3

Simulating Life of Virtual Plants, Fishes and Butterflies

Hansrudi Noser, Daniel Thalmann
Computer Graphics Lab, Swiss Federal Institute of Technology
CH-1015 Lausanne, Switzerland

3.1. Introduction

In this chapter, we present a general methodology for simulating the life and behavior of creatures and their reaction to their environment. In a larger sense, we can attribute a behavior to each object. Dead objects, like balls, stones, ... move (behave) in their environment according to physical laws. Plants, as all other living creatures, obey also Newton's laws (gravitation, wind, ..), but besides, they grow, reproduce themselves and react to their environment (light, temperature, ...). The behavior of animals is in addition influenced by emotional and instinctive components. Males and females are attracted by each other, or as an other example, a predator is attracted by his victim, which on the other hand is repelled by the predator. For most of the humans and a lot of animals their behavior is strongly determined by their vision, which is a very important channel, by which the environment is perceived. Finally, the highest level of behavior includes intelligence, which is only attributed to humans.

Our approach is based on L-systems, force fields and synthetic vision. With this approach, high level physical and behavioral animation is possible. Vision based actors find their way without collisions in a L-system environment to given destinations. Dynamically created objects, moving in complex 3D vector force fields can interact with each other and branched structures, simulating behavior of herds, flocks, schools or some physical effects.

Production systems and L-grammars, as introduced by Prusinkiewicz and Lindenmayer (1990) are very powerful tools for creating images. From a user-defined axiom and a set of production rules, the computer creates images with a complexity dependent only on the number of times the productions are applied. The theory of L-systems has been mainly used for the visualization of the development and growth of living organisms like plants, trees and cells. In a previous paper (Noser et al. 1992), we presented the software package LMobject which realizes a timed and parametric L-system with conditional and pseudo stochastic productions for animation purposes. With this software package a user may create realistic or abstract shapes, play with various tree structures and generate a variety of concepts of growth and life development in the resulting animation. To extend the possibilities for more realism in the pictures, we added external forces, which interact with the L-structures and allow a certain physical modeling. External forces can also have an important impact in the evolution of objects.

Prusinkiewicz and Lindenmayer (1990) have proposed two simple cases of external forces. In the first method, the 3D turtle which interprets the symbolic grammar may be aligned horizontal to a vector representing the gravity. Thus, an object (a tree, for example) is able to "feel" the gravity and to react. The second case is specific to plant and tree modeling and allows the simulation of tropism, which is responsible for the bending of branches towards light sources.

In our extended version, tree structures can be deformed in an elastic way and animated by time and place dependent vector force fields. The elasticity of each articulation can be set individually by productions. So, the bending of branches can be made dependent on the branches' thickness, making animation more realistic. The force fields too, can be set and modified with productions. This kind of interaction is based on the principle of tropism as described by Prusinkiewicz and Lindenmayer (1990).

Further, we introduced a third type of force interaction, that affects L-structures. This simulates the displacement of objects in any vector force field dependent on time and position. An object's movement is determined by a class of differential equations Eq.(3.1), which can be set and modified by productions. The mass of the turtle, which represents the object, can be set as well by using a special symbol of the grammar.

$$\begin{aligned} \ddot{x} &= \frac{1}{m} f_x(t, x, y, z, \dot{x}, \dot{y}, \dot{z}, X, Y, Z) \\ \ddot{y} &= \frac{1}{m} f_y(t, x, y, z, \dot{x}, \dot{y}, \dot{z}, X, Y, Z) \\ \ddot{z} &= \frac{1}{m} f_z(t, x, y, z, \dot{x}, \dot{y}, \dot{z}, X, Y, Z) \end{aligned} \tag{3.1}$$

To solve the differential equation system, we evolve an initial value problem for Eq.(3.1) using the 4th-order Runge Kutta Method. This vector force field approach is particularly convenient for simulating the motion of objects in fluids (air, water) as described by Wejchert and Haumann (1991).

Behavioral animation was studied in detail by Reynolds (1987). He gives a good overview of different methods and its problematic. Susan Amkraut and Michael Girard showed in the film "Eurhythmy" (Girard and Amkraut 1990) a flock of birds flying around and avoiding collisions between themselves and obstacles in their environment using a force field animation system to realize the simulation. Repulsion forces around each bird and around static objects are responsible for collision avoiding. At the beginning of the animation, the space field and the initial positions, orientations, and velocities of objects are defined and the rest of the simulation is evolved from these initial conditions.

The force field approach in behavioral animation is well suited for modeling an instinct driven, animal behavior for a large number of actors forming schools, herds or flocks. The behavior of intelligent actors, however, needs more sophisticated techniques. To simulate intelligent or human behavior in path searching and obstacle avoidance in a synthetic environment, we have developed a synthetic vision based global navigation module (Renault et al. 1990; Noser et al. 1994). The task of a navigation system is to plan a path to a specific goal and to execute this plan, modifying it as necessary to avoid unexpected obstacles (Crowley 1985). With the use of synthetic vision we simulate the way a real human perceives the environment. Thus, the first step in the simulation of a natural and intelligent behavior is done. Moreover, in a L-system defined environment, where there is no 3D geometric database of the environment because the world exists only after the execution of production rules, synthetic vision gives an elegant and fast way to provide information about the environment to the actor.

In the L-system based animation system, described in (Noser et al. 1992), only a global force field determining the system of differential equations of all objects could be defined in the axiom or in the production rules. Thus, only one movement type could be defined at a given time. Collision detection and object interaction were not possible. In our new approach, each generated object has its own force field acting on other objects or branching structures. This extension allows collision detection and behavioral animation. In addition, each object has its own differential equation, which determines his movement in the global force field given by the contributions of all other objects. So, at the same time several types of movements are possible (falling apple , jumping ball, school of fish, ...).

In this chapter, we also present the implementation of synthetic vision into the L-system based animation software, where the behavior of an actor is at a higher level determined by an automata, which allows an actor to find a path from his current position to a given destination in known or unknown environments by avoiding collisions. Several actors can be generated and personalized in the axiom or in a production rule. More details about our synthetic vision approach may be found in another paper (Noser et al. 1994), including the visual memory representation by an octree and the path searching algorithms (3D), working with heuristics, used to personalize an actor's behavior. In Section 3.2 we present the formal definition of our L-system, explained in (Noser et al. 1992). Section 3.3 and 3.4 describe some theoretical aspects of the new extensions. In Section 3.5 some implementation details are given, as well as the semantic of the symbols 'C' and 'M' of the formal grammar which represent the behavioral features of force field (symbol 'C') and vision based (symbol 'M') animation. As part of turtle interpreted L-systems (or LOGO), these symbols can be considered as high level turtle operation symbols, in contrast to lower level, standard symbols like 'f' (forward) or '+' (rotate), which only advance the turtle or rotate it around an axis by given values. Our new approach allows, for example, an easy animation of flocks of butterflies, of schools of fishes or of path searching actors in a L-system modeled environment. Simple physical simulations, like explosions or bouncing balls are also feasible.

3.2. The L-system

In this section, we introduce the formal description of our L-system; it defines how an object at a certain age is derived by the derivation function D from an initial word (= axiom = sequence of parametric symbols) and the production rules. The symbols of the alphabet are interpreted by means of a 3D turtle, whose current orientation is given by a local orthonormal coordinate system with the axis H(heading), L(left) and U(up). Some symbols control the position and the orientation of the turtle's coordinate system, others represent geometrical primitives, drawn in the turtle's coordinate system, and others perform special operations. For readers not familiar with L-systems we highly recommend the lecture of "The Algorithmic Beauty of Plants" (Prusinkiewicz and Lindenmayer 1990) where different types of L-systems are well introduced. More details about our implementation and extension may be found in (Noser et al. 1992).

We start the formal description of the L-system by giving some elementary definitions.

V:	an alphabet
R^+:	the set real positive numbers
N :	the set integer positive numbers
Σ :	a set of formal parameters
$E(\Sigma)$:	the arithmetic expressions in Polish notation
$C(\Sigma)$:	the locical expressions in Polish notation
ω :	an axiom
P :	$P = \{ p_{a,i} \| a \quad V, i \quad N \}$ the set of productions
Π:	$p_{a,i} \rightarrow [0,1]$ a pseudo -statistical distribution with $\sum_i \pi(p_{a,i}) = 1$
$(a, x_o, \ldots, x_6) \quad V \times R^7$ a parametric symbol	

The productions of an L-system describe how symbols of a timed and parametric string are replaced if they pass their maximal age during evolution of time.

$P \quad V \times R^7 \times C(\Sigma) \times \Pi \times (V \times E(\Sigma)^7)^*$: the production space

$p_{a,i} \quad P$: a single production

$$p_{a,i}: (a, \alpha, x_1, \ldots, x_6) \xrightarrow{Cond(\alpha, x_1, x_2, x_3, T) \text{ and } \Pi(p_{a,i})} (a_1, f_{10}, \ldots, f_{16}) \ldots\ldots (a_n, f_{n0}, \ldots, f_{n6})$$

α:	maximal age of symbol a
f_{j0}	initial age of symbol a_j
x_1, x_2, x_3	3 parameters
x_4, x_5, x_6	values of the evaluated growth functions
$f_{j0} + t$	local age of the symbol
T	global time
$f_{j1}, \ldots, f_{j3}$	parameter expressions
$f_{j4}, \ldots, f_{j6}$	growth functions
f_{jk}	$f_{jk}(f_{j0}, x_1, x_2, x_3, t, T)$
$p_{a,i}$	a conditional and pseudo -statistical production

In the above definition the symbol a is replaced with a certain probability under a given condition by a parametric string given by the right part of the production. The way, how this replacement is done, is controlled by the derivation function D and its mathematical axioms.

D: $\left(V \times R^7\right)^* \times R \rightarrow \left(V \times R^7\right)^*$

Axiom 1: The development of each symbol is independent of each other in tim

$$D\left(\left(a_1,x_0,\ldots,x_6\right)\ldots\left(a_n,x_0,\ldots,x_6\right),t\right) = D\left(\left(a_1,x_0,\ldots,x_6\right),t\right)\ldots D\left(\left(a_n,x_0,\ldots,x_6\right),t\right)$$

Axiom 2: The Growth of a symbol before terminal age

if $x_0 + t < \alpha$ then

$$D\left(\left(a,x_0,\ldots,x_6\right),t\right) = \left(a,x_0+t,x_1,x_2,x_3,f_4,f_5,f_6\right)$$

The growth functions are evaluated

Axiom 3: Application of stochastic production at terminal age

if $(x_0 + t \geq \alpha)$ $\left(Cond\left(\alpha,x_1,x_2,x_3,T\right) = TRUE\right)$ then with a probability $\pi\left(p_{a,i}\right)$

$$D\left(\left(a,x_0,\ldots,x_6\right),t\right) = D\left(\left(a_1,f_{10},\ldots,f_{13},x_4,x_5,x_6\right)\ldots\left(a_n,f_{n0},\ldots,f_{n3},x_4,x_5,x_6\right),t-\left(\alpha-x_0\right)\right)$$

The initial age and parameter expressions $f_{10},\ldots,f_{13}$ are evaluated.

Axiom 4: Selection of random numbers for the productions

$$random\_number = rand\_table\left[\left(x+y[x]\right) \bmod z\right]$$

rand_table:	table of size z with uniform random values between 0 and 1
x	: recursion depth of function D
y [x]	: position of the production in the derivation tree of D at the x depth

To guarantee under certain conditions a continuous growth of the plants the axiom 4 has to be added (Noser et al. 1992) if stochastic productions are used in the plant definition by production rules.

3.3. Behavioral Modeling using Force Fields

In a "force field animation system" the 3D world has to be modeled by force fields. Some objects have to carry repulsion forces, if there should not be any collision with them. Other objects can be attractive to others. Many objects are both attractive at long distances and repulsive at short distances. The shapes and sizes of these objects can vary, too. Space fields like gravity or wind force fields can greatly influence animation sequences or shapes of trees.

The system of differential equations, given in Eq.(3.2), describes the movement of a point object *i* with mass m_i in a force field. The global force field is given by the sum of all objects' contributions f_{kj}. The individual part g_{ki} of each object determines it's behavior in the global field,

$$\ddot{x}_i = \frac{1}{m_i}\left(g_{xi}(x_i,y_i,z_i,\dot{x}_i,\dot{y}_i,\dot{z}_i,t) + \sum_{j,j\neq i} f_{xj}(x_i,y_i,z_i,x_j,y_j,z_j,\dot{x}_j,\dot{y}_j,\dot{z}_j,r_{ij},t)\right)$$

$$\ddot{y}_i = \frac{1}{m_i}\left(g_{yi}(x_i,y_i,z_i,\dot{x}_i,\dot{y}_i,\dot{z}_i,t) + \sum_{j,j\neq i} f_{yj}(x_i,y_i,z_i,x_j,y_j,z_j,\dot{x}_j,\dot{y}_j,\dot{z}_j,r_{ij},t)\right)$$

$$\ddot{z}_i = \frac{1}{m_i}\left(g_{zi}(x_i, y_i, z_i, \dot{x}_i, \dot{y}_i, \dot{z}_i, t) + \sum_{j, j \neq i} f_{zj}(x_i, y_i, z_i, x_j, y_j, z_j, \dot{x}_j, \dot{y}_j, \dot{z}_j, r_{ij}, t)\right) \qquad (3.2)$$

where :

i	= index of an object
m_i	= the mass of object i
x_i, y_i, z_i	= position components of object i
$\dot{x}_i, \dot{y}_i, \dot{z}_i$	= velocity components of object i
$\ddot{x}_i, \ddot{y}_i, \ddot{z}_i$	= acceleration components of object i
t	= time
r_{ij}	= distance between object i and object j

The behavior of an object in the global force field is determined by a predefined curve (e.g. spline, fixed) or it's individual part *g* of Eq.(3.2). The terms of *g* can depend on the object's actual position and it's speed. Speed dependent terms can be used to model friction properties. The position variables allow it to make the object's behavior position dependent.

Tropism forces act on the articulations of branching structures (Prusinkiewicz and Lindenmayer 1990). The bending of branches is simulated by a rotation of the turtle in direction of the tropism forces. If our dynamically created objects have to interact with branching structures, then their force fields have to be added to the tropism force. The following equations describe this interaction in detail.

$\vec{M} = \left(\vec{H} \times \vec{F}\right) / e$	torque vector(e =elasticity of articulation)	
$m = \left\|\vec{M}\right\|$	torque	
$\vec{A} = \left(\vec{H} \times \vec{F}\right) / \left\|\left(\vec{H} \times \vec{F}\right)\right\|$	rotation vector	
$\beta = \sin^{-1}\left(\left\|\vec{A}\right\|\right)$	angle between turtle heading $\vec{H}$ et $\vec{F}$	(3.3)
$\alpha = \beta\left(1 - \frac{1}{1+m}\right)$	resulting rotation angle	
$\vec{F} = \vec{f}_{tropisme} + \sum_i \vec{f}_i$	global force field	

The effective rotation angle is proportional to the torque m, produced by the global force by acting on the turtle's heading vector, but it never exceeds the angle between the turtle heading and the force vector at the turtle's position.

3.4. Vision Based Animation

In our approach to synthetic vision (Noser et al. 1994), a dynamic occupancy octree grid serves as a global 3D visual memory and allows an actor to memorize the environment that he sees and to adapt it to a changing and dynamic environment. His reasoning process allows him to find 3D paths based on his visual memory by avoiding impasses and circuits. The global behavior of the actor is based on the navigation automata, shown in Figure 3.1, representing the automata of an actor, who has to go from his current position to different places, memorized in a list of destinations. He can displace himself in known or unknown environments. After the initialization of his memory and his vision system (off -> mem_set -> vision_set) and the definition of his destinations (end_point_set) the path searching is initialized. Before starting a search, he looks around and memorizes what he sees (look_around). Then, he tries to find a path by reasoning (search_new_path). If he finds one, he follows it (move). If he encounters an obstacle while moving, he stops, turns around, looks for a new path and restarts moving. This process is repeated until he arrives at his destination. There, he takes the next destination from the list and restarts the whole procedure described above.

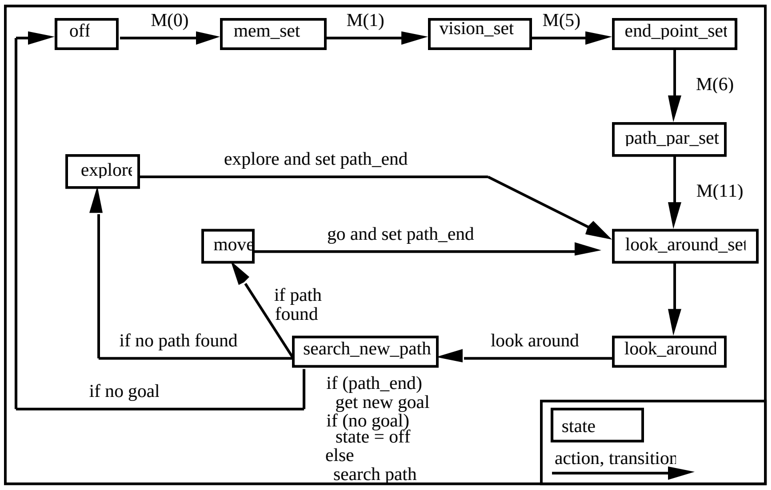

Figure 3.1. Global navigation automata of the actor.

In unknown environments, if he uses a conditional heuristic, he cannot find a path to an invisible destination (behind an obstacle, for example). In this case, he starts to explore his environment according to his heuristic, which will lead him to his destination, if there is any path.

3.5. Implementation of the Behavioral Features

We integrated the physical modeling and behavioral animation features in the LMobject software package (Noser et al. 1992). This L-system is a timed parametric context-free grammar with conditional and pseudo-stochastic productions. All grammar symbols have three parameters and three growth functions. It should be noted, that neither the formal object, nor the graphical object need to be stored completely in memory. Only the formal description of the current state of the derivation function needs to be maintained on a stack, resulting in memory requirements proportional to the recursion depth of the derivation function. Each time the derivation functions encounters a leaf of the derivation tree, it interprets the symbol and executes immediately the corresponding action. This action can be, for example, a turtle operation, a force field declaration, a camera manipulation or the drawing of a graphical primitive in the turtle's coordinate system. The actions are determined by the semantic of the grammar.

The new extension allows us, to dynamically create new point objects with a special symbol **C** of the grammar. Each time the derivation function of the L-system which derives the object from the axiom and the production rules encounters this symbol **C**, it acts on a global interaction table containing the trace of all generated objects. This interaction table has the following structure:

```
typedef struct {
        short   id_depth, id_large;
        float   posVel[6];
        char    *force[3], *eqDiff[3], *type;
        float   mass
}Tinteraction;
Tinteraction InteractionTable[MAX_OBJECTS]
```

The recursive derivation function identifies with its fourth axiom each object in the derivation tree and returns the unique object identification *id_depth* and *id_large.*

If the object is encountered for the first time, it is appended to the global interaction table. The initial position (turtle) and the speed (from the symbol C) are copied into the table posVel and the force (*force) , the differential equation (*eqDiff(3)) and the type (*type) pointers are directed to the corresponding character expressions in the parametric symbol C. This way, the mass of the object can be set as well.

If the object already exists in the interaction table, the derivation function starts an iteration step of the numerical solution of the object's differential equation by regarding all the contributions of the other force fields of the same type in the global interaction table. Then, the turtle is placed at the resulting position. The iteration step is calculated by the method of Runge Kutta of degree four and solves the system of differential equations given in Eq.(3.2). This system describes the Newtonian movement of a point with a mass m in the 3D space under the influence of a 3D vector force field. As the position and speed of each created object are stored in the global interaction table, the relative distances r_{ij} from one object i to an other object j, can be calculated and thus used in the expressions of the force fields and differential equations. The semantic of the parametric symbol **C**

C (t) (x) (y) (z) $(f1)$ $(f2)$ $(f3)$

depends on the parameter x. If $x = 1$, the expressions $f1$, $f2$ and $f3$ define the three components of the vector force field of the current object. If $x = 2$, the current object's differential equation is determined in the same way. If $x = 3$, the value of y sets the mass, and the object's movement is started with an initial speed given by the evaluated expressions $f1$, $f2$ and $f3$.

Sometimes it is useful, that an object carrying a force field moves along a predefined path. If $x = 4$, the differential equation of the current object is not evaluated, but the object is placed at the position given by $(f1, f2, f3)$. The fi's, for example, can be timed spline defining a 3D path.

The integration of the vision module into the L-system is realized via the special symbol **M** of the grammar. The semantic of **M** depends also on it's parameter x. In Figure 3.1, we can see its influence on the state of the actor.

With $x = 0$, the symbol

M (t) (0) (y) (z) $(f1)$ $(f2)$ $(f3)$

initializes and scales the visual octree memory of an actor. The values of $f1$ and $f2$ determine the cube including the whole scene and $f3$ sets the resolution of the octree (maximal tree depth).

M (t) (1) (y) (z) $(f1)$ $(f2)$ $(f3)$

permits to personalize the actors vision system. With $f1$ and $f2$ the near and far clipping is set, and $f3$ gives the range of interest within which the actor adapts his memory to changes in the environment.

With the symbol

M (t) (6) (y) (z) $(f1)$ $(f2)$ $(f3)$

the path searching and exploring procedure can be personalized. The value of $f1$ gives the distance, the actor previews along his actual found path, to detect collisions with his environment. By $f2$ the size of voxels to be examined during path searching and exploring is set. With $f3$, the user can determine one of the predefined heuristics to use in the path searching or exploring algorithm. It is also possible to set initial positions and look at points, destinations and moving speeds of the actors by using the corresponding parametric symbol **M**.

So, a user can define and personalize one or several actors in the axiom. He can give him several destinations, and then the animation will evolve according to the global navigation automata shown in Figure 3.1. We also implemented the ability to place and orient the camera according to an object's position and velocity so that nice animation from the perspective of an actor with vision or a force field influenced object can be realized.

3.6. Applications

3.6.1. Growing Tree Structure

In this section we illustrate the working principle of the derivation function D by deriving the formal object of a continuously growing simple tree structure at several ages. We consider the alphabet V = {**p**, **q**, **+**, **-**, **[**, **]** } with all characters printed in bold. **p** and **q** are timed and parametric symbols with growth functions. To establish the link with the formal description of Section 3.2 we use the following notation.

(**p**, a, x, f(x, t))	the parametric symbol
p	the symbol
a	the parameter x0 which is the actual or initial age of the symbol.
x	the parameter x1
f(x,t)	the growth function f_4
t	the time

The symbols **p** and **q** represent line segments with a length of the actual value of the growth function. The line segment is drawn from the actual position of the turtle in direction of its heading vector H. After having drawn the line segment the turtle is placed at its end without changing its direction. The symbols **+** and **-** of the alphabet V represent rotations of the turtle around its U vector by an angle of +-90 degrees. The brackets **[** and **]** correspond to push and pop operations of the turtle state (position and orientation) which are introduced to delimit branches. Thus, the brackets enable the construction of tree structures.

The tree structure is defined by an axiom and one production for the symbol **p**.

Axiom: (**p**, a=0, x=1, f=6xt)
Production: (**p**, α = 1) ------------> (Condition = TRUE, probability = 1)
(**q**, a=0, x=x, f=6x+t) **[+** (**p**, a=0, x = x/2, f=6xt) **] [-** (**p**, a=0, x=x/2, f=6xt) **]**

The axiom corresponds to a line segment (symbol **p**) with initial age a=0, a parameter x=1 and a linear growth function f = 6xt = 6t (as x =1). So, the segment will linearly grow until it reaches its maximal age α = 1 given at the left side of the production. The production is applied with a probability of 1.0 without any precondition. Figure 3.2 illustrates some applications of the production

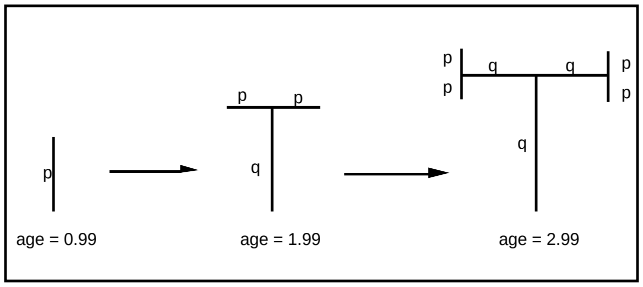

Figure 3.2. Continuous growth of a simple tree.

The line segment p at maximal age of 1 has to be replaced by a segment q of the same length at its initial age of a=0. So, the parameter x of the symbol p (which is replaced) is passed without any change. The growth function fq=6x+t satisfies this continuity condition as f_p(t=1, x=1) = f_q(t=0, x=1) = 6. The term t is responsible for a further linear but smaller growth rate of the segment q. So each branch of the segment continues to grow. The two new branches p should be a factor 2 shorter then the preceding one. Therefore, the parameter x is passed to the new symbols by dividing it by 2 (x = x/2). As they start their growth from zero the same growth function has to be used as for the symbol p in the axiom. Thus, a continuous and consistent further replacement is guaranteed during the following iterations.

In the next paragraph, we provide some details about derivation steps of the formal object. We show how the derivation function D works and when the corresponding axioms of the derivation function D are applied. To avoid some confusion the reader is asked to distinguish between the enumerated mathematical axioms of the derivation function D and the "string" axiom of the tree definition.

Object age t = 0.5:

D(Axiom, 0.5)= D((**p**, 0, 1, 6xt), 0.5) =

(Axiom 2, as a+t = 0.5 < α=1, the growth function is evaluated)

(**p**, a+t, x, 6xt) =

(**p**, 0+0.5, 1, 6*1*0.5) =

(**p**, 0.5, 1, 3)

Object age t = 0.9

D(Axiom, 0.9) = ... = (**p**, 0.9, 1, 5.4)

Object age t = 1.5

D(Axiom, 1.5) = D((**p**, 0, 1, 6xt), 1.5) =

(Axiom 3, as a+1.5 = 0+1.5 = 1.5 > α = 1,
the parameter x from the preceding symbol p is passed to the symbols of the production by evaluating the corresponding expressions)

D((**q**, 0, x, 6x+t) **[** + (**p**, 0, x/2, 6xt) **]** **[** - (**p**, 0, x/2, 6xt) **]**,1.5-(α-a)) =

D((**q**, 0, 1, 6x+t) **[** + (**p**, 0, 0.5, 6xt) **]** **[** - (**p**, 0, 0.5, 6xt) **]**, 0.5) =

(Axiom 1)

D((**q**, 0, 1, tx+t), 0.5) **[**+D((**p**, 0, 0.5, 6xt), 0.5) **][**- D((**p**, 0, 0.5, 6xt), 0.5)**]** =

(Axiom 2, as a+t = 0+0.5 < α=1,
the growth functions are evaluated by using the evaluated parameter x)

(**q**, 0.5, 1, 6.5) **[** + (**p**, 0.5, 0.5, 1.5) **]** **[** - (**p**, 0.5, 0.5, 1.5) **]**

3.6.2. School of Fishes

In the following animation fishes with mass m=1 are generated at the same place at a regular rate. Their repulsive force field is given by Eq.(3.4). All these fishes are attracted by a moving object, the bait, with an attractive force field at long distances and a repulsive one at short distances (Eq.(3.5)). The movement of each fish is damped by Eq.(3.6). Figure 3.3 illustrates the bait's force field.

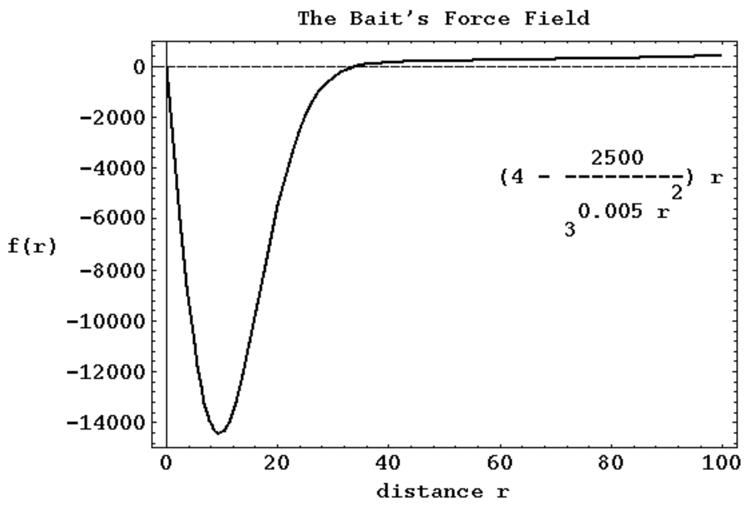

Figure 3.3. The bait's force field.

The fish's force field has about the same form at short distances. At long distances however, it remains zero.

$$\vec{f}_{fish}(r) = -2500 \;\; 3^{-0.005r^2} \;\; r \;\; \vec{n}$$

$$\text{with } \;\vec{r} = (\vec{X}_{object} - \vec{x}), \;\; r = \left| \vec{X}_{object} - \vec{x} \right| \quad \text{and } \vec{n} = \frac{\vec{r}}{r}$$

$\vec{X}_{object}$: Position of the object (fish) (3.4)

$\vec{x}$: position in the the force field

r : distance from the force field center of the object

$$\vec{f}_{bait}(r) = (4 - 2500 \;\; 3^{-0.005r^2}) \;\; r \;\; \vec{n} \tag{3.5}$$

$$\vec{g}_{fish}\left(\dot{\vec{x}}\right) = -3\dot{\vec{x}} \tag{3.6}$$

The bait moves along a given 3D path spline. Each new generated fish joins immediately the school, following the bait. The strongly repulsive components of the force fields at short distances of all moving objects, prevent collisions. In Figure 3.4 (see Color Section), we can see some pictures from the whole animation sequence.

Figure 3.5 shows parts of the axiom and the production rules used for the above animation sequence.

```
Axiom
...... /* some symbols, describing the environment It follows the definition of the bait */
f ()  () () ()          (0) (0) (0)
     /* the symbol f places the turtle at (0,0,0) and determines the initial bait position*/
C ()        (1) () ()
     ((4-2500*3^(-0.005r^2) * (X-x))
     ((4-2500*3^(-0.005r^2) * (Y-y))
     ((4-2500*3^(-0.005r^2) * (Z-z))
     /* Symbol C with parameter x=1 defines the bait's force field. (X,Y,Z) are the coordinates of the current
     object (in this case the bait). (x,y,z) is the position of an object, feeling the force field. The variable r is
     given by r=sqrt((X-x)^2 + (Y-y)^2 + (Z-z)^2), the distance of a fish to the bait. */
C ()        (4) () ()  (spline_1(t)) (spline_2(t)) (spline_3(t))
     /* Symbol C with parameter x=4 places the turtle (= the bait) at the position (spline_1(t),  spline_2(t),
     (spline_3(t)) determined by a predefined timed 3D spline */
z ()() () ()                    () () ()
     /* the symbol z is a germ for the bait. It is an arbitrary geometrical figure, defined by the corresponding
     production rule, not given here. It is drawn in the turtle coordinate system */
x ()        () () ()            () () ()
     /* The symbol x is a germ for a fish force field declaration */
...... /* some more symbols, describing something arbitrary */
EndAxiom

Production1
x (maximal_age = 1) --------------------->/*symbol x to be replaced by the following symbols*/
     /* it follow the parametric symbols, defining the behavior of a fish */
f () () () ()                   (50) (50) (50)
     /* the symbol f places the turtle (=fish) at the position (50, 50, 50) and thus determines the initial position of
     a fish */
C ()        (1) () ()
     ((-2500*3^(-0.005r^2) * (X-x))
     ((-2500*3^(-0.005r^2) * (Y-y))
     ((-2500*3^(-0.005r^2) * (Z-z))
     /* Symbol C with parameter x=1 defines the fish's force field, felt by other objects. */
C ()        (2) () ()  (-3u) (-3v) (-3w)
     /* Symbol C with parameter x=1 defines the individual part of each generated fish of the Eq.(2). The vector
     (u,v,w) represents the velocity of the current fish. */
C ()        (3) (1) ()(5) (3) (2)
     /* Symbol C with parameter x=3 starts the evolution of the generated fish with the initial velocity (5, 3, 2)
     and the mass 1 */
y (1)       () () ()               () () ()
     /* Symbol y represents a germ of a fish with an initial age of 1 */
x ()        () () ()               () () ()
     /* The symbol x is a germ for a fish force field declaration */
EndProduction1
```

Figure 3.5. Parts of axioms and production rules for the fish animation.

This textual description of the axiom and the production are interpreted and controlled by the derivation function D of the L-system. In this case, at each time interval of 1 (1 = maximal symbol age of a fish germ x) a new fish is generated and immediately starts it's typical behavior. It follows the bait and avoids collisions.

3.6.3. Tree-Ball Interaction

In this example, we show the interaction of a ball and a tree. The ball moves towards a tree. As it carries a repulsive "velocity" force field, the tree and the branches bend under it's influence and try to avoid collision with the ball. The ball's force field is shown in Eq.(3.7)

$$\vec{f}_{ball}\left(r, \dot{\vec{x}}\right) = (18\ 3^{-0.005r^2})\ \dot{\vec{x}} \qquad (3.7)$$

Here, the force field is always in the direction of the ball's velocity, even behind the ball, so wind, caused by the movement of the ball is simulated. Figure 3.6 (see Color Section) shows some pictures of the animation sequence.

3.6.4. A Butterfly in a Flower Field

In the animation sequence, illustrated in Figure 3.7 (see Color Section), a butterfly searches his way through a flower field, guided by his vision. It is an example of the use of 3D heuristic search (Noser et al. 1994). The flowers are animated by a wind force field. The butterfly is modeled by some symbols of the grammar just as the flying motor.

The first destination of the butterfly is the reflecting sphere. When it arrives there, it looks around and searches for a path to the second destination at the other end of the flower field. Guided by its vision, it is able to avoid collisions and to find a path, as well as to memorize in its visual memory everything it sees.

3.7. Conclusion

With our current L-system based animation system, simulations in all of the above mentioned domains can be realized and combined in one sequence. As Eq. (3.2) is based on Newton's laws, physical simulations are immediate. Growing and reproduction features are inherent in the L-system. We can realize, for example, animation sequences of a growing tree, producing apples, which contain germs for new trees. These germs develop to new trees, after the apples have fallen to the ground. Thus, a whole forest can develop from a single apple (Noser et al. 1992) Beside the physical modeling, the force field approach allows as well to simulate instinctive or emotional behavior (attractive, repellent forces) of individuals or groups and to interact physically with plants and dead objects. Animation according to the following description are possible:

"A small lake is populated with plants, predators and a fish. The fish spawns. The spawn floats away in a water current. After some time the spawn develops to fish which join to a school following a guide. They avoid the predators. "

The vision based features, allow the actors to avoid collisions and to simulate intelligent behavior. The global navigation automata represents 'intelligent' behavior. It allows an actor to find, for example, the exit of a maze, even if there are impasses and circuits, and to memorize the topology of the seen environment. This learned information he can use in future path searching.

In a future development, we plan to improve the force field animation module. Since most objects' force fields have only short distance ranges, it would save much calculation time if only the force field contributions of nearby objects had to be considered. To solve this problem, we plan to introduce a dynamic space subdivision with octrees, in which the force fields can be placed. Thus, neighboring objects could easily be identified. With such an approach much more complex animation would be feasible at still reasonable calculation times.

The synthetic vision module represents a very universal and powerful tool for future behavioral models. By designing and combining new automata and including more sophisticated actors with walking, speaking and grasping features, high level script based animation can be realized in a extremely dynamic environment with partially autonomous actors.

Acknowledgements

This research was sponsored by the Swiss National Foundation for Scientific Research.

References

Crowley JL, Navigation for an Intelligent Mobile Robot (1985) *IEEE Journal of Robotics and Automation*, Vol. RA 1, No 1, pp31-41

Girard M, Amkraut S (1990) Eurhythmy: Concept and Process, *Journal of Visualization and Computer Animation*, Vol. 1, pp.15-17

Noser H, Renault O, Thalmann D, Magnenat Thalmann D, Vision-Based Navigation for Synthetic Actors, (Submitted for publication, 1994)

Noser H, Thalmann D, Turner R (1992) Animation based on the Interaction of L-systems with Vector Force Fields, *Proc. CGI '92*, Springer, Tokyo

Prusinkiewicz P, Lindenmayer A (1990), *The Algorithmic Beauty of Plants*, Springer-Verlag

Renault O, Magnenat Thalmann N, Thalmann D (1990), A Vision-based Approach to Behavioral Animation, *Journal of Visualization and Computer Animation*, Vol 1, No 1, pp 18-21

Reynolds C (1987), Flocks, Herds, and Schools: A Distributed Behavioral Model,*Proc. SIGGRAPH 1987, Computer Graphics*, Vol.21, No4, pp.25-34

Wejchert J, Haumann D (1991), Animation Aerodynamics, *Proc. SIGGRAPH 1991, Computer Graphics,* Vol.25, No4, pp. 19-2.