ARTICLE IN PRESS

Simulation Modelling Practice and Theory xxx (2008) xxx–xxx

Contents lists available at ScienceDirect

Simulation Modelling Practice and Theory

journal homepage: www.elsevier.com/locate/simpat

Beyond particle systems: Operator networks

Mauno Rönkkö [a,*], Marina Waldén [b], Ralph-Johan Back [b]

[a] Department of Computer Science, University of Kuopio, P.O. Box 1627, 70211 Kuopio, Finland
[b] Department of Information Technologies, Åbo Akademi University, Joukahainengatan 3-5A, 20520 Åbo, Finland

ARTICLE INFO

Article history:
Received 18 January 2008
Received in revised form 5 May 2008
Accepted 8 May 2008
Available online xxxx

Keywords:
Particle systems
Operator networks
Non-linear dynamics
Complex systems

ABSTRACT

Particle systems are used for simulating non-linear dynamics of complex systems. They are computationally attractive, because the models are simple difference equations. The difference equations, however, constitute a closed system lacking scalability and intentionality; it is hard to "reverse engineer" the equations, to understand the relations of the variables and coefficients to the dynamics displayed by the simulation. Consequently, much of the modeling work goes into finding workarounds. In this paper, we study a potential solution. As the main contribution, we formalize particle system computations as mathematical operator networks, to gain intentionality and modularity. Operators also support the inclusion of processes outside the mathematical domain of difference equations. We illustrate the use of operator networks by simulating the construction and dynamics of an hourglass.

1. Introduction

Particle systems are used for simulating non-linear dynamics of complex systems that are hard or impossible to capture by using any other means. Particle systems are typically used for simulating fluids and gasses [4,21,22]. Recently, particle systems have also been used for simulating non-linear dynamics of non-rigid bodies [3,5,7,12,16], and motion of biological entities [6,16]. There have even been attempts [15] to simulate the entire spectrum of dynamics arising in an artificial ecosystem using particle systems only.

Particle based modeling is a complementary method to other existing methods, such as bond graphs [2] and FFT based methods [19]. Particle systems are also closely related to finite element methods [8,18] and spring-mass models [23].

Particle systems are found computationally attractive, because the models are mere difference equations over matrices [15,22], independently of the underlying complexity of the simulated object. Consequently, the models are compact and computed easily. Furthermore, the simulations often exhibit emergent dynamics [9,15], delivering a level of realism [12] that is otherwise hard or impossible to capture. By emergent dynamics, we mean here dynamics of the whole that is not evident or predictable from the dynamics of the components. More specifically, particle systems reveal basic emergence [9], which means behavior that is reducible to particle-to-particle interactions without any evolutionary processes involved. The displayed emergent dynamics varies from model to model, and may be anything from simple friction [5] to complex hunting behavior [15]. Because of this, when simulating with particle systems, the goal is to develop as simple model as possible, where the desired properties emerge. In this way, the model not only gains more predictive power, but also delivers a higher level of realism. The use of difference equations as the model is, however, also the major source of problems when modeling with particle systems; the difference equations constitute an inherently closed system. This manifests in many ways. The models are hardly reusable either in parts or as a whole. Also, a significant amount of work time is invested in finding appropriate variable values for the equations as well as in finding workarounds to capture specifics and exceptions. Most

* Corresponding author. Tel.: +358 17 163 793; fax: +358 17 162 595.
E-mail address: mauno.ronkko@uku.fi (M. Rönkkö).

1569-190X/$ - see front matter © 2008 Elsevier B.V. All rights reserved.
doi:10.1016/j.simpat.2008.05.003

Please cite this article in press as: M. Rönkkö et al., Beyond particle systems: Operator networks, Simulat. Modell. Pract. (2008), doi:10.1016/j.simpat.2008.05.003

importantly, however, the difference equations lack intentionality, that is, it is hard to "reverse engineer" the completed equations, to understand the relations of the variables and coefficients to the dynamics displayed by the simulation. Because of this, validation is problematic; it is hard to ensure that the equations describe the very aspects that they are believed to describe. Consequently, adding new components to an existing model may result in abandoning the model altogether and starting from scratch. This is a significant drawback, especially when modeling systems with several components, such as for instance in [15].

In this paper, we study a potential solution to "open up" the difference equations. As the main contribution, inspired by graph models of computation [10] and category theory [11], we formalize particle systems using mathematical operators. The operators form together a network that collectively represents the difference equation used for the computation. Then, rather than using a fixed set of difference equations as the model, the model is defined as an operator network. In this way, the model remains open to new operators, and the operators remain independent and reusable. Furthermore, as the operators speak only of some isolated computations, the model becomes modular and readable, bringing in the much needed intentionality.

Formally, as discussed in Section 2, an operator is expressed as a higher-order function. It returns a set of difference equations based on given parameters. The difference equations are then used in the simulation, to compute the particle system dynamics. Moreover, as we formalize the operator connections using the standard forward composition, even the entire operator network remains also as an operator.

The use of operators and networks is illustrated in Sections 3 and 4, when we model the construction and the Newtonian dynamics of a simple hourglass. Then, in Section 5, we present the simulation results for the hourglass example, and discuss what is emergent in the dynamics. Finally, in Section 6, follows the conclusion.

2. Formalization

We start by formalizing particles, operators, and operator networks. We formalize particles as vectors over 4-tuples. We then formalize operators as higher-order functions and operator networks as forward compositions of operators. We also formalize the repeated application of an operator as an operator network.

2.1. Particles

In general, a particle may be of any shape [22]; however, we assume throughout this paper that a particle is a sphere with a varying radius, but with a constant and uniformly distributed mass. Consequently, we model particles using only four variables: position, velocity, acceleration, and radius. From these variables, only the radius is a positive real-valued number, $\mathbb{R}_+$, whereas, all the other variables are three-dimensional vectors, $\mathbb{R}^3$. As $\mathbb{R}^3$ refers here to a subspace of Euclidean space, we shall denote the components of $\mathbb{R}^3$ by $\mathtt{x}$, $\mathtt{y}$, and $\mathtt{z}$, respectively.

Formally, a particle type $\mathbf{P}$ is a 4-tuple:

$$\mathbf{P} \triangleq (\mathbb{R}^3, \mathbb{R}^3, \mathbb{R}^3, \mathbb{R}_+) \tag{1}$$

In $\mathbf{P}$, the components represent in the given order the position, the velocity, the acceleration, and the radius. We shall denote these components in the sequel by $\mathtt{p}$, $\mathtt{v}$, $\mathtt{a}$, and $\mathtt{r}$, respectively. Then, for instance, the velocity of a particle $\mathbf{p}$ along the y-axis is referred to as $\mathbf{p}_{\mathtt{v}_\mathtt{y}}$. A set of particles is expressed as a particle vector; thus, for instance, $\mathbf{P}^{10}$ captures 10 particles.

2.2. Operators

An operator is formalized as a parametrized collection of maps. In general, a map is a function from one domain to another. In this paper, we discuss mostly maps that are functions from one particle vector to another. The particle vectors need not be of the same dimension. Such a map from a vector with m particles to a vector with n particles is expressed as $\mathbf{P}^m \to \mathbf{P}^n$.

Let Σ denote a (possibly empty) parameter domain. Then, an operator $\mathtt{o}$ is expressed as a higher-order function of the form $\Sigma \to (\mathtt{P}^m \to \mathtt{P}^n)$. Thus, for some parameter values $\sigma \in \Sigma$, the expression $\mathtt{o}(\sigma)$ returns a map of the form $\mathtt{P}^m \to \mathtt{P}^n$, and for a vector $\mathtt{p}$ of the form $\mathtt{P}^m$, the expression $\mathtt{o}(\sigma)(\mathtt{p})$ returns a vector of the form $\mathtt{P}^n$.

2.3. Networks

Operators can be applied in sequence. As operators are higher-order functions, the sequential application of operators falls under the standard forward composition [1]. Consequently, a sequence of operators, an operator network, remains as an operator also in the formal sense.

Consider operators $\mathtt{a} : \Sigma \to (\mathtt{P}^m \to \mathtt{P}^n)$ and $\mathtt{b} : \Gamma \to (\mathtt{P}^n \to \mathtt{P}^k)$, parameters $\sigma \in \Sigma$ and $\gamma \in \Gamma$, and a vector $\mathtt{p}$ of the form $\mathtt{P}^m$. Then, the forward composition $\mathtt{a}(\sigma) \bullet \mathtt{b}(\gamma)$ gives the map $\mathtt{P}^m \to \mathtt{P}^k$ and is defined as:

$$(\mathtt{a}(\sigma) \bullet \mathtt{b}(\gamma))(\mathtt{p}) \triangleq \mathtt{b}(\gamma)(\mathtt{a}(\sigma)(\mathtt{p})) \tag{2}$$

Please cite this article in press as: M. Rönkkö et al., Beyond particle systems: Operator networks, Simulat. Modell. Pract. (2008), doi:10.1016/j.simpat.2008.05.003

Note that we use here forward composition, so that the application of the operators in a sequence can be read from left to right. This clarifies the equations later, when we define sequences of operators. Consider a variable i, which may appear in σ, and some $j < k$. Then, the repeated application of an operator a, denoted by $\mathsf{a}^{i:j..k}(\sigma)$, is defined as:

$$\mathsf{a}^{i:j..k}(\sigma) \triangleq \mathsf{a}^{i:j}(\sigma) \bullet \mathsf{a}^{i:j+1..k}(\sigma) \tag{3}$$

Here, $\mathsf{a}^{i:j}(\sigma)$ denotes the application $\mathsf{a}(\sigma)$ with i having the value j. Note that the well-definedness of a repeated application of an operator depends on the operator and the given parameters, and has to be checked for each application separately.

3. Constructing an hourglass

We illustrate next use of the operators in modeling and simulating the construction of an hourglass. As we wish to model a symmetric hourglass, we construct it as a spin shape. A spin shape is a three-dimensional shape that is obtained by revolving a two-dimensional profile around an axis. Spin shapes are frequently used when producing objects, such as bottles, glasses, vases, and tires, with CAD programs.

Although the process of constructing a spin shape from a profile sounds simple enough, it is not trivial to model it mathematically. Clearly, the final spin shape can be expressed implicitly, by describing the relation of the object boundary to the profile; however, this does not help directly in modeling the actual construction process.

Technically, a spin shape is constructed in three phases: constructing a profile, growing a shape with the profile, and pruning away overlapping particles. We shall formalize each of these phases as operators. We then formalize the construction of a spin shape as an operator network. Lastly, we use the operator network in computing the actual shape of the hourglass.

3.1. Constructing a profile

We start by formalizing the construction of a two-dimensional profile for the spin shape. The profile is given as a particle vector, and we construct it from a two-dimensional matrix. In the matrix, each element describes the radius of a particle in the corresponding position of a two-dimensional space. An element of the matrix indicates the existence of a particle only if it has a positive value; thus, an element of the matrix with a non-positive value indicates the absence of a particle in the corresponding position of the two-dimensional space.

To formalize the construction of a profile, we define one primitive operator, merge. It merges the particles of one vector with the particles of another. With the merge operator, we then define another operator, add, that adds new particles to a particle vector based on a given matrix of radii. The actual construction of a profile is then formalized as an application of the add operator, and we define it as the $\mathsf{profile}$ operator. We shall now present the formalization.

3.1.1. Merging

The merge operator merges two particle vectors. It is of the form $\mathsf{merge} : \mathsf{P}^n \to (\mathsf{P}^m \to \mathsf{P}^{m+n})$, and we define it as an augmented vector:

$$\mathsf{merge}(\mathsf{u})(\mathsf{p}) \triangleq \begin{bmatrix} \mathsf{p} \\ \underline{\mathsf{u}} \end{bmatrix} \tag{4}$$

Thus, the operator returns a new vector, where the top is the original vector and the bottom is the vector given as a parameter.

3.1.2. Adding

We define the operator for adding new particles in steps. As the operator adds particles based on a given matrix of radii, we start by defining first the addition of a single particle. We then define the addition of particles based on a row of a matrix, and, lastly, we define the addition of particles based on the entire two-dimensional matrix. We define all these additions as operators with the same name, as it is clear by the parameters which one of them is to be applied.

We start by formalizing the addition of a single particle with a given initial position $\vec{p}$ and radius r, but with a zero initial velocity and acceleration. As the particle is added only if it has a positive radius, the operator is of the form $\mathsf{add} : (\mathbb{R}^3, \mathbb{R}) \to (\mathsf{P}^m \to \mathsf{P}^n)$ with either $n = m$ or $n = m + 1$. We define the operator as:

$$\mathsf{add}(\vec{p}, r)(\mathsf{p}) \triangleq \begin{cases} \mathsf{merge}((\vec{p}, \emptyset, \emptyset, r))(\mathsf{p}), & \text{if } r > 0 \\ \mathsf{p}, & \text{if } r \leqslant 0 \end{cases} \tag{5}$$

Next, we formalize the addition of new particles based on a row i of a matrix $\hat{r}$ of the form $\mathbb{R}^{u \times v}$. The operator is of the form $\mathsf{add} : (\mathbb{I}, \mathbb{R}^{u \times v}) \to (\mathsf{P}^m \to \mathsf{P}^n)$, where $m \leqslant n \leqslant m + v$ depending on the given row of the matrix. We define the operator as:

$$\mathsf{add}(i, \hat{r})(\mathsf{p}) \triangleq \mathsf{add}^{j:1..v}((j-1, u-i, 0), \hat{r}_{(i,j)})(\mathsf{p}) \tag{6}$$

Please cite this article in press as: M. Rönkkö et al., Beyond particle systems: Operator networks, Simulat. Modell. Pract. (2008), doi:10.1016/j.simpat.2008.05.003

Note that, for the new particles, the vertical position is an inverted index, $u - i$, to retain the vertical orientation of the shape induced by the matrix.

Lastly, we formalize the addition of new particles based on the entire matrix $\hat{r}$ of the form $\mathbb{R}^{u\times v}$. The operator is of the form $\mathsf{add} : (\mathbb{R}^{u\times v}) \to (\mathsf{P}^m \to \mathsf{P}^n)$ with $m \leqslant n \leqslant m + uv$, and we define it as:

$$\mathsf{add}(\hat{r})(\mathsf{p}) \triangleq \mathsf{add}^{i:1..u}(i, \hat{r})(\mathsf{p}) \tag{7}$$

3.2. Constructing

We formalize the construction of a profile from a matrix of radii, $\hat{r}$ of the form $\mathbb{R}^{u\times v}$, as an operator of the form $\mathsf{profile} : \mathbb{R}^{u\times v} \to (\emptyset \to \mathsf{P}^m)$. We define it as an application of the add operator:

$$\mathsf{profile}(\hat{r})() \triangleq \mathsf{add}(\hat{r})(\emptyset) \tag{8}$$

Thus, as the particles are added to an empty particle vector, $(\emptyset)$, the profile is a particle vector that is induced in its entirety by the matrix of radii, $\hat{r}$.

3.3. Growing a shape

Next, we formalize the second phase in constructing a spin shape: the growing of a shape with a profile. We assume that the profile is given as a particle vector. Then, the shape is grown by rotating the profile about the y-axis, and by merging it with the existing shape.

To formalize the growing, we need in addition to the previously defined primitive operator, merge, another operator, rotate. It rotates the particles of a vector about the y-axis. The actual growing is then formalized as a composite of the two primitive operators, and we define it as the grow operator. We shall now present the formalization.

3.3.1. Rotating

The rotation operator rotates all the particles of a vector by a given angle about the y-axis. For the rotation operator, we first define three auxiliary functions: ang, dist, and rot. The function $\mathsf{ang} : \mathbb{R}^3 \to \mathbb{R}$ returns the angle of a three-dimensional coordinate $\vec{p}$ on the x–z plane, and it is defined as [20]:

$$\mathsf{ang}(\vec{p}) \triangleq \begin{cases} \arctan(\vec{p}_z/\vec{p}_x) + \pi, & \text{if } \vec{p}_x < 0 \\ \arctan(\vec{p}_z/\vec{p}_x), & \text{if } \vec{p}_x \geqslant 0 \end{cases} \tag{9}$$

The function $\mathsf{dist} : \mathbb{R}^3 \to \mathbb{R}$ returns the distance of a three-dimensional coordinate $\vec{p}$ to the y-axis, and it is defined as [20]:

$$\mathsf{dist}(\vec{p}) \triangleq \sqrt{\vec{p}_z^2 + \vec{p}_x^2} \tag{10}$$

The function $\mathsf{rot} : (\mathbb{R}^3, \mathbb{R}) \to \mathbb{R}^3$ returns a three-dimensional coordinate that is obtained by rotating the original coordinate $\vec{p}$ for θ radians about the y-axis, and it is defined as [20]:

$$\mathsf{rot}(\vec{p}, \theta) \triangleq (\mathsf{dist}(\vec{p}) \cos(\mathsf{ang}(\vec{p}) + \theta), \vec{p}_y, \mathsf{dist}(\vec{p}) \sin(\mathsf{ang}(\vec{p}) + \theta)) \tag{11}$$

Now, the rotation operator is of the form $\mathsf{rotate} : \mathbb{R} \to (\mathsf{P}^m \to \mathsf{P}^m)$, and we define it as:

$$\mathsf{rotate}(\theta)(\mathsf{p}) \triangleq \mathsf{u}, \quad \text{where each } \mathsf{u}_i = (\mathsf{rot}(\mathsf{p}_{i_p}, \theta), \mathsf{p}_{i_v}, \mathsf{p}_{i_a}, \mathsf{p}_{i_r}) \tag{12}$$

3.3.2. Growing

We present now the formalization for the growing of a shape p with a profile u that is rotated by θ radians about the y-axis. The operator is of the form $\mathsf{grow} : (\mathsf{P}^n, \mathbb{R}) \to (\mathsf{P}^m \to \mathsf{P}^{m+n})$, and we define it as the composite:

$$\mathsf{grow}(\mathsf{u}, \theta)(\mathsf{p}) \triangleq \mathsf{merge}(\mathsf{rotate}(\theta)(\mathsf{u}))(\mathsf{p}) \tag{13}$$

3.4. Pruning

We formalize next the last phase of constructing a spin shape: pruning away overlapping particles of a shape. We assume that the shape is given as a particle vector. Then, the pruning removes all those particles of the shape that overlap to a given degree with the rest of the particles.

Clearly, the pruning process, as described above, is ambiguous. Interestingly, it is easier to formalize the pruning process unambiguously as an inverse process, as a reconstruction process. Then, we first define a conditional merge operator that merges a single particle with a particle vector, provided that its distance to any of the vector particles is greater than a given

Please cite this article in press as: M. Rönkkö et al., Beyond particle systems: Operator networks, Simulat. Modell. Pract. (2008), doi:10.1016/j.simpat.2008.05.003

lower limit. The reconstruction process is then formalized as a repeated application of the conditional merge operator, and we define it as the prune operator. We shall now present the formalization.

3.4.1. Conditional merging

We formalize the conditional merge operator by using the primitive merge operator that was defined earlier. The conditional merge operator is of the form $\mathsf{merge} : (\mathsf{P}, \mathbb{R}) \to (\mathsf{P}^m \to \mathsf{P}^n)$, where either $n = m$ or $n = m + 1$ holds. We define it for a particle u and a lower limit d as:

$$\mathsf{merge}(\mathsf{u}, d)(\mathsf{p}) \triangleq \begin{cases} \mathsf{p}, & \text{if } \|\mathsf{u}_\mathsf{p} - \mathsf{p}_{i_\mathsf{p}}\| \leqslant d \text{ holds for any} i \\ \mathsf{merge}(\mathsf{u})(\mathsf{p}), & \text{otherwise} \end{cases} \tag{14}$$

Thus, the particle u is merged with the particle vector p, only if its distance to any particle p_i is greater than d. The distance between u and p_i is computed as the Euclidean norm over the difference of their current positions, $\|\mathsf{u}_\mathsf{p} - \mathsf{p}_{i_\mathsf{p}}\|$.

3.4.2. Pruning

We present now the formalization for the pruning of a shape p with m particles, such that the distance between any pair of the remaining particles is greater than d units. The pruning operator is of the form $\mathsf{prune} : (\mathbb{R}) \to (\mathsf{P}^m \to \mathsf{P}^n)$, where $0 \leqslant n \leqslant m$ holds, and we define it as a repeated application of the conditional merge operator:

$$\mathsf{prune}(d)(\mathsf{p}) \triangleq \mathsf{merge}^{i:1..m}(\mathsf{p}_i, d)(\emptyset) \tag{15}$$

Note that, above, the pruning is defined as a reconstruction process that starts always with an empty particle vector, $(\emptyset)$.

3.5. Constructing a spin shape

As mentioned earlier, a spin shape is constructed in three phases: constructing a profile, growing a shape with the profile, and pruning away overlapping particles. We present now the model of the construction process using the operators defined above. We assume throughout this paper that a spin shape is constructed from 200 segments, where the ith segment is the profile rotated by $\frac{i\pi}{100}$ radians. We also assume that the spin shape is not dense, so that the distance of any two particles is at least 0.5 units.

Let $\hat{r}$ be the matrix of radii. We formalize the construction of a spin shape from $\hat{r}$ as an operator of the form $\mathsf{spin} : \mathbb{R}^{u \times v} \to (\emptyset \to \mathsf{P}^m)$, and we define it as:

$$\mathsf{spin}(\hat{r})() \triangleq \left(\mathsf{grow}^{i:0..199} \left(\mathsf{profile}(\hat{r})(), \frac{i\pi}{100} \right) \bullet \mathsf{prune}(0.5) \right)(\emptyset) \tag{16}$$

Thus, the profile is first created from the matrix of radii, $\hat{r}$. Then, the profile is used for growing the spin shape iteratively. Note that growing process starts with an empty particle vector, $(\emptyset)$. Finally, the grown shape is pruned so that the distance of any two particles is at least 0.5 units.

The spin operator above is a network of operators. The network is depicted in Fig. 1. The figure clearly shows that there are only two primitive operators involved in the construction of a spin shape: merge and rotate.

3.6. Constructing the hourglass

We shall now show how the hourglass is constructed using the spin operator. The hourglass is a volumetric shape that is composed of an exterior and a body of sand. Both of these components are constructed as spin shapes.

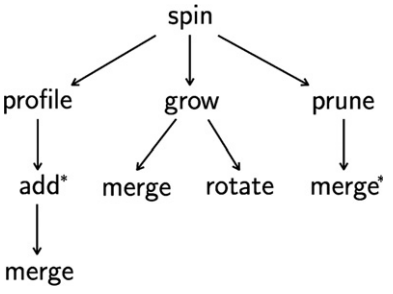

Fig. 1. The network of operators modeling the construction on a spin shape, spin. The asterisk $(*)$ indicates the application of a sequence of operators with the same name.

Please cite this article in press as: M. Rönkkö et al., Beyond particle systems: Operator networks, Simulat. Modell. Pract. (2008), doi:10.1016/j.simpat.2008.05.003

For the construction, we first define two matrices of radii, one for the exterior and the other for the body of sand. As the matrices are large, we show them schematically in Fig. 2. In the figure, the matrix for the exterior is the matrix $\hat{e}$, and the matrix for the body of sand is the matrix $\hat{s}$.

The spin shapes for both the exterior and the body of sand are the given by the spin operator. As for the exterior of the hourglass, we capture its construction as an operator of the form $\mathsf{exterior} : (\emptyset) \to (\emptyset \to \mathsf{P}^m)$, and we define it as:

$$\mathsf{extrior}()() \triangleq \mathsf{spin}(\hat{e})() \tag{17}$$

Similarly, for the body of sand, we capture its construction as an operator of the form $\mathsf{sand} : (\emptyset) \to (\emptyset \to \mathsf{P}^m)$, and we define it as:

$$\mathsf{sand}()() \triangleq \mathsf{spin}(\hat{e})() \tag{18}$$

The spin shapes for the two distinct elements of the hourglass are shown in Fig. 3. The figure also shows the composite shape of the two individual shapes. Note that the composite shape has 5621 particles that are all constructed by the operators from the two matrices of radii.

4. Modeling Newtonian dynamics

We formalize next the Newtonian dynamics for the hourglass. As the exterior of the hourglass is assumed to stay put, we only consider the dynamics of the sand. As the body of sand is composed of individual particles, the Newtonian dynamics describes only the dynamics of the individual sand particles. The dynamics of the whole body of sand is then given as composite dynamics.

The Newtonian dynamics is computed using the Newtonian model of motion. In the Newtonian model of motion, the position of a sand particle changes depending on its velocity, which, in turn, changes depending on its acceleration. The accel-

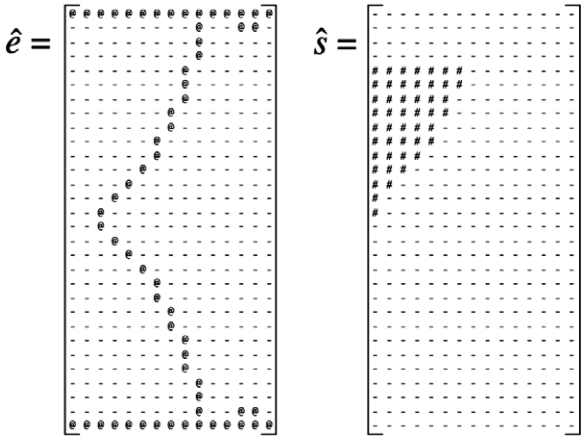

Fig. 2. Two matrices of radii; the matrix for the exterior, $\hat{e}$, on the left and the matrix for the body of sand, $\hat{s}$, on the right. Both of the matrices have the dimensions 30×15. In the matrices, each element captures a particle radius; "-" denotes the value 0, "@" denotes the value 1.4, and "#" denotes the value 0.9.

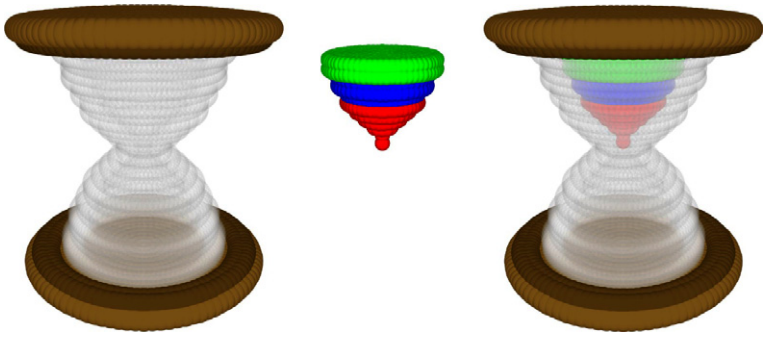

Fig. 3. The resulting spin shapes: the exterior on the left, body of sand in the middle, and the composite shape on the right. The colors have been added for illustrative purposes.

Please cite this article in press as: M. Rönkkö et al., Beyond particle systems: Operator networks, Simulat. Modell. Pract. (2008), doi:10.1016/j.simpat.2008.05.003

eration of a sand particle is determined by forces affecting it. There are four kinds of forces affecting a sand particle simultaneously: a gravity force, a dampening force, collision forces between the sand particles, and collision forces between the sand and exterior particles. The collision forces between the sand particles keep the sand particles apart, whereas the collision forces between the sand and exterior particles keep the sand particles inside the hourglass.

To capture all this, we start by formalizing the forces as operators gravitate, dampen, keepApart, and keepInside. To formalize the Newtonian model of motion, we also define one more primitive operator, integrate, that integrates all the forces into the motion. In effect, the integrate operator computes the Euler approximation of the Newtonian dynamics. The Newtonian model of motion is then given as an operator network consisting of the force operators and the integrate operator. Lastly, we discuss briefly the relation of the operator network to a set of difference equations used in capturing similar kind of non-linear dynamics with particle systems.

4.1. Gravity

We formalize first the gravity force on sand particles. We assume that the sand particles are given as a particle vector.

For the formalization, we define first one primitive operator, force, that captures the effect of a constant force. With it, we can then formalize the constant gravity force as an operator gravitate. We shall now present the formalization.

4.1.1. Forcing

A constant force on a particle is modeled implicitly as its effect, as acceleration. Thus, an operator modeling a constant force adds a constant acceleration to the particles. Let p be a particle vector of the form P^m. Also, let $\hat{a}$ be a matrix of the form $\mathbb{R}^{m\times 3}$, capturing the acceleration on the particles. Then, a constant force is modeled as an operator of the form $\mathsf{force} : (\mathbb{R}^{m\times 3}) \to (\mathsf{P}^m \to \mathsf{P}^m)$, and we define it as:

$$\mathsf{force}(\hat{a})(\mathsf{p}) \triangleq \mathsf{u}, \quad \text{where each } \mathsf{u}_i = (\mathsf{p}_{i_\mathsf{p}}, \mathsf{p}_{i_\mathsf{v}}, \mathsf{p}_{i_\mathsf{a}} + \hat{a}_i, \mathsf{p}_{i_\mathsf{r}}) \tag{19}$$

4.1.2. Gravitating

With the force operator, we formalize the constant gravity force. Let p be a particle vector of the form P^m, and $\hat{g}$ be a vector of the form $\mathbb{R}^{m\times 3}$ capturing the acceleration due to gravity. The constant gravity force is then modeled as an operator of the form $\mathsf{gravitate} : (\emptyset) \to (\mathsf{P}^m \to \mathsf{P}^m)$, and we define it as:

$$\mathsf{gravitate}()(\mathsf{p}) \triangleq \mathsf{force}(\hat{g})(\mathsf{p}) \text{where each } \hat{g}_i = (0, -300, 0) \tag{20}$$

Note that the coefficient -300 above is a computational coefficient, and, deviates therefore from the gravity force as known in physics [24]. The relation between the known gravity constant, $-9.81\ m/s^2$, and the constant -300 becomes fixed later, when we define the model of motion used for approximating the motion dynamics. As the model of motion uses a fixed time step, 0.01, the used gravity constant -300 scales back to the value $-3.00\ \mathrm{m/s^2}$. Thus, in a way, the gravity force defined in Eq. (20) includes also the effect of a strong air resistance.

4.2. Dampening

We formalize next the dampening force on the sand particles. The dampening force captures loss of kinetic energy; thus, the magnitude of the dampening force depends on the velocity of a particle. Because of this, the sand particles have a tendency to stop moving in the absence of external forces. We assume in the sequel that there is only a moderate dampening force affecting the sand particles.

We formalize the dampening force using the primitive force operator defined above. Let p be a particle vector of the form P^m, and $\hat{d}$ be a vector of the form $\mathbb{R}^{m\times 3}$ capturing the acceleration due to dampening. The dampening force on the sand particles is then modeled as an operator of the form $\mathsf{dampen} : (\emptyset) \to (\mathsf{P}^m \to \mathsf{P}^m)$, and we define it as:

$$\mathsf{dampen}()(\mathsf{p}) \triangleq \mathsf{force}(\hat{d})(\mathsf{p}), \quad \text{where each } \hat{d}_i = -5\mathsf{p}_{i_\mathsf{v}} \tag{21}$$

Above, the factor -5 translates to a moderate loss of kinetic energy. As such, it is only a computational coefficient, and does not correspond to a physical constant. The relation of the dampening constant -5 to the motion velocity becomes fixed later, when we define the model of motion used for approximating the motion dynamics. As the model of motion uses a fixed time step, 0.01, the used dampening constant -5 scales back to the value -0.05, which means a 5% loss of kinetic energy in the motion.

4.3. Keeping the sand particles apart

We formalize next the collision forces between the sand particles. Such forces keep the sand particles apart by pushing any overlapping sand particles away from each other. Two sand particles are considered to overlap, when their distance is less than the sum of their radii.

Please cite this article in press as: M. Rönkkö et al., Beyond particle systems: Operator networks, Simulat. Modell. Pract. (2008), doi:10.1016/j.simpat.2008.05.003

To formalize the collision forces between the sand particles, we first define a new primitive operator, collide, that models generic collision forces between two particle vectors. With it, we then formalize the collision forces between the sand particles as an operator keepApart. We shall now present the formalization.

4.3.1. Colliding

We model the generic collision forces on particles as their effect, as acceleration on the particles. We formalize the collision forces as an operator that adds acceleration due to collision forces on a particle vector that collides with another particle vector. The magnitude of the acceleration is controlled with a coefficient matrix.

For the collision operator, we first define an auxiliary function, *col*. It returns for a single particle the acceleration due to collision forces with a particle vector. Such an acceleration is the sum of all the accelerations due to the individual particle-to-particle collisions. The magnitude of the particle-to-particle collision forces are controlled with a coefficient matrix. Let p be the colliding particle, u be a particle vector of the form P^n, and $\hat{c}$ be a coefficient matrix of the form $\mathbb{R}^{\mathbb{n}}$. Then, the function $col : (P, \mathbb{R}^{\mathbb{n}}, \mathbb{P}^{\mathbb{n}}) \rightarrow \mathbb{R}^3$ is defined as:

$$\mathsf{col}(\mathsf{p}, \hat{c}, \mathsf{u}) \triangleq \sum_{i=1}^{n} \hat{c}_i(\mathsf{u}_{i_\mathsf{p}} - \mathsf{p}_\mathsf{p}) \min\left\{0, 1 - \frac{(\mathsf{u}_{i_\mathsf{r}} + \mathsf{p}_\mathsf{r})^2}{\|\mathsf{u}_{i_\mathsf{p}} - \mathsf{p}_\mathsf{p}\|^2}\right\} \tag{22}$$

Here, for the pair of particles p and u_i, the $\min\{\cdot\}$ expression evaluates to 0 if the two particles do not overlap; then, their distance is greater than the sum of their radii. Consequently, the particles are considered to collide, only if they overlap. Then, the collision force for the particle p is directed directly away from the particle u_i, because the $\min\{\cdot\}$ expression becomes negative. The magnitude of the collision force is controlled by $\hat{c}_i$.

The col function, as defined above, is a generalization of the collision function used before for computing the collision of particles in a particle system [5–7,12,14–16]. The characteristics of that collision function are discussed in more detail in a technical report [13].

We formalize now the generic collision operator for a particle vector. The formalization uses the primitive force operator. Let p be the colliding particle vector of the form P^m. Also, let u be a particle vector of the form P^n capturing the particles that collide with p. Furthermore, let $\hat{c}$ be a matrix of the form $\mathbb{R}^{m\times n}$ capturing the collision coefficients for all the pairs of particles of p and u. Then, the generic collision operator, collide, is of the form $\mathsf{collide} : (\mathsf{P}^n, \mathbb{R}^{m\times n}) \rightarrow (\mathsf{P}^m \rightarrow \mathsf{P}^m)$, and we define it as:

$$\mathsf{collide}(\mathsf{u}, \hat{c})(\mathsf{p}) \triangleq \mathsf{force}(\hat{a})(\mathsf{p}), \quad \text{where each } \hat{a}_i = \mathsf{col}(\mathsf{p}_i, \hat{c}_i, \mathsf{u}) \tag{23}$$

4.3.2. Keeping apart

As mentioned earlier, the purpose of the collision forces between the sand particles is to keep the sand particles apart. We formalize now the collision forces between the sand particles as an operator. Let p be a particle vector of the form P^m capturing the sand particles. Also, let $\hat{k}$ be a coefficient matrix of the form $\mathbb{R}^{m\times m}$ capturing the magnitude of the collision forces between each pair of sand particles. Then, the operator keeping the sand particles apart is of the form $\mathsf{keepApart} : (\emptyset) \rightarrow (\mathsf{P}^m \rightarrow \mathsf{P}^m)$, and we define it as:

$$\mathsf{keepApart}()(\mathsf{p}) \triangleq \mathsf{collide}(\mathsf{p}, \hat{k})(\mathsf{p}), \quad \text{where } \hat{k}_{ij} = \begin{cases} 100, & \text{if } i \neq j \\ 0, & \text{if } i = j \end{cases} \tag{24}$$

Thus, all the sand particles collide with a force that has the same collision coefficient, 100. Moreover, a sand particle can never collide with itself. The relation of the collision constant 100 to the motion velocity becomes fixed later, when we define the model of motion used for approximating the motion dynamics. As the model of motion uses a fixed time step, 0.01, the used collision constant 100 scales back to the value 1, meaning that the collision force between the sand particles is exactly that obtained by the function in Eq. (22).

4.4. Keeping the sand particles inside the hourglass

We formalize next the collision forces between the sand particles and the exterior particles. Such forces keep the sand particles inside the hourglass.

We formalize the collision forces between the sand and exterior particles using the previously defined collide operator. Furthermore, we use the exterior() operator to capture the exterior particles, as defined earlier in Section 3.6. The operator returns a particle vector of the form P^n. Let p be a particle vector of the form P^m capturing the sand particles, and $\hat{c}$ be a coefficient matrix of the form $\mathbb{R}^{m\times n}$. Then, the operator keeping the sand particles inside the hourglass is of the form $\mathsf{keepInside} : (\emptyset) \rightarrow (\mathsf{P}^m \rightarrow \mathsf{P}^m)$, and we define it as:

$$\mathsf{keepInside}()(\mathsf{p}) \triangleq \mathsf{collide}(\mathsf{exterior}(), \hat{c})(\mathsf{p}), \quad \text{where each } \hat{c}_{ij} = 200 \tag{25}$$

Thus, each sand particle collides with each exterior particle. The collision coefficient for each collision force between the sand and exterior particle is 200, whereby the collision forces between the sand and exterior particles are two times stronger than

the collision forces among the sand particles. The collision coefficients reflect the fact that the body of sand is a fluid material with respect to the exterior. The relation of the collision constant 200 to the motion velocity becomes fixed later, when we define the model of motion used for approximating the motion dynamics. As the model of motion uses a fixed time step, 0.01, the used collision constant 200 scales back to the value 2, meaning that the collision force between the sand particles is two times stronger than that obtained by the function in Eq. (22).

4.5. The Newtonian model of motion

We formalize next the Newtonian model of motion for the sand particles. In the Newtonian model of motion, as mentioned earlier, the position of a sand particle changes depending on its velocity, which, in turn, changes depending on its acceleration. The acceleration of sand particles is determined by the four forces formalized above: a gravity force, a dampening force, collision forces between the sand particles, and collision forces between the sand and exterior particles. The forces affect the sand particles simultaneously.

We start by formalizing an operator netForce that captures the simultaneous effect of the forces on the sand particles. We then formalize the integration of the forces into the motion as a primitive operator integrate. With these operators, we then define the Newtonian model of motion as an operator network, captured by an operator called motion.

4.5.1. The net force

All the forces affect the sand particles simultaneously. We model the simultaneous effect, the net force, as an operator network. As the operators for the forces are mutually associative and commutative, the order of the operators in the network is free. Let p be a particle vector of the form P^m capturing the sand particles. We formalize the net force as an operator of the form $\mathsf{netForce} : (\emptyset) \to (\mathsf{P}^m \to \mathsf{P}^m)$, and we define it as:

$$\mathsf{netForce}()(\mathsf{p}) \triangleq (\mathsf{gravitate}() \bullet \mathsf{dampen}() \bullet \mathsf{keepApart}() \bullet \mathsf{keepInside}())(\mathsf{p}) \tag{26}$$

4.5.2. Integrating

We integrate the forces into the motion by using Euler approximation [25]. More specifically, we define the operator that integrates the acceleration to the velocity and the velocity to the position of particles as an Euler approximation of the trivial Newtonian model of motion [24] with forward error correction. As Euler approximation is a numerical approximation, the resulting accuracy depends on a coefficient that acts as a fixed time step. Let p be a particle vector of the form P^m, and t be the fixed time step. Then, the integration is an operator of the form $\mathsf{integrate} : (\mathbb{R}) \to (\mathsf{P}^m \to \mathsf{P}^m)$, and we define it as:

$$\mathsf{integrate}(t)(\mathsf{p}) \triangleq \mathsf{u}, \quad \text{where each } \mathsf{u}_i = (\mathsf{p}_{i_\mathsf{p}} + t\mathsf{u}_{i_\mathsf{v}}, \mathsf{p}_{i_\mathsf{v}} + t\mathsf{p}_{i_\mathsf{a}}, \emptyset, \mathsf{p}_{i_\mathsf{r}}) \tag{27}$$

Thus, due to use of forward error correction, the integration operator computes first the updated velocity using the current acceleration. It then computes the updated position using the updated velocity. Lastly, it sets the acceleration back to zero. Note that forward error correction is used, because it improves the stability of simulated dynamics for a particle system [14].

4.5.3. The model of motion

Lastly, we formalize the model of motion for the sand particles. To recall, we do not need to consider the motion of the exterior particles, as the exterior of the hourglass is assumed to stay put. Let p be a particle vector of the form P^m capturing the sand particles. The model of motion is an operator of the form $\mathsf{motion} : (\emptyset) \to (\mathsf{P}^m \to \mathsf{P}^m)$, and we define it as:

$$\mathsf{motion}()(\mathsf{p}) \triangleq (\mathsf{netForce}() \bullet \mathsf{integrate}(0.01))(\mathsf{p}) \tag{28}$$

Thus, the model of motion has two distinct computational phases: computation of the affecting forces, and integration of the forces into the motion. The integration uses 0.01 as the fixed time step, which translates to moderate computation accuracy. Note that if we were to increase the computation accuracy by making the time step smaller, we would also have to rescale the constants for the gravity, dampening and collision forces in Eqs. (20), (21), (24), and (25).

The motion operator is a network of operators, as depicted in Fig. 4. The figure shows clearly, how the model of motion is build on top of two primitive operators: force and integrate. The figure also shows how the model of motion relates back to the operator network for the construction of a spin shape, spin.

4.6. The induced set of difference equations

The model of motion, as captured by the motion operator, induces a set of difference equations. The main body for the difference equations comes from the integrate operator. The netForce operator determines then the acceleration for each sand particle.

As defined earlier, let $\hat{g}$, $\hat{d}$, $\hat{k}$, and $\hat{c}$ be the gravity coefficient vector, the damping coefficient vector, the collision magnitude vector between the sand particles, and the collision magnitude vector between the sand and exterior particles, respectively. Also, let p be particle vector capturing the sand particles, u be a particle vector capturing the exterior particles, and a be a

Please cite this article in press as: M. Rönkkö et al., Beyond particle systems: Operator networks, Simulat. Modell. Pract. (2008), doi:10.1016/j.simpat.2008.05.003

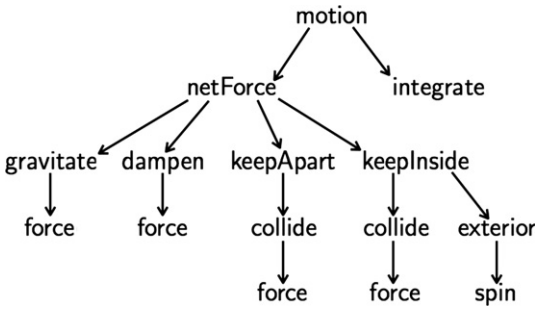

Fig. 4. The network of operators capturing the model of motion for the body of sand in the hourglass. Note that the operator spin is actually an operator network of its own.

vector of the form $\mathbb{R}^{3\times m}$ capturing the acceleration induced by the netForce operator. Then, for a sand particle i, the model of motion, motion, induces the set of difference equations:

$$\begin{aligned} \mathsf{p}'_{i_\mathsf{p}} &= \mathsf{p}_{i_\mathsf{p}} + t\mathsf{p}'_{i_\mathsf{v}} \\ \mathsf{p}'_{i_\mathsf{v}} &= \mathsf{p}_{i_\mathsf{v}} + t\mathsf{a}'_i \\ \mathsf{a}'_i &= \hat{g}_i + \hat{d}_i + \mathrm{col}(\mathsf{p}_i, \hat{k}_i, \mathsf{p}) + \mathrm{col}(\mathsf{p}_i, \hat{c}_i, \mathsf{u}) \end{aligned} \tag{29}$$

The set of difference equations, Eq. (29), is a special case of a set of difference equations used in earlier studies to compute particle system dynamics [5–7,12,14–16]. However, the set of difference equations lacks the intentionality that is made explicit in the operator network, as illustrated in Fig. 4 above. Moreover, the operator network also reveals the modularity and the connectivity between the operators, making it easier to modify and adjust the model intentionally. Also, because of the modularity, it is possible to extend and reuse parts of the operator network in other models coherently.

Interestingly, although the set of difference equations above is far from the standard laws of physics, the dynamics that they capture adheres to the laws of physics. This is discussed in a case study on the realism of particle system dynamics, presented in [12]. The paper presents a method, called simulated tendency, that is used to analyze the computed dynamics for realism.

5. Simulation results

To simulate the dynamics of a hourglass, we first form a model of the hourglass as an operator network using the presented operators. Although the operators are higher-order functions, they are also computable. Thus, an operator network captures not only the formalization of the processes, but also the computation of the simulation.

We shall now present the simulation results for the hourglass, and discuss what is emergent in the simulated dynamics. We then study briefly how the shape of the hourglass affects the dynamics.

5.1. The computed dynamics of the hourglass

Before we can compute the dynamics of an hourglass, we have to fix the model for it. Clearly, in the model, we use all the operators as defined earlier, including the matrices of radii that define the shape of the hourglass. In the sequel, we are interested in the dynamics only for the first 2500 iteration rounds, whereby we fix the model for the hourglass as:

$$\mathsf{hourglass}()() \triangleq \mathsf{motion}^{i:1..2500}(\mathsf{sand}()()) \tag{30}$$

This model reflects very clearly that only the body of sand is to move in the simulation, while the exterior of the hourglass stays put.

The resulting dynamics is illustrated with an image series in Fig. 5. As the figure shows, the behavior of the body of sand is disordered although the model is deterministic; it is impossible to say where an individual sand particle ends up in the lower bulb, after it drops down from the upper bulb.

Fig. 5 indicates also another, more interesting property: some six sand particles remain in the upper bulb. At the first glance, this may seem like an error in the simulation; however, a more careful inspection confirms that the phenomenon is true to the laws of physics. Namely, by studying the matrices shown earlier in Fig. 2, it is easily verified that the opening between the two bulbs in the hourglass is of a circular form with a radius of 0.6 units, whereas the radius of a sand particle is 0.9 units. Thus, a sand particle cannot pass from the upper bulb to the lower bulb without an external force. Moreover, the gravity force on a sand particle, or rather the collective gravity force on the six sand particles, is not enough, either. However, as the body of sand clearly does pass through the opening in the simulation, the simulation must, therefore, capture pressure. As pressure is not addressed in any way in the model; it is an emergent property.

Please cite this article in press as: M. Rönkkö et al., Beyond particle systems: Operator networks, Simulat. Modell. Pract. (2008), doi:10.1016/j.simpat.2008.05.003

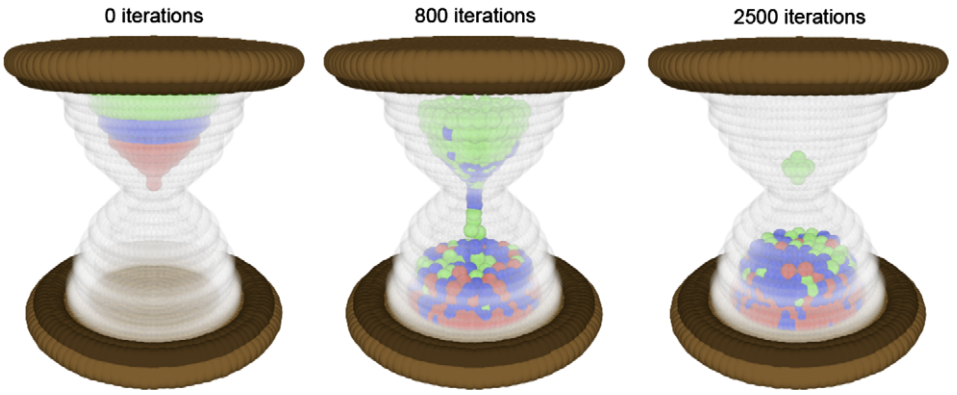

Fig. 5. An image series showing how the sand particles drop down during the simulation; the number of computed iteration rounds is indicated above each image.

To study the emergent pressure in more detail, we have plotted the evolution of the number of sand particles in the upper bulb during the simulation as a graph in Fig. 6. The graph shows a clear exponential decay with a half-life at about 600 iterations, indicating a decay constant λ of about 0.0011. An exponential decay is often associated with many natural science phenomena, including rates of chemical reactions, change in temperature difference, and pressure changes in a medium.

5.2. The effect of the shape to the dynamics

To ensure that the emergent pressure is not just due to the collective dynamics of the sand particles, we study the effect of the shape of the exterior to the dynamics of the body of sand. As mentioned earlier, we have formalized the construction of the hourglass as a process. Therefore, to study the effect, we only need to change the matrices of radii that capture the profile for the hourglass, and compute the simulations with the existing operators.

We study here two cases that are of interest. Firstly, we study how the flow of sand particles is affected if the opening between the two bulbs is widened. Secondly, we study how the flow of sand particles is affected if the two bulbs are widened while the opening between the bulbs remains the same.

5.2.1. Wider opening between the bulbs

In order to construct an hourglass with a wider opening, we need to change only the matrix of radii for the exterior. Such a matrix $\hat{e}$ is shown in Fig. 7. When comparing the matrix to the original matrix, shown earlier in Fig. 2, only the middle rows are changed to widen the opening. Fig. 7 shows also the matrix of radii for the body of sand, $\hat{s}$, which has remained unchanged.

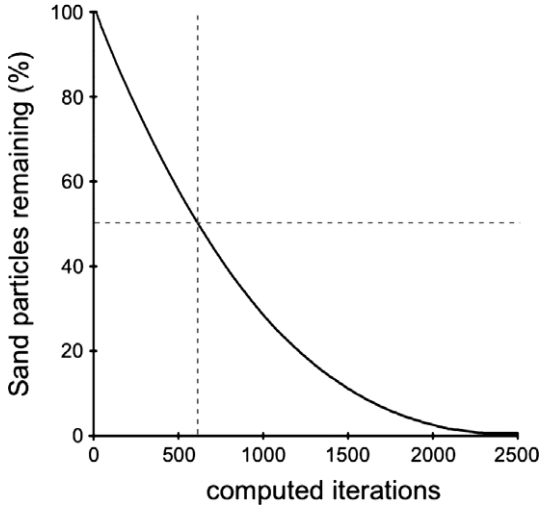

Fig. 6. A graph indicating how the sand particles drop from the upper bulb of the hourglass to the lower bulb during the simulation.

Please cite this article in press as: M. Rönkkö et al., Beyond particle systems: Operator networks, Simulat. Modell. Pract. (2008), doi:10.1016/j.simpat.2008.05.003

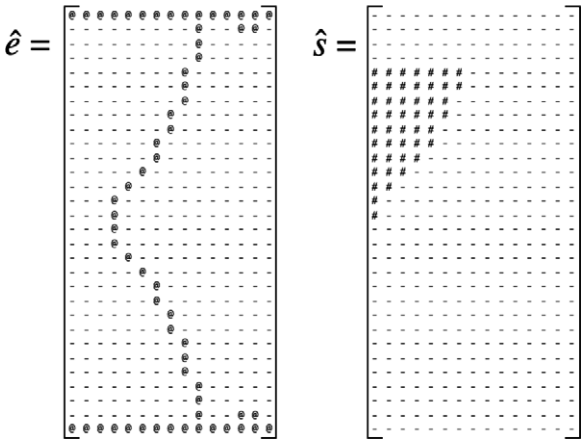

Fig. 7. Two matrices of radii for the hourglass with a wider opening; the matrix for the exterior, $\hat{e}$, on the left and the matrix for the body of sand, $\hat{s}$, on the right. Both of the matrices have the size 30×15. In the matrices, each element captures a particle radius; "-" denotes the value 0, "@" denotes the value 1.4, and "#" denotes the value 0.9.

The simulation result is depicted in Fig. 8, where the evolution of the number of sand particles in the upper bulb during the simulation is plotted as a graph. The figure also shows an intermediate state in the simulation, after 80 iteration rounds, to confirm that the flow of sand particles is now much wider than in the original hourglass, shown earlier in Fig. 5.

The graph in Fig. 8 confirms that the body of sand flows much quicker through the hourglass than in the original case, shown earlier in Fig. 6. Again, the graph in Fig. 8 shows an exponential decay, but this time with a half-life at about 125 iterations. Thus, the decay constant λ is this time about 0.0055. What is particularly important here to note, is that the flow dynamics for the body of sand was changed although we only changed the shape of the exterior. Thus, the exterior particles, although they do not move in any way, do affect the pressure, whereby the emergent pressure is not only due to the sand particles, but rather due to all the particles of the hourglass.

5.2.2. Wider bulbs

In this last test, we study the interplay between the number of the sand particles and the shape of the exterior. Thus, we widen the bulbs of the hourglass while keeping the narrow opening. Furthermore, as the bulbs have a bigger volume, we add more sand particles to the hourglass. The resulting matrices of radii for the exterior and sand particles, $\hat{e}$ and $\hat{s}$, are shown in Fig. 9. Both of the matrices differ now from the two original matrices shown earlier in Fig. 2.

The simulation result is depicted in Fig. 10, where the evolution of the number of sand particles in the upper bulb in the simulation is plotted as a graph. The figure also shows an intermediate state in the simulation, after 400 iteration rounds, to confirm that the flow of sand particles is similar to that of the original hourglass, shown earlier in Fig. 5.

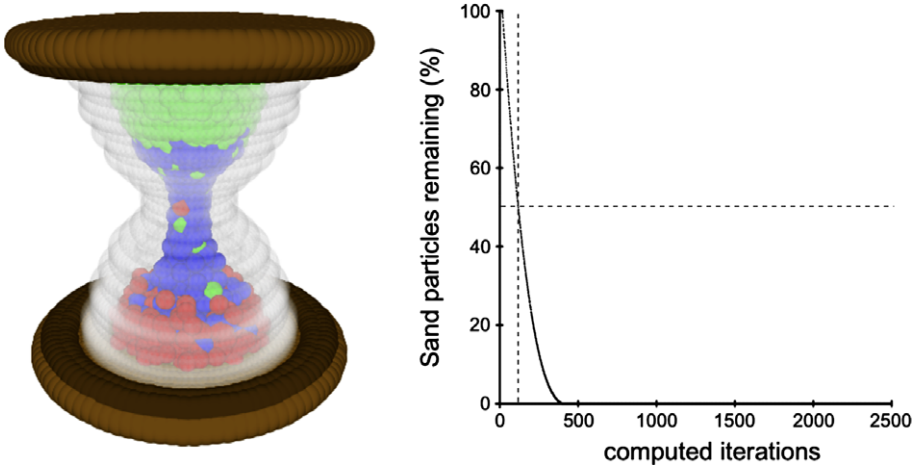

Fig. 8. Image of the hourglass with a wider opening after 80 iteration rounds on the left; a graph indicating how the sand particles drop from the upper bulb of the hourglass to the lower bulb during the computation on the right.

Please cite this article in press as: M. Rönkkö et al., Beyond particle systems: Operator networks, Simulat. Modell. Pract. (2008), doi:10.1016/j.simpat.2008.05.003

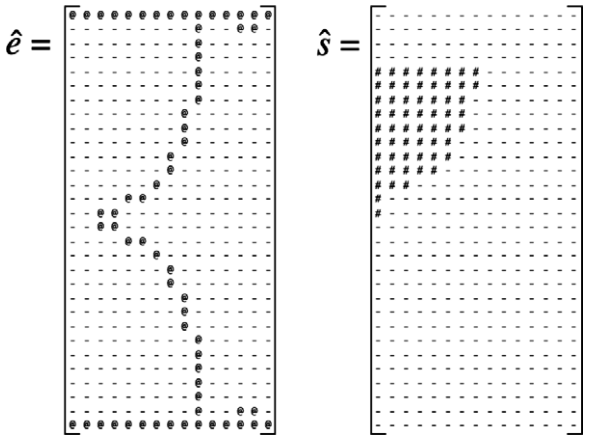

Fig. 9. Two matrices of radii for the hourglass with wider bulbs; the matrix for the exterior, $\hat{e}$, on the left and the matrix for the body of sand, $\hat{s}$, on the right. Both of the matrices have the size 30×15. In the matrices, each element captures a particle radius; "-" denotes the value 0, "@" denotes the value 1.4, and "#" denotes the value 0.9.

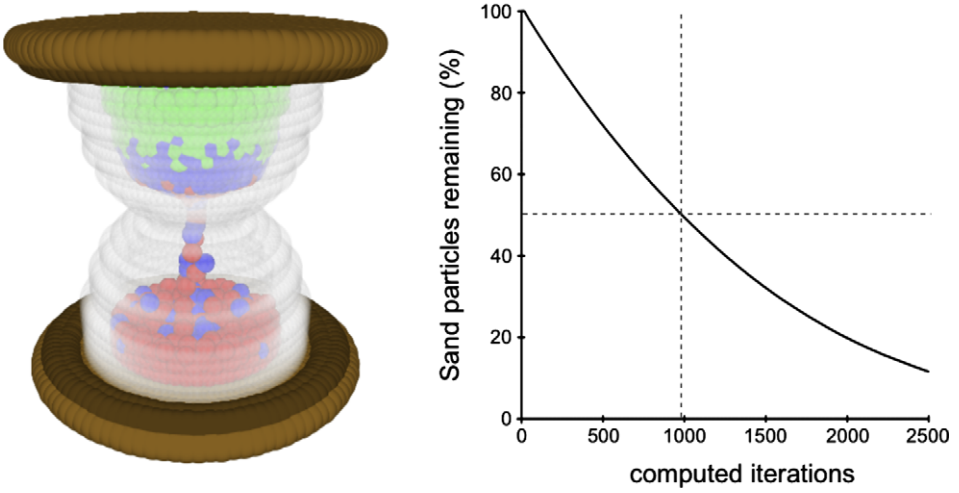

Fig. 10. Image of the hourglass with wider bulbs after 400 iteration rounds on the left; a graph indicating how the sand particles drop from the upper bulb of the hourglass to the lower bulb during the computation on the right.

The graph in Fig. 10 confirms that it takes now a longer time for the body of sand to flow through the hourglass than in the original case, shown earlier in Fig. 6. The graph in Fig. 10 shows again an exponential decay, but this time with a half-life at about 1000 iterations. Thus, the decay constant λ is this time about 0.00069. Consequently, the graph confirms that the emergent pressure captures also the tradeoff properties between the volume and the shape, when considering the flow of sand particles.

In summary, the three different kinds of hourglasses illustrate how significant it is to allow the emergence of properties in a model, and how intricate such properties are even in the simple cases. Based on the shown dynamics, one could argue that embedding of shape sensitive pressure dynamics of shown complexity in any other way would indeed be a daunting task. By using particle based models that support emergent dynamics such details "come for free". Furthermore, as the whole model was formalized using operators, there is now a clear intuitive connection between the initial values, the simulation, and the final results. Because of this, we can be assured that a change in the initial setting, in the shape of the hourglass, remains local to the construction and does not affect the actual model of motion, which was the same in all three cases. Consequently, the resulting changes in the dynamics are provably due to the changes in the shape of the hourglass.

6. Conclusion

In this paper, we discussed simulation of non-linear dynamics using a particle system. As the main contribution, we formalized particle system computations as mathematical operator networks, to gain intentionality and modularity. We

Please cite this article in press as: M. Rönkkö et al., Beyond particle systems: Operator networks, Simulat. Modell. Pract. (2008), doi:10.1016/j.simpat.2008.05.003

illustrated use of operator networks in modeling and simulating the construction and the dynamics of an hourglass. We also showed how an operator network model supports experimentation, as we simulated the dynamics of differently shaped hourglasses. The actual computations were implemented using C++ and a publicly available software package, Atoms [17].

The reason why particle based models are considered attractive, is that they are computationally light and, yet, they exhibit emergent dynamics that is hard or impossible to capture by any other means. Emergent dynamics was also displayed by the dynamics of the hourglass; we discovered an emergent pressure for the body of sand, which we then also studied and discussed. Because the model was an operator network, it was easy to study the effect of the shape of the hourglass to the pressure. Such investigation would have been considerably harder, if the model was given as a set of difference equations of the traditional form, without any support for reshaping the hourglass.

As indicated by the hourglass example, the use of operators and operator networks helps bringing in the intentionality into a model. The operators capture small, independent computations, whereby the intuition behind an operator is clear. Moreover, the operators can be reused even within the same model. This was illustrated, for instance, with the force operator in the model of the hourglass. The reuse of the operators improves the coherence within the model.

The operators are connected together using the standard functional forward composition. Then, the obtained operator network is also an operator in the formal sense. This particular property is needed for bringing in modularity. It also simplifies the formal treatment of the model, as there are no special cases or exceptions. As illustrated with the hourglass example, an operator network can capture a complex process, such as the creation of a spin shape. Still, the operator network can be used within a model just like any other operator.

It should be noted that, because the operators are formalized as higher-order functions, they are formally analyzable. The operators could be analyzed, for instance, by using Refinement Calculus [1]. Within Refinement Calculus, an operator is technically a relational update, as an operator may not have an inverse image. Furthermore, as the functional composition used for connecting the operator is also part of Refinement Calculus, the behavior of an entire operator network can also be analyzed using Refinement Calculus. This opens many new, exciting possibilities and intriguing topics for future research. In particular, it seems that Refinement Calculus could be a tool for proving the correctness of an operator and its implementation. Then, we could use virtually any programming language to implement the operators to gain better performance, while ensuring with Refinement Calculus that the implementation is correct.

Acknowledgements

We thank the anonymous reviewers for reading and constructive comments. We also thank Keith Baverstock for insightful discussions.

References

[1] R.J.R. Back, J. von Wright, Refinement Calculus: A Systematic Introduction Graduate, Texts in Computer Science, Springer-Verlag, New York, 1998.
[2] J.L. Balino, A.E. Larreteguy, E.F.G. Raso, A general bond graph approach for computational fluid dynamics, Simulation Modelling Practice and Theory 14 (2006) 884–908.
[3] B. Eberhardt, A. Weber, W. Strasser, A fast, flexible, particle-system model for cloth draping, Computer Graphics and Applications 16 (5) (1996) 52–59.
[4] N. Foster, R. Fedkiw, Practical animation of liquids, in: Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques, ACM Press, New York, 2001.
[5] J. Holopainen, M. Rönkkö, Embedding control into a particle system: a case study, in: Proceedings of the Ninth IASTED International Conference on Intelligent Systems and Control, Acta Press, Anaheim, 2006.
[6] J. Holopainen, M. Rönkkö, Plausible motion simulation: inchworm vs. roller, in: Proceedings of the Second International Conference on Computer Graphics Theory and Applications, INSTICC Press, 2007.
[7] J. Holopainen, M. Rönkkö, Roller: the effect of the environment to the emergence of behavior, in: Proceedings of the 2007 International Conference on Intelligent Agent Technology (IAT 2007), IEEE, 2007, pp. 35–38.
[8] J. Kim, J.-C. Yoon, B.-S. Kang, Finite element analysis and modeling of structure with bolted joints, Applied Mathematical Modelling 31 (2007) 895–911.
[9] A. Kubík, Toward a formalization of emergence, Artificial Life 9 (2003) 41–65.
[10] D.F. Martin, G. Estrin, Path length computations on graph models of computations, IEEE Transactions on Computers 18 (6) (1969) 530–536.
[11] B.C. Pierce, Basic Category Theory for Computer Scientists, MIT Press, 1991.
[12] M. Rönkkö, J. Choudhury, Particle system: motion analysis, in: Proceedings of the 16th IASTED International Conference on Applied Simulation and Modeling, Acta Press, 2007.
[13] M. Rönkkö, Previsualization in robotics: an atomic approach, Tech. Rep. A/2003/4, Department of Computer Science, University of Kuopio, 2003.
[14] M. Rönkkö, Previsualization in robotics: an atomic approach, in: Proceedings of the First International Conference on Informatics in Control Automation and Robotics, INSTICC Press, 2004.
[15] M. Rönkkö, An artificial ecosystem: emergent dynamics and life-like properties, Artificial Life 13 (2) (2007) 159–187.
[16] M. Rönkkö, Hybrid systems: modeling and analysis using emergent dynamics, Nonlinear Analysis: Hybrid Systems 1 (2007) 560–576.
[17] M. Rönkkö, Atoms.5: a particle system laboratory, Department of Computer Science, University of Kuopio, <http://www.cs.uku.fi/~mronkko/atoms.html>, 2008.
[18] H. Saadawi, G. Wainer, Modeling physical systems using finite element cell-devs, Simulation Modelling Practice and Theory 15 (2007) 1268–1291.
[19] S. Shi, X. Ye, Z. Dong, Y. Zhang, Real-time simulation of large-scale dynamic river water, Simulation Modelling Practice and Theory 15 (2007) 635–646.
[20] M.R. Spiegel, Mathematical Handbook of Formulas and Tables, Schaum's Outline Series, MgGraw-Hill, New York, 1997.
[21] S.T. Thornton, J.B. Marion, Classical Dynamics of Particles and Systems, fifth ed., Brooks/Cole – Thomson Learning, Belmont, 2004.
[22] D. Tonnesen, Dynamically coupled particle systems for geometric modeling, reconstruction, and animation, Ph.D. Thesis, University of Toronto, Canada, 1998.
[23] X. Tu, D. Terzopoulos, Artificial fishes: physics, locomotion, perception, behaviour, in: Proceedings of 21st Annual Conference on Computer Graphics and Interactive Techniques, ACM Press, 1994.
[24] H.D. Young, R.A. Freedman, University Physics with Modern Physics, 10th ed., Addison-Wesley, San Francisco, 2000.
[25] D.G. Zill, M.R. Cullen, Differential Equations with Boundary-Value Problems, fourth ed., Brooks/Cole Publishing Company, Pacific Grove, 1997.

Please cite this article in press as: M. Rönkkö et al., Beyond particle systems: Operator networks, Simulat. Modell. Pract. (2008), doi:10.1016/j.simpat.2008.05.003