digifly does not detect any collision and it is not hungry or can not find any food, it will fly towards the center of other digiflies that are detected by its vision sensor.

random flight When none of the above rules are triggered, a lonely digifly will randomly change its heading direction to wander around.

The behavioural simulation framework developed in this research allows the above rules to be formally specified with symbolic expressions of *CCS, the calculus of communicating systems* [9]. In the simulation, these rules will be triggered according to the local circumstance of each digifly.

A scene containing the glass temple in Figure 3 and a flower in a glass box is used. There is a gap on the top side of the box. When a set of digiflies are released into the scene, one should expect that those digiflies will flock towards the box, fly through the gap, eat the food, and fly away. This is exactly what happened into the simulation. Figure 10 shows the trajectory of two digiflies in the scene. Figure 11 is a snapshot of a group of sixty digiflies.

The hardware used in this research are the Silicon Graphics Indigo workstations. The OpenInventor is

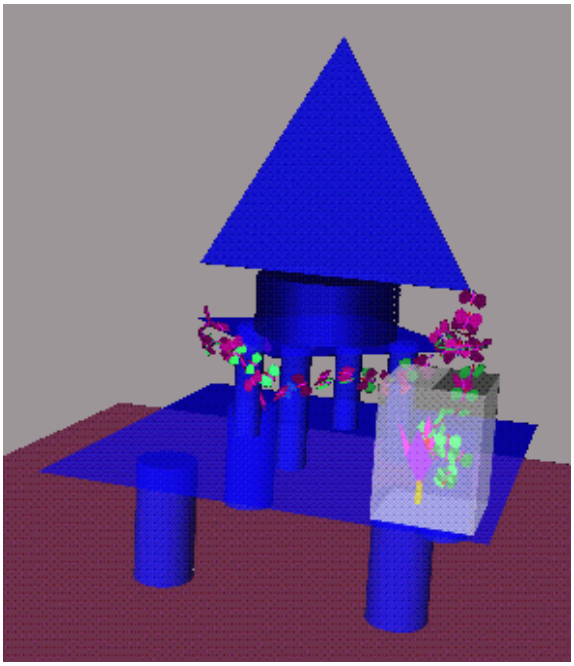

Figure 10: The trajectory of two digiflies

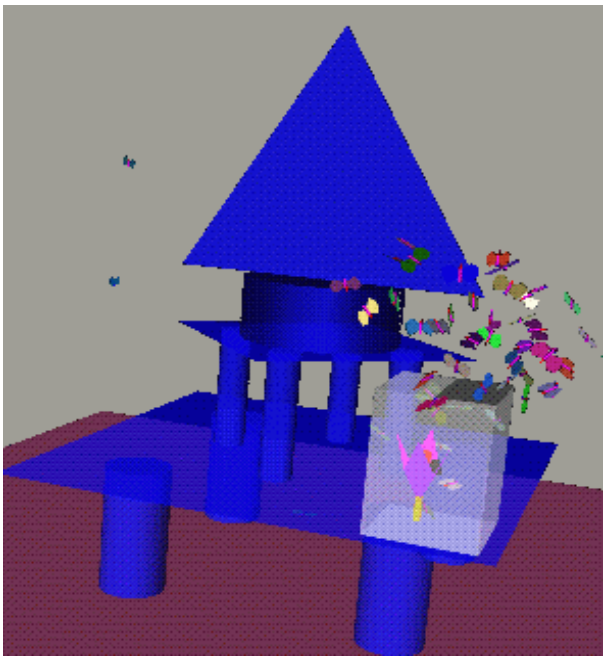

Figure 11: A snapshot of sixty digiflies

used as the graphical system. The visualization of the scent distribution is achieved by the SGI Explorer. The program is written in C++.

6 CONCLUSION

We have presented a unified framework which endorses the important concepts in behaviour simulation. We have shown how this framework which uses a voxel space representation can be applied to a group of actors exhibiting emergent behaviour.

As demonstrated in this research, there are several advantages to our approach. Firstly, it is intuitive. A living system knows it touches an obstacle by sensing that the touching point is occupied by the obstacle, not by solving equations. Hence, it is easy to be used in the behavioural simulations that use empirical knowledge. Secondly, the data structure of the voxel space is very simple. Any geometric models can be converted into the voxel representation. This implies that the actors do not have to have the same geometric data structure in order for them to interact with each other. Therefore, the voxel representation can serve as a common ground for different parties to work collectively to build a complex scene animation.

These advantages must be traded against the loss of precision since voxel space modeling operates in discrete space. It is possible to generate undesired animations. For example, the feet of a walking animal may sometimes penetrate slightly into the ground or float above the ground. Our future work will look at adaptive voxel subdivision and other possible approaches to ameliorating this problem as well as optimizing the existing algorithms to enable the use of actors with more complicated structures and behaviour.

Acknowledgments

The authors would like to thank the many students and faculty members who have contributed towards the GraphicsJungle project at the University of Calgary. This work is partially supported by the Natural Sciences and Engineering Council of Canada. Hongwen Zhang would like to thank the SAGE Systems Division of Valmet Automation for sponsoring him to present this paper to Computer Animation 97.

References

[1] David Attenborough. *The Trials of Life — A Natural History of Animal Behavior*. Little, Brown and Company, 1990.

[2] John Cleary and Geoff Wyvill. Analysis of an Algorithm for Fast Ray Tracing Using Uniform Space Subdivision. *The Visual Computer*, 4(2), 1988.

[3] Bill Coderre. Modeling behavior in petworld. In *Artificial Life*, 1989.

[4] Ned Greene. Voxel space automata : Modeling with stochastic growth processes in voxel space. *ACM SIGGRAPH 89*, 23(3):175–184, 1989.

[5] D. R. Haumann and R. E. Parent. The behavioral test-bed : Obtaining complex behavior from

simple rules. *The Visual Computer*, 4(6):332–347, 1988.

[6] A. Kaufman. Efficient algorithms for 3d scan-conversion of parametric curves, surfaces, and volumes. *ACM SIGGRAPH 87*, 21(4):171–179, 1987.

[7] A. Kaufman and E. Shimony. 3d scan conversion algorithm for voxel based graphics. In *ACM Workshop on Interactive 3D Graphics*, pages 45–75, NC, October 1986.

[8] A.F. Mills. *Basic Heat and Mass Transfer*. Richard D. Irwin, 1995.

[9] Robin Milner. *Communication and concurrency*. Prentice Hall, 1 edition, 1989.

[10] J. Thomas Ngo and Joe Marks. Spacetime constraints revisited. *ACM Computer Graphics Proceedings*, pages 343–350, 1993.

[11] W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling. *Numerical Recipes in C — The Art of Scientific Computing*. Cambridge University Press, 1989.

[12] Craig W. Reynolds. Flocks, herds and schools. *ACM Computer Graphics*, 21(4):25–34, July 1987.

[13] W. M. Rohsenow and H. Choi. *Heat, Mass, and Momentum Transfer*. Prentice-Hall, 1961.

[14] Karl Sims. Evolving virtual creatures. In *SIGGRAPH'94*, pages 15–22. ACM, 1994.

[15] X. Tu and D. Terzopoulos. Artificial fishes: Physics, locomotion, perception, behavior. In *SIGGRAPH'94*, pages 43–50. ACM, 1994.

[16] Michiel van de Panne and Eugene Fiume. Sensor-actuator networks. *ACM Computer Graphics*, pages 335–342, 1993.

[17] Michiel van de Panne, Ryan Kim, and Eugene Fiume. Virtual wind-up toys for animation. In *Proceedings of the Graphics Interface'94*, pages 208–215, 1993.

[18] S. Ward. Chemotaxis by the nematode *caenorhabditis elegan*: identification of attractants and analysis of the response by use mutants. In *Proceedings of the National Academy of Sciences (U.S.A.)*, pages 70:817–821, 1973.

[19] Josie Wernecke. *The Inventor Mentor*. Addison-Wesley Publishing Company, 1994.

[20] Jane Wilhelms and Robert Skinner. A notion for interactive behavioral animation control. *IEEE Computer Graphics and Applications*, 10(3):14–22, May 1990.