Recent Developments in the Evolution of Morphologies and Controllers for Physically Simulated Creatures*

Tim Taylor
International Centre for Computer Games and Virtual Entertainment (IC CAVE)
University of Abertay Dundee
Bell Street
Dundee DD1 1HG U.K.
http://www.iccave.com
http://www.abertay-dundee.ac.uk
tim.taylor@abertay.ac.uk

Colm Massey
Department of Zoology
Oxford University
South Parks Road
Oxford OX1 3PS U.K.
colm.massey@zoo.ox.ac.uk

Abstract Karl Sims' work [25, 26] on evolving body shapes and controllers for three-dimensional, physically simulated creatures generated wide interest on its publication in 1994. The purpose of this article is threefold: (a) to highlight a spate of recent work by a number of researchers in replicating, and in some cases extending, Sims' results using standard PCs (Sims' original work was done on a Connection Machine CM-5 parallel computer). In particular, a re-implementation of Sims' work by the authors will be described and discussed; (b) to illustrate how off-the-shelf physics engines can be used in this sort of work, and also to highlight some deficiencies of these engines and pitfalls when using them; and (c) to indicate how these recent studies stand in respect to Sims' original work.

Keywords
morphological evolution, brain–body evolution, virtual creature evolution, physics engine

1 Introduction

There can be few readers of this journal who are unaware of Karl Sims' captivating work in evolving virtual creatures in a three-dimensional physically simulated environment [25, 26]. Sims' work was published in 1994, yet despite the considerable interest it generated (and indeed continues to generate) in the artificial life, graphics, and animation communities, it is remarkable how little work has been published since then on replicating and extending his results.

There has been a reasonable amount of work on the evolution of controllers for physically modeled creatures with fixed or parameterized morphologies [1, 5, 11, 13, 18, 23, 24, 28], along with studies employing other adaptive techniques to automate the generation of controllers for fixed morphology creatures [27]. There have also been a number of studies of various kinds of developmental scheme for the evolution of morphology alone (i.e., without controllers) [6, 8, 10]. However, few studies have followed Sims' lead in evolving both the creature's morphology (body shape) and its controller. Dellaert and Beer studied the development of morphology and controllers for simple agents in discrete, two-dimensional grid worlds [7]. Ventrella has published a series of papers describing ongoing experiments on the evolution of morphology and controllers for creatures in continuous, physically modeled environments [29–32]. However, his interest in the ecology and coevolution of large populations of creatures has dictated the use of somewhat simpler morphologies and controllers than those used by Sims, and a simplified physical environment (e.g., Ventrella's more recent models are two-dimensional [30–32]).

* The authors conducted the work described here while employed at MathEngine PLC (http://www.mathengine.com).

Table 1. Recent published work with evolving creatures' morphologies and controllers.

Authors	Physics Engine	Simulation Technology (see text for details)	Comments
Komosinski et al. [14, 15]	Own	Finite element methods	Evolved swimmers and crawlers
Taylor, Massey	MathEngine	General Lagrange multiplier constraint solver with Baumgarte stabilization	Evolved swimmers and crawlers
Lipson, Pollack [19]	Own	Relaxation by energy minimization for quasi-static motion	Created real robots from evolved virtual crawlers
Bongard, Paul [4]	MathEngine	General Lagrange multiplier constraint solver with Baumgarte stabilization	Investigating morphological symmetry and locomotive efficiency of crawlers
Ray [22]	MathEngine	General Lagrange multiplier constraint solver with Baumgarte stabilization	User-guided aesthetic evolution

One of the reasons why Sims' system produced such good results was that he modeled the physics of a three-dimensional environment to a sufficiently accurate degree that objects moved realistically when subjected to forces and torques. Hence the beautiful movements produced by many of his evolved creatures were due just as much to the accurately modeled physical environments as they were to the creatures' individual controllers.

Two possible reasons why more people have not embarked upon similar research are that (a) these sorts of evolutionary system require considerable computational power (Sims ran his original work on a Connection Machine CM-5 parallel computer), and (b) programming the physics of the environment such that it is sufficiently accurate yet also computationally efficient is a considerable challenge.

However, the power of standard PCs has now reached a point where they are capable of running evolutionary systems such as these at a tolerable speed. Also, a number of off-the-shelf physics engines are now available (some of which are free or carry a nominal fee for academic use), liberating artificial life researchers from the chore of programming this component themselves [12, 20, 21]. These two factors have contributed to a welcome increase in research activity in this area within the last couple of years. This work is summarized in Table 1. In Section 2, our own work in re-implementing Sims' system on a PC is described, and our experience of using an off-the-shelf physics engine for this type of work is discussed. In Section 3, the other studies listed in Table 1 are also briefly described. We conclude by relating these new studies to Sims' original work and by suggesting some possible lines of future research.

2 A Re-implementation of Sims' Work Using the MathEngine Physics Engine

We now describe our own work in this area, conducted in 1999 and early 2000. Our project was in the first batch of a recent spate of studies to use MathEngine's commercially available physics engine, a version of which (SDK 1.1) is available free for

academic use [20]. The system was basically a re-implementation of that written by Karl Sims in 1994. The differences were more technical than scientific: (a) We used MathEngine's physics engine, rather than code designed specifically for the application, to provide a realistic physical environment for the creatures to live in; and (b) we performed our experiments on mid-range PCs (mostly 400 MHz Celerons) rather than the Connection Machine parallel computer used by Sims.

One technical difference between the MathEngine physics simulation and that used by Sims is that MathEngine employed a general Lagrange multiplier-based constraint solver [2] with Baumgarte stabilization [3] for modeling joints and contacts, whereas Sims used a combination of Featherstone's reduced coordinate approach [9] for the jointed bodies, with the penalty method for resolving contacts. Although Featherstone's method is fast—O(N) with respect to number of degrees of freedom—and was clearly sufficient to generate a large range of complex character interactions for Sims, constraint-solving approaches can potentially handle more challenging simulations involving multiple contacts and loops in the creatures' morphologies.

Another major difference is in the way we implemented the effectors to control the relative motion of body parts. Sims' effectors worked by applying forces directly between body parts. In contrast, we used orientation-based proportional-derivative (PD) actuators [28] acting on ball-and-socket joints. The input to these controllers was interpreted as a desired orientation for the joint, and the controller would exert a torque on the joined parts to move them toward this orientation. Early experiments indicated that the use of more sophisticated effectors such as these led to the much more rapid evolution of useful movements.[1]

Most other technical details of the system's design were identical to Sims' system, and the reader is therefore referred to his papers for a full description [25, 26]. Here we will only give an outline of the design and will focus instead on the practical problems and issues we encountered when building the system.

Each creature is built up from a genetic description in the form of a nested directed graph. The genetic information describes both the creature's morphology and its control architecture. This representation provides modularity to the mapping from genotype to phenotype and naturally leads to features such as duplication and recursion of body parts. One difference between our work and Sims' is that we used cylinders with hemispherical ends ("sphyls"), rather than cuboids, as the basic body parts, because collision detection can be performed much more efficiently on sphyls. The controller was an "augmented" neural network exactly as described by Sims [26].

A run was started by randomly generating a population of genotypes. Each genotype in turn was translated into a physical creature and then evaluated in a physically simulated environment for its performance at a given task. We used two basic environments, sea and land. The sea environment included a simplistic model of fluid drag (a retarding force was applied to each sphyl, proportional to the square of the component of its velocity perpendicular to its long axis), and the land environment included gravity, a ground plane, and frictional forces for ground contacts.

We used a number of different fitness functions for scoring the success of each creature in its environment, but they all basically rewarded creatures for movement. The definition of the fitness function in fact turned out to be surprisingly difficult to get right, even when we just wanted to reward creatures for moving forward. A straightforward

1 New types of actuators are now available with the latest versions of the MathEngine and Havok dynamics toolkits, which give a much greater level of control and stability to the joint effectors. These are referred to as "force limited velocity constraints" (FLVCs) in the MathEngine Dynamics Toolkit (version 2.0) [20] and "dashpots" in Havok (version 1.3) [12]. Dashpots are PDs but they can handle much stiffer joints (allowing for greater control). FLVCs allow the user to specify an exact relative velocity, which the solver will generate, provided that the force required to achieve it is not in excess of a user-specified limit. This is ideal for modeling realistic virtual muscle responses.

function that simply measured the distance moved by the creature's center of mass over the period of evaluation had a tendency to select for creatures that (in the fluid environment) produced an initial thrust to move away from their starting position but showed no further movement and soon slowed to a halt. Such creatures would have high fitness relative to most of the randomly generated creatures in the early generations and would therefore be selected. However, it is clear that their fitness could be improved if they repeated the thrust movement to swim further and faster. Unfortunately it appeared that in many cases where these "one push" creatures were selected in the initial generations, the population reached an evolutionary impasse (a local optimum in the fitness landscape) and had no easy mutational routes to higher fitness. It is also possible that, in some cases at least, if we had evaluated each creature for a longer period, then there would have been more selection pressure for creatures generating repeated thrust. This highlights a dilemma that we came across many times when experimenting with this system; no matter how long we evaluated each creature, we could never be sure that it would continue to exhibit the same behavior continuously, if simulated for periods longer than the evaluation time. There is a trade-off in evaluation time: A longer time generates more confidence (but no guarantee) that the creatures will perform the selected behaviors continuously, even after the duration of the evaluation period, whereas a shorter time is desirable from a practical point of view of running the genetic algorithm and evaluating hundreds of individuals over tens of generations in a reasonable time.

There are various ways that can be imagined to improve the fitness function to solve these problems. Sims himself experimented with a variety of functions [26]. For example, for swimming, he gave the velocities of a creature during the final phase of the test period a stronger relative weight in the total fitness value. This was apparently successful at rewarding continuing movement over that from a single initial push. In our implementation we found that this kind of function did lead to some improvements, but the population was still liable to get stuck at a local optimum fairly frequently, where one-push creatures were selected in the early generations. Additionally, if the distance moved by the creature was being measured at various time slices throughout the evaluation period (so that these various distances can be weighted and summed to give a final fitness score), we needed to decide whether to score distance moved in *any* direction at any one time slice equally (in which case there was no pressure to evolve creatures that swam in a straight line over the whole evaluation period), or whether to reward only distance moved in one particular direction (and if so, in *which* direction). It was not difficult to make pragmatic decisions about such choices, but the point is that the choice of fitness function even for seemingly straightforward behaviors is not trivial and usually requires considerable experimentation to get right. The function that successfully produces the desired behaviors can often be somewhat more complicated than might initially have been thought.

Even the method used to measure the position of a creature at a given instant was not straightforward. In most runs we used the center of mass. However, in some runs creatures evolved that would initially adopt a compact, folded configuration, then as the evaluation period proceeded they would "unfold" in a particular direction. This unfolding had the effect of shifting the creature's center of mass, thereby increasing its fitness. Again, if this trick was selected in the early generations of a run, it was sometimes difficult for the population to jump out of this local fitness optimum and find continuous movements that would generate higher fitness scores. We experimented with various other ways of measuring distance moved, such as using the distance moved by the body part that had moved least over the duration of the evaluation. The general problem is, no matter what fitness function is used, there often seems to be a way for creatures to score highly on it while not performing the sort of behavior that we, as designers of the function, had hoped for. This problem is not insurmountable; with a

more careful specification of the function all "undesired" behaviors could presumably be detected and given low fitness scores. However, this need for careful design of very specific, detailed fitness functions runs counter to one of our goals of implementing the system, namely, to use it as a method of automatically generating creatures given only a high level specification of the required behavior. Nevertheless, while the use of very specific fitness functions can certainly increase the chances of evolving the desired behaviors in any given run, even using straightforward fitness functions will *sometimes* produce the desired results (as will be demonstrated in the rest of this section), so our goal was at least partially fulfilled.

One way to help prevent the runs getting stuck in local fitness optima would be to maintain the genetic diversity of the population by introducing some kind of incomplete mixing (e.g., by using an island model genetic algorithm [33]). We have performed some initial experiments with incomplete mixing, but not enough at this stage to say how effective this strategy is at solving the problem.

The preceding issues aside, our evolutionary runs proceeded in much the same way as described by Sims [26]. We placed upper limits on the number of body parts a creature could have (this limit was generally in the range of 4–10), and on the number of controller components within each body part (typically, 5–8). We also introduced bias terms (which were parameters of the system) that favored the selection of oscillator components (sine and saw wave generators) over other sorts of controller components when the initial randomly generated population of creatures was being created (or when new controller components were added to existing networks by mutations). These terms could be adjusted to vary the probability of the generation of creatures that displayed oscillatory movements.

Other typical settings for our runs were as follows. The population size was generally 300, and runs lasted for 50–100 generations. The top 20% of genotypes from each generation were transmitted unaltered to the next generation. The remaining 80% of the new generation was created by selecting single parents by tournament selection (with tournament size 2 and a 90% probability of selecting the fitter of the two creatures) and reproducing them asexually (i.e., copying them) with probability 40%, by crossover with another genotype with probability 30%, or by grafting with another genotype with probability 30% (the difference between crossover and grafting is described by Sims [26]). For reproduction by crossover or grafting, the second parent was chosen with a uniform random distribution from the elite 20% of genotypes from the parent population. Mutations were then applied stochastically to the newly generated genotypes (excluding the elite 20%). The probability of mutation was defined in terms of the mean number of mutations per genotype plus the standard deviation (we typically used a value of 2.0 for both of these, rounding all numbers selected from this distribution to the nearest integer, and counting negative numbers as zero).

The integration step for the physics engine was generally 0.05 seconds. We used evaluation periods in the range of 10–50 s of simulated time, which (without visualization) ran somewhat faster than real time. Like Sims, we improved the speed of the system by prematurely aborting the evaluation of creatures that did not perform well. First, any creatures that had no controller connections to their actuators were discarded without evaluation, as they would be unable to generate any movements at all. In addition, the interim fitness of each creature was measured after one-third and one-half of the total evaluation period had elapsed. If, after one-third of the period, the creature had not moved at all then that creature's evaluation was aborted. Similarly, if, after half of the period, it had moved by a distance less than one-fifth[2] of that moved over the total evaluation period by the least fit elite creature of the previous generation (i.e.,

2 This figure was chosen after a small amount of experimentation with different values.

the least fit creature from the previous generation that was transmitted directly to the current generation), then the evaluation was also aborted.

A number of checks were also made to overcome limitations in the simulation software. Despite various attempts to limit the magnitude of the forces applied to joints, creatures would still sometimes evolve whose movements entailed forces and velocities that were too great for the physics engine to resolve at the given size of the integration step. In these cases, the physics engine tended to accumulate numerical errors to a point where the creature unrecoverably exploded (i.e., the constraint solver failed to converge on a solution, and the integrator then generated incorrect velocities, giving the impression that the body parts had blown apart in random directions). The choice of integration step size for the physics engine is clearly another compromise that must be made with this sort of system; a smaller step size produces a more stable simulation than a larger step size but takes longer to run in real time for a given duration of simulated time. The MathEngine SDK does generate some runtime warnings that indicate that this kind of situation is imminent. We kept a tally of the number of such warnings that each creature generated and aborted the simulation of any creature that had generated more than a certain threshold number of them. We also checked whether a creature had actually exploded throughout its evaluation (by checking for high velocities, etc.) and immediately aborted any that had. Note that we were using MathEngine's SDK 1.1 for this work; subsequent experience with using their latest offering (the Dynamics Toolkit 2.0 alpha release) suggests that the software is now much more stable. However, our more recent experiences with using both MathEngine and other physics engines (e.g., Havok [12]) for this sort of work suggest that they all have some weaknesses in stability of simulation in certain situations. Unfortunately, it is in the nature of evolutionary algorithms that such weaknesses will almost inevitably be encountered. A recent review article has tested the stability of the MathEngine, Havok, and Ipion engines in a variety of situations [16, 17]. Although these products are improving, the current situation is that, no matter which physics engine is used, it is likely that a certain number of stability checks of the type just described will be required in any evolutionary system of this kind.

A typical run (i.e., population size 300, 50–100 generations) would take 4–8 hr on a single PC. A large proportion of the evolved swimmers were snake-like creatures of various sorts (examples are shown in Plates 1 and 2). A certain amount of subjective selection was employed to create a variety of different creatures; we would often inspect runs after a small number of generations had passed and only proceed with those in which interesting or unusual creatures (both fairly subjective criteria) seemed to be evolving. A number of different strategies were observed, including the use of various kinds of appendages to push against the water (e.g., Plates 3 and 4), and the adoption of a spiraling, "corkscrew"-type motion (e.g., Plates 5 and 6). We had less time to investigate the evolution of crawlers, but the results we obtained included creatures that used their whole body for locomotion (e.g., Plate 7) and others that employed controlled movement of appendages to push the whole body forward (e.g., Plate 8). Pictures and movies of a wider variety of the evolved creatures are available online at http://computing.tay.ac.uk/timtaylor/demos/mathengine/. These examples demonstrate that the evolutionary process is a useful tool for exploring interesting regions of the vast space of different creature designs describable with the genetic system used; it is a creative machine for generating suitable and interesting forms and behaviors, not limited by the preconceptions of a human designer's imagination.

3 Other Recent Work

Komosinski and colleagues have developed a system called Framsticks [14, 15], in which the morphology of creatures composed of connected "sticks" (modeled as a pair

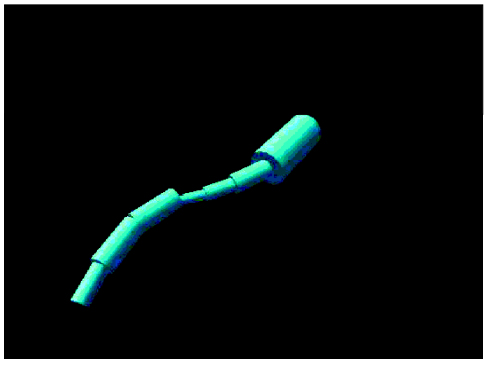

Plate 1. Tadpole.

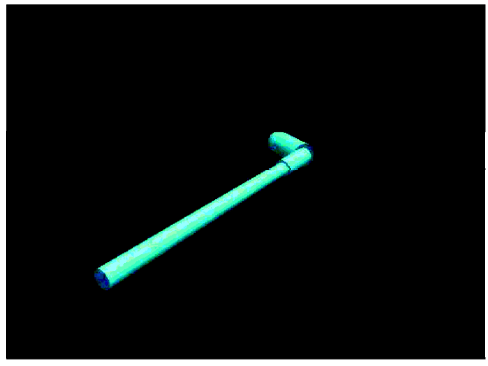

Plate 2. Long-nosed snake.

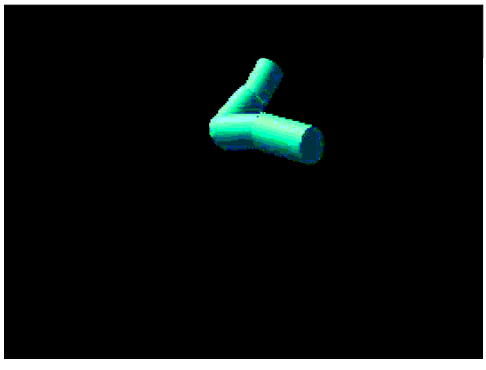

Plate 3. Breast stroke.

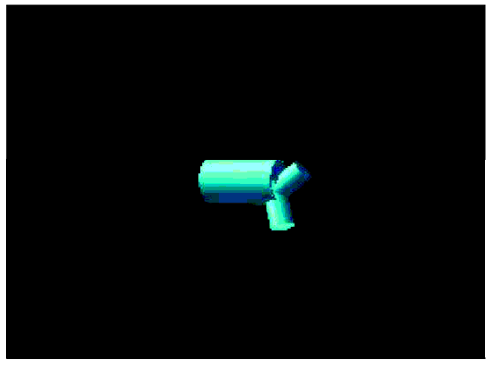

Plate 4. Two-legged kicker.

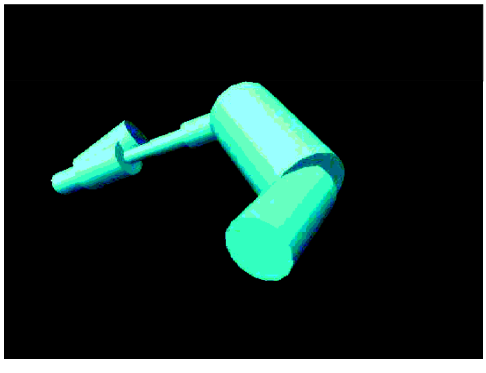

Plate 5. Corkscrew 1.

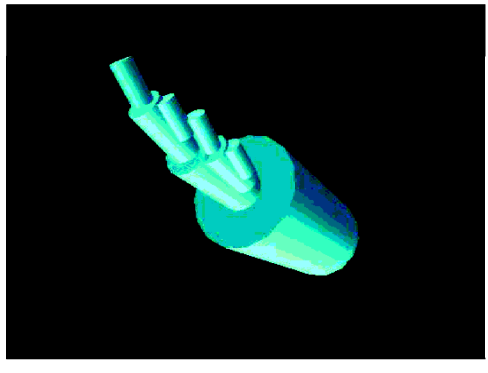

Plate 6. Corkscrew 2.

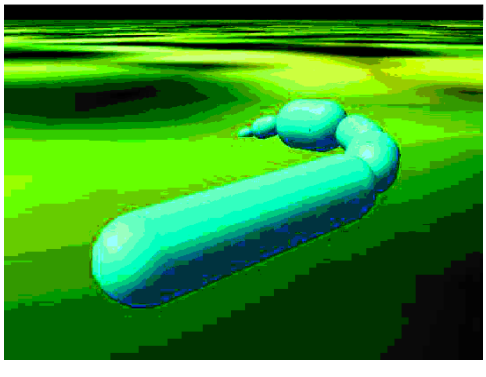

Plate 7. Archer.

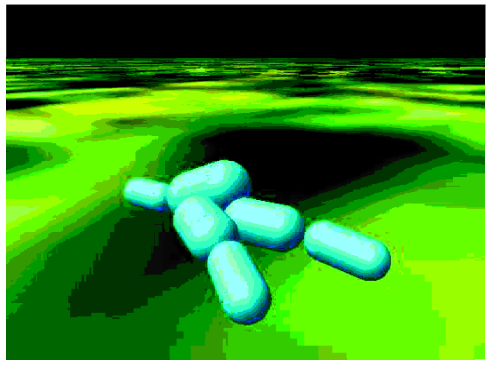

Plate 8. Spider.

of flexibly joined particles using finite element methods), together with their neural network controllers, can evolve. The design of the system is fairly general (e.g., it can handle the simulation of multiple creatures existing in the environment concurrently), although the evolutionary results reported so far concern only selection for simple behaviors such as swimming and walking in single creatures.

Lipson and Pollack have evolved morphologies for creatures composed of collections of bars and linear actuators connected by ball-and-socket joints, along with their neural network controllers [19]. The creatures were modeled using a "quasi-static" simulation method, where each frame is assumed to be statically stable; although it is not a general simulation, it is efficient to implement and can adequately simulate certain kinds of low-momentum motion such as crawling. Their reported results to date have concentrated on evolving crawlers. The major innovation of this work is that Lipson and Pollack have designed the "building blocks" of their creatures in such a way that creatures evolved in simulation can subsequently be automatically manufactured as physical robots using a commercial rapid prototyping ("3D printing") machine. When this process is complete, the linear actuators just have to be snapped into place, and the robot connected to an offline power supply and computer to simulate the controller. It was found that the physical robots manufactured in this way behaved in a qualitatively similar fashion to their simulated counterparts (although quantitative differences emerged due largely to the simplified simulation techniques).

Bongard and Paul studied the evolution of morphologies and controllers for crawling creatures (using MathEngine's SDK 1.1) [4]. Specifically, they looked at the relationship between the level of bilateral symmetry in the evolved creatures and their locomotive efficiency (quantified according to a variety of measures). They found that creatures with a higher degree of bilateral symmetry tended to exhibit greater locomotive efficiency than creatures with less bilateral symmetry. Their study is a good example of how this technology may be used as a tool for investigating fairly general questions about the consequences of evolutionary selection pressures on both morphology and behavior.

Finally, Ray has applied user-guided evolution to a derivative of Sims' system (again using MathEngine's SDK 1.1) [22]. Rather than using an explicit fitness function, the system displays creatures as they are being simulated and allows the user to select whichever ones she prefers to be used as parents for the next generation. In this respect Ray's work is similar to Dawkins' "biomorphs" (although the biomorphs were only static two-dimensional structures) [6]. An interesting aspect of his work is that, rather than trying to avoid the sorts of stability problems with the physics engine as discussed in the previous section, Ray actually embraced them as another source of interesting behavior; some of his evolved creatures were actually selected (by Ray himself) such that their body parts would fly apart at some instances, but they would always eventually restore themselves to their "proper" configurations (i.e., where the joint constraints were satisfied). In other words, he selected creatures that caused some degree of constraint violation when simulated but rejected those that got into situations where these violations were unrecoverable by the physics engine. Ray was more concerned with the aesthetic appeal of the creatures' behaviors rather than their adherence to "real world" physics. His work demonstrates that it can also be interesting (from an aesthetic point of view, and also, perhaps, from a scientific one) to simulate environments that behave somewhat differently from real-world physics.

4 Directions for Future Work

The studies described in the previous section all point to interesting directions for future research. Additionally, the continuing increase in available computing power

greatly expands the possibilities for studying coevolutionary systems (with two or more interacting creatures simulated concurrently) in three-dimensional physically modeled environments. The feature sets of the physics engines mentioned here continue to expand, making it easier to study the evolution of creatures modeled not just as rigid bodies, but in other ways (e.g., soft bodies) as well. The inclusion of a wider variety of actuators, and placing more characteristics of the actuators under genetic control, would create the potential for new types of movement to evolve.

Furthermore, experimentation with lifetime learning techniques, in conjunction with purely genetic approaches, could be rewarding. With the ready availability of low cost, high performance computers and physics engines, it is likely that the recent research interest in the evolution of morphologies and controllers for physically modeled creatures will continue to grow. A workshop on the topic at the recent Seventh International Conference on Artificial Life was well attended and produced a lively discussion (see http://computing.tay.ac.uk/timtaylor/cobb/ for details). As a result of the workshop, a mailing list was set up to promote discussion on the various scientific and technical issues involved in the subject. Instructions for joining this mailing list can be found at http://www.alife.org/mailman/listinfo.cgi/ec2m-list.

Acknowledgments

The authors thank the two anonymous reviewers for their comments and suggestions on a draft of this paper, and Will Wray of MathEngine for comments on our description of the simulation software.

References

1. Arnold, D. (1997). *Evolution of legged locomotion*. Unpublished master's thesis, School of Computing Science, Simon Fraser University, Burnaby, Canada.

2. Baraff, D. (1996). Linear-time dynamics using Lagrange multipliers. In H. Rushmeier (Ed.), *Computer graphics* (SIGGRAPH 96 Proceedings) (pp. 137–146). New York: ACM SIGGRAPH.

3. Baumgarte, J. (1972). Stabilization of constraints and integrals of motion in dynamical systems. *Computer Methods in Applied Mechanics and Engineering, 1*, 1–16.

4. Bongard, J. C., & Paul, C. (2000). Investigating morphological symmetry and locomotive efficiency using virtual embodied evolution. In J.-A. Meyer, A. Berthoz, D. Floreano, H. Roitblat, & S. W. Wilson (Eds.), *From animals to animats: Proceedings of the Sixth International Conference on the Simulation of Adaptive Behavior* (pp. 420–429). Cambridge, MA: MIT Press.

5. Cliff, D., & Miller, G. F. (1996). Co-evolution of pursuit and evasion II: Simulation methods and results. In P. Maes, M. Mataric, J.-A. Meyer, J. Pollack, & S. W. Wilson (Eds.), *From animals to animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior* (pp. 506–515). Cambridge, MA: MIT Press.

6. Dawkins, R. (1986). *The blind watchmaker*. Harlow, UK: Longman.

7. Dellaert, F., & Beer, R. D. (1996). A developmental model for the evolution of complete autonomous agents. In P. Maes, M. Mataric, J.-A. Meyer, J. Pollack, & S. W. Wilson (Eds.), *From animals to animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior* (pp. 393–401). Cambridge, MA: MIT Press.

8. Eggenberger, P. (1997). Evolving morphologies of simulated 3D organisms based on differential gene expression. In P. Husbands & I. Harvey (Eds.), *Proceedings of the Fourth European Conference on Artificial Life* (pp. 205–213). Cambridge, MA: MIT Press.

9. Featherstone, R. (1987). *Robot dynamics algorithms*. Boston: Kluwer.

10. Fleischer, K. (1996). Investigations with a multicellular developmental model. In C. G. Langton & K. Shimohara (Eds.), *Artificial life V: Proceedings of the Fifth International*

Workshop on the Synthesis and Simulation of Living Systems (pp. 229–230). Cambridge, MA: MIT Press.

11. Gritz, L., & Hahn, J. K. (1997). Genetic programming evolution of controllers for 3-D character animation. In J. R. Koza, K. Deb, M. Dorigo, D. B. Fogel, M. Garzon, H. Iba, & R. L. Riolo (Eds.), *Genetic programming 1997: Proceedings of the Second Annual Conference* (pp. 139–146). San Francisco: Morgan Kaufmann.

12. Havok GPI from Telekinesys Research Ltd. http://www.havok.com

13. Ijspeert, A. J. (2000). A 3-D biomechanical model of the salamander. In J.-C. Heudin (Ed.), *Proceedings of the Second International Conference on Virtual Worlds* (pp. 225–234). Heidelberg: Springer.

14. Komosinski, M., & Ulatowski, S. (1999). Framsticks: Towards a simulation of a nature-like world, creatures and evolution. In D. Floreano, J. D. Nicoud, F. Mondada (Eds.), *Advances in artificial life: Proceedings of the Fifth European Conference on Artificial Life* (pp. 261–265). Heidelberg: Springer.

15. Komosinski, M., & Rotaru-Varga, A. (2000). From directed to open-ended evolution in a complex simulation model. In M. A. Bedau, J. S. McCaskill, N. H. Packard, & S. Rasmussen (Eds.), *Artificial life VII: Proceedings of the Seventh International Conference on Artificial Life* (pp. 293–299). Cambridge, MA: MIT Press.

16. Lander, J., & Hecker, C. (2000). Physics engines, part one: The stress tests. *Game Developer*, 7(9), 15–20. (Available online at http://www.gdmag.com)

17. Lander, J., & Hecker, C. (2000). Physics engines, part two: The rest of the story. *Game Developer*, 7(10), 13–18. (Available online at http://www.gdmag.com)

18. Lee, W.-P., Hallam, J., & Lund, H. H. (1996). A Hybrid GP/GA approach to co-evolving controllers and robot bodies to achieve fitness-specific tasks. In *Proceedings of the 1996 IEEE Conference on Evolutionary Computation* (pp. 384–389). Piscataway, NJ: IEEE Press.

19. Lipson, H., & Pollack, J. B. (2000). Automatic design and manufacture of robotic lifeforms. *Nature, 406*, 974–978.

20. MathEngine Dynamics Toolkit 2.0 (and SDK 1.1) from MathEngine PLC, http://www.mathengine.com

21. McMillan, S. DynaMechs (Version 3.0) [computer software]. http://www.sourceforge.lkams.kernel.org/dynamechs/

22. Ray, T. S. (2000). Aesthetically evolved virtual pets. In C. C. Maley & E. Boudreau (Eds.), *Artificial Life 7 Workshop Proceedings* (pp. 158–161). Available online at http://www.hip.atr.co.jp/~ray/pubs/alife7a/

23. Reeve, R. (1999). *Generating walking behaviors in legged robots*. Unpublished doctoral dissertation, Division of Informatics, University of Edinburgh, Scotland.

24. Reil, T. (1999). *Artificial evolution of neural controllers in a real-time physics environment*. Unpublished master's thesis, University of Sussex, England.

25. Sims, K. (1994). Evolving 3D morphology and behavior by competition. In R. Brooks & P. Maes (Eds.), *Artificial life IV: Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems* (pp. 28–39). Cambridge, MA: MIT Press.

26. Sims, K. (1994). Evolving virtual creatures. In A. Glassner (Ed.), *Computer graphics* (SIGGRAPH 94 Proceedings) (pp. 15–22). New York: ACM SIGGRAPH.

27. Terzopoulos, D., Tu, X., & Grzeszczuk, R. (1994). Artificial fishes: Autonomous locomotion, perception, behavior, and learning in a simulated physical world. *Artificial Life, 1*, 327–351.

28. Van de Panne, M., & Fiume, E. (1993). Sensor-actuator networks. In J. T. Kajiya (Ed.), *Computer graphics* (SIGGRAPH 93 Proceedings) (pp. 335–342). New York: ACM SIGGRAPH.

29. Ventrella, J. (1994). Explorations in the emergence of morphology and locomotion behavior in animated characters. In R. Brooks & P. Maes (Eds.), *Artificial life IV: Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living*

Systems (pp. 436–441). Cambridge, MA: MIT Press.

30. Ventrella, J. (1996). Sexual swimmers: Emergent morphology and locomotion without a fitness function. In P. Maes, M. Mataric, J.-A. Meyer, J. Pollack, & S. W. Wilson (Eds.), *From animals to animats 4: Proceedings of the Fourth International Conference on the Simulation of Adaptive Behavior* (pp. 484–493). Cambridge, MA: MIT Press.

31. Ventrella, J. (1998). Attractiveness vs. efficiency: How mate preference affects locomotion in the evolution of artificial swimming organisms. In C. Adami, R. K. Belew, H. Kitano, & C. E. Taylor (Eds.), *Artificial life VI: Proceedings of the Sixth International Conference on Artificial Life* (pp. 178–186). Cambridge, MA: MIT Press.

32. Ventrella, J. (1999). Animated artificial life. In J.-C. Heudin (Ed.), *Virtual worlds: Synthetic universes, digital life and complexity* (pp. 67–94). New York: Perseus Books.

33. Whitley, D., Rana, S., & Heckendorn, R. B. (1999). Exploiting separability in search: The island model genetic algorithm. *Journal of Computing and Information Technology* 7(1), 33–47.