DOI: 10.1111/cgf.142661
EUROGRAPHICS 2021
H. Rushmeier and K. Bühler
(Guest Editors)

Volume 40 (2021), Number 2
STAR – State of The Art Report

Virtual Creature Morphology - A Review

G. Lai[1,2], F.F. Leymarie[1], W. Latham[1], T. Arita[2] and R. Suzuki[2]

[1]Goldsmiths, University of London, London, United Kingdom
[2]Graduate School of Informatics, Nagoya University, Nagoya, Japan

Abstract
We present a review of methods for procedurally generating the morphology of virtual creatures. We include a range of methods, with the main groups being from ALife over art to video games. Even though at times these groups overlap, for clarity we have kept this distinction. By including the word virtual, we mean that we focus on methods for simulation in silico, and not physical robots. We also include a historical perspective, with information on methods such as cellular automata, L-systems and a focus on earlier pioneers in the field.

CCS Concepts
• ***Applied computing*** → *Computational biology;* • ***Hardware*** → ***Biology-related information processing;*** • ***Software and its engineering*** → ***Interactive games;*** • ***Computing methodologies*** → ***Artificial life;***

1. Introduction

In this paper, we present a review of methods for the procedural generation of virtual creatures. As procedural content generation methods gain popularity, we now realise that not only humans can initiate creation. Research fields such as mixed-initiative co-creative interfaces are introducing computational agents as full creators and collaborators. This review is uniting a number of fields that would typically be disparate; ALife, games and the arts. When artists design and model creatures, they are often inspired by real-life animals and anatomy. While a human artist studies anatomy and existing works to get better at their job [MS07], what will the computer study to get better at understanding creature morphology? Increasingly with the rising popularity of machine learning, in other fields, the answer has been that the computer studies what the human studies. However, except for the works of Toor & Bertsch [TB20] described later in this paper, which isn't an approach that has gained momentum when it comes to modelling creature morphology. Instead, the machine can learn about anatomy through physical simulation and evolving forms. This approach started with the works of Karl Sims [Sim94] and has remained popular to this day. This paper will describe many such methods in detail.
Computer-generated creatures are not only interesting subjects of study in their own right [Kri19, Hay99]. They can teach animators and creature designers how to become better at their craft, and represent a fun and intriguing way to learn about evolution, as the anecdotes of Lehman et al. [L*20] and the annual virtual creatures competition [CGKR21] bear witness to. Virtual creatures represent a field with far-ranging applications, from robotics, arts and video games over sorting algorithms [Hil90] to the simulation of ecosystems for learning about evolutionary issues such as what comes first; behaviour or morphology? [IPSA13] Similarly, the methods for generating artificial life forms are plentiful; cellular automata, L-systems, rigid bodies, finite elements, gene regulatory networks and more. We will cover all these.
Making "heads or tails" of such a huge area has been difficult. Simply organising this review chronologically and listing work after work tells us little of each method's purpose and the context under which it was developed. Instead, we aimed to learn under which circumstances each method is useful and how it relates to others. Therefore, we have sought to separate the works based on the primary methods and have divided this paper into sections on L-systems, cellular automata, gene regulatory networks, and so forth. As we have researched in preparation for this survey, Artificial life (ALife), the arts (Art) and video games (Games) have emerged as the major fields in which methods for generating the morphology of virtual creatures are developed and used. ALife methods present a process-oriented, computing centric field. On the other hand, the methods presented for PCG in video games represent a more goal-oriented approach, with the algorithm being led by the designer's or player's ideas. Artistic methods often exist somewhere in between those two approaches. Unfortunately, with many video games, even where it is clear that they use procedural methods — in whole or in part — for generating their creatures, often very little information has been made publicly available about the methods used. The main exception here is Spore [Max08], described in section 11. The only other game described in-depth in this review is Darwin's Avatars [LR15] (section 3.2.1). Thus, with only two games, instead of having a separate chapter dedicated to those, we have described them, where relevant.

1.1. Framework

Growing virtual creatures and making them come to life is a vast and complex undertaking involving many disciplines, such as generating the morphology, making the morphology look believable for a specific context, animating the creature and simulating behaviours. For this review, we are focusing solely on procedural generation of virtual creature morphology.

It can be difficult to conceptually separate behaviour and morphology. In fact, Darwin [Dar08] teaches us that evolution of behaviour and morphology are closely interlinked. However, it is often helpful to try to analyse sub-parts separately before looking at the overall connections and interplay between those parts. Much work has been done on the behavioural side with research on procedurally- or computationally-assisted animation [AvdP16, ACBS*17, BC05, CM16, FMN94, GCH*16, GP12, GvdPvdS13, HKS17, KP11, LPKL14, LPY16, PBvdP16, PBYV17, ZSKS18], while relatively little work has focused on morphology. Therefore, the main focus of this study is on the analysis of algorithms for the design of the morphology of virtual creatures. Where relevant and intrinsically interlinked with morphology, this study also describes the behavioural side of an algorithm.

Neural networks are often used for simulating behaviour [Sim94, Ray01, SK03, SCH04, LLD07, ASA15, AJI*16, ACBS*17, CEA95, CEA07, PJ08, MC05, Mic08, Kra08, Krč07], as well as morphology. In particular, Compositional Pattern-Producing Networks [Sta07]) (CPPNs) have often been used for modelling morphology [HL10, CMCL13, C*16, CCGS*18]. Works that solely use neural networks for modelling behaviour are not covered in this study, but works that use them for modelling morphology are. Section 7 gives an introduction to this approach.

While there has been a lot of work on evolving physical robots, our work is focused squarely on virtual creatures. However, the design of physical robots often happens through evolving virtual robots in simulators (e.g. Veenstra et al. [VFRS17]). In our review, we include such works in robotics when the main focus remains on the virtual simulation.

Humans are the only type of creatures deliberately left out of this survey, as it quite a considerable area that deserves its own treatment and has to deal with its own particular set of problems, such as the Uncanny Valley [MMK12]. Except for not considering humans, we have not discriminated against other types of creatures and have included animals such as fish [Tu96, TT00].

1.2. Semantics

Some papers [LFM13, LFM14a, LR15, LFMR15] use the term *Evolutionary Virtual Creature* (EVC) for the generated creatures. Others use the term *Animat* [TR97, JW12, JKDW13b]. While animats do not have to be evolutionarily generated, EVCs and animats both refer to animated creatures or animals. In the context of this review, we will use the term animat as it is a slightly broader term than EVC. A single paper [JPH19] refer to their creations not as creatures, but as inspired by *Braitenberg Vehicles* [Bra84]: i.e. virtual agents moving around in an environment using sensors. With a few exceptions, all of the EVCs and animats in this review can be seen as generalised Braitenberg Vehicles as most move around a virtual environment using sensors. The words *virtual* and *simulated* can often be interchanged and applied to a non-physical entity whose movements and evolution is calculated by a software algorithm inside a computer.

1.3. Previous Surveys on Virtual Creatures

To the best of the authors' knowledge, there has been no recent survey focusing on the morphology of virtual creatures, although some related work has appeared. Sythiya et al.'s survey of nature-inspired "intelligent" robots [SCBN17] includes papers on morphological evolution, however, its focus is mainly on physical robotics. Additionally, the paper gives a good introduction to concepts often used in evolutionary computation, such as genetic algorithms, neural networks and L-systems. Similarly, Pollack et al. [PHLF03] describe a series of works that include morphological evolution, but their focus is on a chronological historical description of the work of the lab the authors belong to. We have only deemed the last of those lab works relevant for this review. It is described more in-depth in section 4 on L-systems. Recently Shah et al. [SYK*20] published a survey on methods for shape-changing robots. They include simulation methods, and their focus is on applications that translate to physical robots.

1.4. Overview

This paper is divided into six conceptual parts. After the Early Works section on generating creature morphology (section 2), comes the main part with a review of mainly ALife methods (section 3-11), followed by a review of artistic methods (section 12). There is an overview of simulations and ecosystems (section 13), and then finally before the conclusion (section 15), comes section 14 reviewing methods that compare the efficiency of different genotype representations.

2. Early Works

Of course, it is impossible to talk about the morphology of creatures without mentioning the books On the Origin of Species by Darwin [Dar08] and On Growth and Form by D'Arcy Thompson [Tho42]. While both works were written before the advent of digital simulation, both have been hugely influential. Darwin laid the foundations of the evolutionary theory that so many works in the review base themselves on. D'Arcy Thompson instead focused on the connection between form and pressure from physical forces, coming up with explanations for a multitude of the spectacular forms we see in nature. While the works of Darwin and D'Arcy Thompson are at times juxtaposed against each other, it is more accurate to say they work at two different levels of abstraction, or as Thompson puts it when paraphrasing Aristotle, "the house is there that men may live in it; but it is also there because the builders have laid one stone upon another." [Tho42]

Grounding ourselves in an age of computers, we'll start with a thorough look at cellular automata and L-systems before moving on to the landmark works of Dawkins [Daw86], Sims [Sim94], and Latham & Todd [TL92] that caused such an explosion of interest in the field.

2.1. Cellular Automata

Conway's cellular automaton Game of Life (GoL) from 1970 [Gar70] has an evocative life-like look, that when simulated, reminds us of cells in a petri dish. Depending on the start configuration of the cells, you can have simulations where it looks like lifeforms are walking across the screen (gliders) and others such interesting forms [Wai74]. This evocative look comes from obeying four simple rules that determine if a cell is dead or alive. Though Conway is the person most often associated with cellular automata, John Von Neumann [VN66] and Stanislaw Ulam invented the concept while working at the Los Alamos National Laboratory. Von Neumann's original notes [VN66] are from 1948. Nils Barricelli [Fog06] was one of the first researchers to bridge cellular automata to ALife, as his evolutionary simulations and organic-looking grid-based setups were clearly reminiscent of cellular automata and seemed to foreshadow Conway's GoL.
In 1983 Stephen Wolfram [Wol83] published a treatise, focusing largely on 1D cellular automata. Wolfram describes cellular automata through the bit pattern they produce. For example, the rules for rule 30 produce the base-2 number for 30, as seen in table 1. We see that mathematically this rule is the same as $left\ cell$ XOR ($middle\ cell$ OR $right\ cell$). While many rules quickly produce regular repetitive patterns, rule 30 is of particular interest because its output is chaotic looking and difficult to predict.

111	**110**	**101**	**100**	**011**	**010**	**001**	**000**
0	0	0	1	1	1	1	0

Table 1: *Output for Wolfram's rule 30 [Wol83]*

2.2. L-systems

L-systems were first formalised by Aristid Lindenmayer [Lin68] in 1968 as a method for simulating plant cells and modelling cellular growth, but as we will see later, L-systems have also been used for modelling virtual creatures.
An L-system can be described by the tuple $\mathbf{G} = (\mathbf{V}, \omega, P)$, where $\mathbf{V}$ is the alphabet, ω is the starting symbol(s), and P is the set of possible productions.
In computer graphics, the easiest way of visualising L-systems is to imagine a turtle that draws lines between symbols, with a set length between them. Additionally, some symbols will direct the turtle to turn a certain number of degrees. Imagine, for example, how this can be used for drawing tree-like structures. Using a turtle to draw and visually represent L-system was first introduced by Prusinkiewicz [Pru86], who also introduced square brackets into the grammar as a way of representing pushing [and popping] a state on or off a stack of states. Especially for drawing operations, a stack of states simplifies things a lot, as the turtle doesn't have to be turned and directed towards a previous location. Works on virtual creatures using L-systems will be further elaborated on, in section 4.
Chomsky discovered Chomsky Hierarchies in 1956 [Cho56] some years before L-systems, and the two are closely related. One of the main differences between the two is that while the latter are evaluated in parallel, Chomsky Hierarchies are evaluated sequentially [McC93].

2.3. Yoichiro Kawaguchi

GROWTH (Growth Rationale Object Work Theorem) is a program for automatically generating spiral organic structure. Kawaguchi [Kaw82] describes a program that can generate spiral structures such as horns, shells, some plants, claws and tusks. Kawaguchi's observation, which is similar to Latham & Todd's later approach (described in section 2.5), focuses on how shapes are generated in nature rather than on exact geometric measurements. The result is very organic looking shapes. Kawaguchi focuses on local changes and how many local changes can result in an overall global look. The basic rule that Kawaguchi describes is:

1. Define a trunk and derived branches with diameters D_0,D_i, and heights H_0, H_i
2. The top and bottom of a base are always parallel.
3. Two adjacent trunks are always connected by at least a single point, called a knot K.
4. Adjacent trunks connected by a knot, can be have an angle of χ between them at K.
5. A branch always derived from a knot.
6. The diameter of a branch D_i is affected by the height (angle) of the knot.

2.4. Biomorphs

In 1986, Richard Dawkins published his influential book The Blind Watchmaker [Daw86], which describes a computer program, Watchmaker, that simulates and evolves shapes called *biomorphs*. Dawkins was inspired by the zoologist and painter Desmond Morris, who called the figures in his paintings Biomorphs. Watchmaker's interface works by having the user select one form out of a number of options, and then that form is chosen as a parent for the next generation of forms that the user can again select from, and so on. The original source code is still available [Daw]. The screenshots in Figure 1 are taken from a re-implementation in Java, also available at the same location. The British Broadcast Corporation (BBC) has a brief film clip on their website, featuring Dawkins explaining biomorphs and showing the interface for evolving forms [BD91]. Dawkins' biomorphs have nine genes.

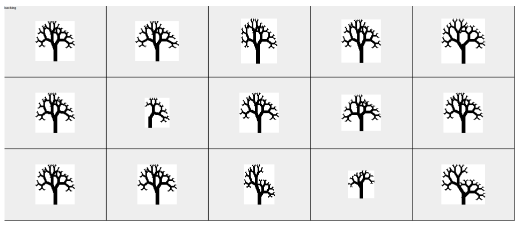

Figure 1: *Screenshot from a java port of Richard Dawkin's original Watchmaker application [Daw]*

Dawkins doesn't use crossover for his evolutionary algorithm, only mutation, so each biomorph has only a single parent. Offspring differ from their parent only in that one of the nine genes are modified by a +1 or -1.
Nearly all later work, including landmark works such as Sims [Sim94] and Todd & Latham [TL92] can be seen as taking

inspiration from Dawkin's ideas, though Dawkins has faced recent criticism. In a review of Dawkin's book *Brief Candle in the Dark: My Life in Science* [Daw15], Comfort critiques the book [Com15]: "In his simulations, life is utterly determined by genes, which specify developmental rules and fixed traits such as colour. The more lifelike his digital animals ("biomorphs") become, the more persuaded he is that real genes work in roughly the same way." and "Genes cooperate, evolving together as units to produce traits. Many researchers continue to find selfish DNA a productive idea, but taking the longer view, the selfish gene per se is looking increasingly like a twentieth-century construct.". While dynamic structures with multiple interdependent *genes* such as neural networks are much more in use today, a lot of work still uses either a hybrid model (some selfish-genes, some dynamic interdependence) and some just still use selfish-genes, as will become more apparent further down in this survey.

2.5. William Latham & Stephen Todd

Haggerty's interview with Latham & Todd [Hag91] describes the origins of Mutator — the pair's groundbreaking work [TL92]. The roots of Mutator go back to a 10x2 meter rule-based "Form Synth " drawing created by the artist William Latham at The Royal College of Art in 1985, who was inspired by evolutionary techniques [Lat89]. Starting with a basic shape —- cone, cube, cylinder, sphere or torus — and applying a series of simple transformations to them—beak, bulge, scoop, stretch, twist and unite — the results are organic-looking composite shapes, whose evolution are mapped out on the enormous drawing. Later, Latham used a solid modelling program, the Winchester Solid Modeler (Winsom [B*89]) at the IBM Research Center in Winchester, before teaming up with Stephen Todd to create the Mutator program [TL91]. In Mutator, the human user performs the role of the fitness function—for each generation, the program presents nine forms to the user, who then chooses which one becomes the next generation's parent.
Latham & Todd describe this and more in the book Evolutionary Art and Computers [TL92], which came out a year after that interview. In the context of this survey, the most interesting part of their work is the Mutator application. It generates organic biological inspired forms. Together with its companion programs Life Cycle and Director, animated films illustrating the life cycle; birth, growth and death of biologically-looking forms can be made. Mutator, which is the system of operators used to create the film *The Evolution of Forms*, is inspired by biomorphs [Daw86] and *Form Synth*. Form Synth was a system of rules used to generate the forms. Part of a painting drawn using the Form Synth rules can be seen in Figure 2. Form Build was an initial attempt at making a digital version of Form Synth. In Form Build, constructed shapes consists of two components; an attachment point on the original shape and a new geometric shape — cones, cylinders, cubes and spheres — which will expand the original shape at the attachment point. Unlike Form Synth, Form Build only supported a single operation — add. Form Build was later replaced by the more complete Form Grow with a richer set of 3D transforms such as twist, bend, stack, branch, cage and add. Built on the ideas of Form Grow and Richard Dawkin's biomorphs [Daw], Mutator uses evolutionary algorithms to create vastly more organic-looking shapes. Initially using an interface akin to Watchmaker [Daw], but with crossover with multiple par-

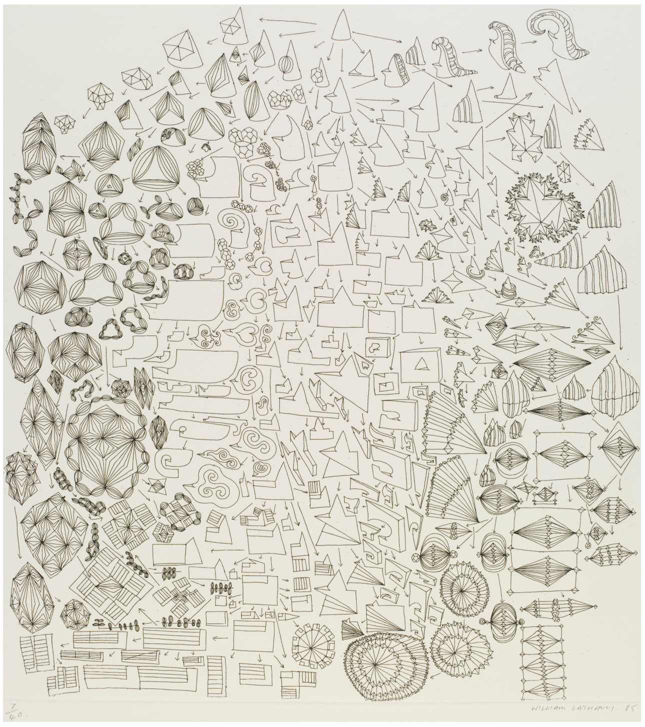
Figure 2: *Drawing from 1985 illustrating the rules of Form Synth [Lat89]*

ents, later versions tried to give more technical insight into the process of selection by showing parts of or the whole of the evolutionary tree. Another interesting initiative they tried is to use simulated annealing [KGV83] to make the jump in gene space smaller over time as the user searches for interesting shapes. Fast-moving genes in a selection are seen as recessive genes that the user is not happy with. Figure 3 shows a selection of shapes generated with Mutator. Animations created with Mutator and its companion applications

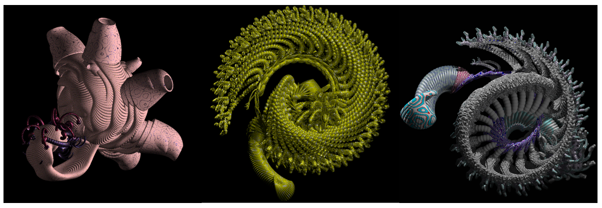
Figure 3: *Shapes generated with Mutator*

look particularly organic. Animations to simulate birth, growth and death, are created by interpolating a bank of genotypes. Both Mutator and Form Grow are based on the Extensible Solid Model Editor (ESME) [Tod21], which was an extensible language for creating CSG shapes used at the IBM Research Center in Winchester.

2.6. Karl Sims

In this seminal work by Sims [Sim94], the author uses evolutionary algorithms to generate simple 3D articulated animal forms or *animats*. Note that Sims did not himself use the word animat which was popularised three years later [MW90]. In Sims' work, evolving the animats consisted of five parts: (1) a fitness function, (2) a directed graph where each node represents an oriented box (limb), and each edge in the graph represents a joint, (3) local neural networks inside each joint, driving the movement of the joint, (4) a global neural network, the output of which is used as parameters to local neural networks situated in the joints and (5) sensors to drive additional input to the neural networks.

A directed node graph represents the genotype of an animat. This graph can be transformed into a phenotype by mapping the nodes into oriented boxes (OBs) and the edges into joints between the OBs. Each node in the graph is a data-structure describing attributes such as the dimensions, joint type, connections to other nodes, and a network of neurons, in what is essentially a Compositional Pattern Producing Network (CPPN) (Stanley [Sta07]). Sims [Sim94] does not use the term CPPN, as the term wasn't coined until 13 years later by Stanley, though neither does he use the term artificial neural network (ANN) or just neural network, instead he uses the word *Neuron*. Stanley et al. [Sta07] state that ANNs and CPPNs function almost the same way, except that CPPNs have a greater range of choice for the node activation functions. However, the authors still insist that CPPNs should not be seen as a specialisation of ANNs, as the primary purpose of CPPNs is to describe mathematical functions that define shape and not to emulate the brain as is the purpose of ANNs.

In Sims' model [Sim94], a single recursive graph of the genotype describes both brain and morphology. In case of recursiveness, the genotype for the body will result in a number of body parts given by the recursive level. Each part of the phenotype has an inner ANN that controls its movement. Additionally, there is a global ANN that is connected to the local ANNs. The global ANN is copied into the phenotype only once. It works like the local networks, except that it can take its input from anywhere, including sensors and local networks, and can apply its output to local networks as well as any joints.

As in Sims' work, the fitness function in nearly all the papers in this review measures the movement of the centre of mass over the simulated time. He evolved different movement types such as walking, jumping, and removed gravity and added viscosity to simulate swimming. Sims also experimented with fitness functions that would reward animats for moving towards and following a light sensor [Sim94].

Sims' master thesis [Sim87] published seven years earlier than his landmark work, describes an interactive creature editor which works by letting the designer create a 2D network of nodes connected by lines. The network is automatically extracted to represent a 3D creature. The editor incorporates logic to automatically identify body parts such as the head, tail, body and legs. The program can automatically generate different forms of gait for the creature, and the creature is able to move across a complex landscape with detection for impossible footholds. For a walking gait, the movement is entirely kinematic with solvers for forwards- and inverse kinematics. For hopping the leg placement is kinematic while the integration of the body physics is solved dynamically.

3. Later Works inspired by Sims

Sims' creations have inspired a plethora of research. Some of which we will describe in this section. Ray's [Ray01] art-inspired creature *zoo* is in most ways a faithful re-implementation and reproduction of Sims' methods [Sim94]. An animat's genotype is initialized by generating a random number of nodes, representing box limbs with random sizes and colours. Each limb has a random number of outgoing connections to other boxes, though boxes can only have one parent. Ray added a number of additional sensors such as time, colour, velocity, etc., and his implementation allows for self-intersection (which is ignored by the physics simulation). Though the author implemented a fitness function based on following another object, many of the animats described in the paper are found through random selection and the author's personal affinity for and desire to follow the fitness of an object. The author evolves some interesting shapes, not usually highlighted in papers reproducing Sims' work.

An interesting note is that Ray chose MathEngine [PLC03] for physics integration and ended up with the common problem of numerical instabilities. The forces in the joints simply ended up becoming too big to keep the object together and would finally explode, causing the object to travel too far. Ray turned math into art by selecting the exploding creatures for breeding so that the simulation time before explosion would increase, in the end getting to a point where the joints would visibly push apart before being pulled back in place by the mathematical simulation.

While Lassabe et al. [LLD07] modelled their creatures' morphology directly on Sims [Sim94], they choose not to explicitly support symmetry and also created a new behaviour controller based on classifiers. The authors successfully tested their work in seven different situations; walking on a flat plane in any direction, walking in a specific direction on a flat floor, navigating trenches, climbing stairs, walking on a heightfield, skating, and cooperating via pushing a block. Lassabe et al. used the Breve simulation environment engine [Kle03]. Unfortunately, the authors did not make a direct comparison between their classifier system and a system based on Sims's neural networks. This could have given us a better basis for judging their new system. Regarding morphology, the creatures presented in their paper seem to have natural symmetry, though the authors have not explicitly programmed it (unlike Sims [Sim94]).

3.1. Flying

Shim et al. [SK03, SCH04] expanded Sims' work by producing a series of papers on generating flying creatures. To this end, Shim et al. used a restricted and modified version of Sims' geno- and pheno-types. Only two types of joints are used; hinge and ball-socket. Hinges are always used between wing segments, while a ball-socket is used between the wing-root and the body-root. Three types of sensors are used: the first one measures the angle of a joint, the second is a joint-stop indicating whether a joint has reached its limit, and the third is a gyroscope. Each creature has one gyroscope, which is attached to the wing root. The genome graph is essentially limited to a limited type of directed graph where, except for the wing-root, each node has exactly one child and one parent. The wing root has two child connections, but they point to the same node, thereby guaranteeing symmetry when unfolding the genotype into a phenotype. For the neural networks, each neuron is limited to a maximum of three inputs. We note that the final phenotype has wing bones rather than proper wings, as these are obtained by spreading a thin film between each wing bone. Force and torque on each bone (as a cylinder) is then calculated by subdividing each wing into smaller triangles, calculating normal and tangential velocity at the vertex points, and then for each cylinder summing up force and torque for each connected sub-triangle. The complete formula is given in Shim et al. [SK03]. In a later work, the authors improve the simulation by creating flyers that can follow a pattern [SCH04]. This is achieved by adding a tail and modifying the basic algorithm to add a second stage, where an ANN is attached and trained to let the creature follow a target. The physics integration and simulation is based on Open Dynamics [Smi]. Neither of Shim et al.'s works [SK03,SCH04] take external forces such as wind into account.

3.2. Encapsulation, Syllabus & Pandemonium

Lessin et al. published a series of works [LFM13, LFM14a, LFM14b,LFMR15] on a method they call ESP (Encapsulation, Syllabus & Pandemonium). They also coined the use of the term EVC (as explained in section 1.2 we instead use the broader term animat for this review). The purpose of their work is to improve the behavioural complexity of animats. As they rightfully point out, not much work has been done since Sims [Sim94] on increasing the behavioural complexity of animats. Like the works described in section 3 most of the structure for the genotype and phenotype done by Lessin et al. builds on Karl Sims' original work [Sim94], with a few minor changes to the morphology and some more radical changes to the algorithm.

For the morphology, Lessin et al. [LFM13] adds the ability to develop photoreceptors as well as muscles. Sims [Sim94] also describes work where animats are able to follow a light source, but unlike this work, his photoreceptors are fixed to exactly three input signals. In the work of Lessin et al., animats can have any number of photosensors, and they can be evolved on any part of any segment. Additionally, instead of applying forces on joints, the ESP method introduces *muscles* which are essentially springs, where the endpoints are attached to two different segments. Just like joints, they connect parent and child segments. Muscles are modelled using a *distance joint* from the PhysX SDK [Cor], where the parameters are set in such a way that they act as a spring. Muscles have a binary input (0 or 1) that comes from the brain, telling the simulation whether the muscle is activated or not, as well as an output to the brain equal to the length of the spring. While the simulation does have joints of different types between a child and parent body segment, unlike Sims' [Sim94] work, joint limits are not evolved as part of the evolutionary process.

While the morphological changes described above are interesting in themselves, a valuable contribution comes from the realisation that the development of morphology and behaviour can happen in separate steps. The basic idea is that different behaviours (or skills as the authors call them) can be evolved even as the basic morphology of segments and joints is kept fixed. Ito et al. [IPSA13] further discuss the development dynamics between morphology and behaviour (details in § 13).

ESP consists of three elements; encapsulation, syllabus, and pandemonium. The syllabus for an animat is a directed graph of skills. Once one of the skills in the syllabus has been learned, *encapsulation* is used for storing the neural network that models a particular skill. The network is modified slightly so that skills can be selectively activated. Finally, a mechanic is added to prioritise between those behaviours where it is not always obvious which has priority. It is up to the designer of the syllabus graph to show who has priority in pandemonium.

Lessin et al. [LFM14a] remove the limitation that the base morphology (segments and joints) has to stop evolving after the first skill has been learned. Instead, if a morphological change reduces the fitness of an already learned skill beyond a preset limit, then the individual's fitness is reduced to 0 (the worst possible fitness in this simulation). If an individual manages to evolve all needed skills without having its fitness set to 0, then it is passed through another learning cycle, where segments and joints are locked, to make sure that earlier learned skills are re-adapted to morphological changes that might have happened during the learning of later skills. Lessin et al. [LFM14b] go into more detail on how a combination of muscles (springs) and joints with no angle limits have replaced Sims' [Sim94] joints with evolved parameters. Lessin et al. summarise their work on animats [LFMR15] by comparing fast ESP with re-tests [LFM13] and general ESP [LFM14a], as well as examples of general fitness growth graphs for *Turn Left* which is a skill that has to be learned from scratch, versus *Turn To Light*, which is a compositional skill, that relies on earlier skills such as *Turn Left*. The fitness for *Turn To Light* grows faster and more regularly than for *Turn Left*. The hierarchical skill development can be cast as an exaptation process. More work on exaptation has been done by Asakura et al. [ASA15], and Corucci et al. [CCGS*18] (see § 8.4).

3.2.1. Darwin's Avatars [LR15]

Darwin's Avatars [LR15] by Lessin & Risi is a game based on the work of the same team that worked on ESP. It is a two-player game, where each player controls their own animat and competes to reach the end of a straight piece of racetrack first. Additionally, the goal must be reached within a set time. If one or both players fail to reach the end of the track, the player who got closest to the goal wins. The time limit depends on the type of creature used for the race. The game offers three kinds of different creatures, though both players must be using the same type.

The players' avatar has been evolved through the ESP method (§3.2). For the computer player, both the morphology and the behaviour is used. In contrast, for the human player, only the creature's morphology is kept as the human makes the avatar move via a keyboard interface, in a way reminiscent of QWOP [Fod08] and Incredipede [Gam12]. The difference is that in Incredipede, there are only two keys to control a creature, while in Darwin's Avatars, each key is connected to independent muscles (though they can be mapped to overlap). Though this scheme complicates the controls, it gives the user the same amount of control as the "original" neural-network controlled brain. When testing the game, if a computer-controlled player was present, it would nearly always win, though the authors indicate that humans found it entertaining to compete against another human.

4. L-systems

Historically in computer graphics, Lindenmayer systems (L-systems) have typically been used for modelling plants. The foundations of L-systems are described in the Early Works section.
Hornby et al. [HLP01] used a combination of L-systems and evolutionary algorithms (EAs) for generating virtual creatures that could be produced directly as physical robots. The authors use a parametric context-free L-system.
For the evolutionary algorithm, the authors typically used 100 individuals in a run of the simulation, this for no more than 100 generations. Given two parents, the crossover of P_0 and P_1 works by replacing a sub-part of P_0's command with a sub-part of P_1's commands and copying that to the new offspring. Mutations can modify both production rules as well as the commands used to create the creature. As usual, the fitness function measures how far the centre of mass of the creature has moved.
Later [HP01] the authors modify their earlier work [HLP01] to be more akin to Sims' [Sim94], where a creature is co-developing brain and morphology, which means movement speed and automatic oscillation of joints are removed. Instead, the movement of a joint is controlled by a neural network. Unlike Sims [Sim94] where there is a local network for each joint as well a global network that can feed its output into the local networks, in this work, there is only one global network that controls the angle of every joint in the body. The genotype for the neural network is described by an L-system grammar. The authors claim that animats created with neural networks move faster than their non-generative counterparts without providing data.

4.1. Framsticks

Because of the stick-like look, it is easy to jump to the conclusion that Framsticks [KUa] are built using L-systems. However, part of the purpose of the Framsticks project is to support different genotype encodings and examine the effect they have on the development process. The Framsticks manual [KUb] lists ten different genotype formats and describes three of them in detail while noting that more formats are easy to add. The format describes both morphology as well as behaviour. Komosiński & Rotaru-Varga [KRV01] studies how those three formats (*f0, f1, and f4*) impact the fitness of phenotypes. *f0* or *simul* is a direct encoding of the animat. Mutation happens by changing an attribute or adding/removing one. Crossover is performed geometrically by using a half-plane for each parent that cuts the parents in two and then merging the halves into a new offspring. Unlike the other encodings described here, *f0* allows for cycles. The direct recurrent encoding, *recur* or *f1* in the Framsticks Manual [KUb], reads like an L-systems string, where each character results in a production; for example, the encoding X(X,X) results in a stick with a branch dividing into two other sticks. In *f1*, mutation and crossover work directly on the encoded string. Mutation modifies single elements of the string or adds or removes an element, while crossover works by swappings the parts of the strings inherited from the parents.

The indirect development encoding *devel* or *f4* from the manual is different from *f1* in two ways; first it allows for repeated subtrees. Secondly, the interpretation of the string is different. In *recur* the string describes creation parameters, while instead for *devel* strings describe changes to the element itself that is currently being worked on.
Komosiński & Rotaru-Varga [KRV01] compared the performance of the three genotypes in three different tests; passive height maximization, active height maximization and locomotion velocity. The difference between passive and active height maximization is that no neural networks are activated in the passive tests. *Recur* and *devel* perform better than *simul* in the height maximization tests. At the same time, there is no statistically meaningful difference between the three models when it comes to locomotion velocity. One interesting speculation from this work is that since *simul* uses a direct representation, it is not restricted by the mapping from genotype to phenotype, and it does not have the same bias towards structure that *recur* and *devel* have. In this case, that bias was useful for growing towards a maximum height.
Framsticks model joints and muscles separately. In Framsticks, joints merely provide a dampened bending axis, unlike Sims [Sim94] where forces are applied directly to the joints, making them seem more like muscles. Instead, the Framsticks model introduces bending and rotating muscles. Unlike real muscles and later works, including muscles such as Lessin et al. [LFM13], these simulated muscles are not only contractive but are capable of both pushing and pulling. Animats in Framsticks have three senses: balance (gyroscope), distance to food and touch (detection of physical contact). Figure 4 illustrates the muscles and sensors of an animat. Even though Framsticks [KUa] is an almost 20 years old project,

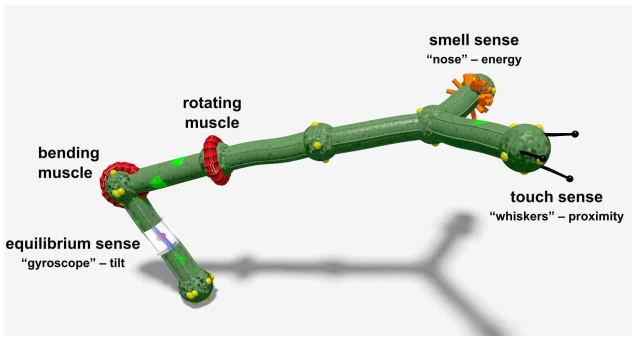

Figure 4: *Adapted rendering of a animat, generated by a newer version of the Framsticks framework, illustrating sensors and muscles*

the website is still alive, and the code repository still has active updates. It seems to be used mainly as an educational toolbox.

4.2. Modular Robots [VFRS17]

Veenstra et al. [VFRS17] compare the effectiveness of a direct versus a generative encoding for evolving virtual robots. The generative encoding is based on L-systems. As with many simulations of this type, fitness was measured by how far the robot can move in a horizontal direction. The authors use 20 seconds of simulation time, with the first 2.5 seconds being discarded, due to some robots falling over.

The direct encoding is graph-like based on the Edhmor system [FBLPD13], where nodes represent robot modules and edges connections modules, in a manner somewhat similar to Sims [Sim94], though the latter is an indirect representation that supports recursive structures. The generative encoding uses an L-system based on a context-sensitive grammar. An example of the grammar, symbolic representation and generated phenotype can be seen in Figure 5. When comparing the performance of the encod-

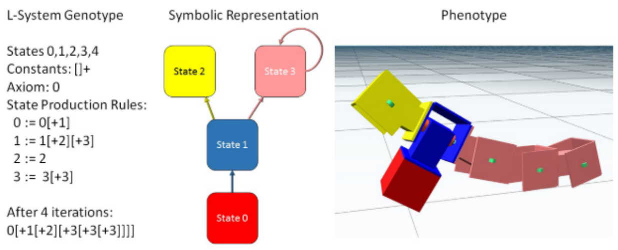

Figure 5: *Adapted reproduction of Figure 3 from [VFRS17] illustrating the generative representation*

ings, the authors found only a significant difference ($p < 0.05$) when the virtual robots had a maximum of five modules. For ten and twenty modules, the generative encoding only performed significantly better at 6,250 runs but not at 12,000 nor at 25,0000. The authors note this is a counter-intuitive result, as one would expect results to be similar when the search space is similarly sized with fewer modules. As the search-space is much larger for the direct encoding, it should be on par with the generative encoding or close to, with fewer modules and lose effectiveness as the maximum number of modules grew. One weakness of this work is that even though the two encodings used the same simulation environment — Virtual Robot Experimentation Platform (V-REP) [RSF13] — the rules that govern the evolutionary algorithms were different, and therefore it makes an accurate comparison difficult.

5. Cellular Automata

As described earlier (§ 2.1), cellular automata were invented by both Von Neumann and Ulam in the 1940s. In recent times, there have been interesting works focusing on moving (the mostly) discrete cellular automata into the continuous domain while enhancing and preserving characteristics of GoL, such as the glider.

5.1. SmoothLife

Rafler [Raf11] argues that one of the defining features of GoL is the glider [Gar70], a creature that moves smoothly across the grid and wraps around at the edges. Rafler describes a continuous version of GoL featuring a glider. The author approximates the Moore neighbourhood of a cell x with a disk, which is divided into two parts. The inner part extends to radius r_i from x and the outer part r_o from x. This gives an inner disk of width r_i (representing the cell x) and an outer ring of width $|r_o$-$r_i|$ (representing the neighborhood). Let $f(x,t)$ be the transition function, let N and M be normalizing factors, then we can define n and m as the following integrals over the entire disk (equations 1 and 2 from Rafler [Raf11]):

$$m = \frac{1}{M}\int_{|\vec{u}|<r_i} f(x+\vec{u},t)d\vec{u} \quad , \tag{1}$$

$$n = \frac{1}{N}\int_{r_i<|\vec{u}|<r_o} f(x+\vec{u},t)d\vec{u} \quad . \tag{2}$$

Thus, working in a continuous domain, the authors expand the idea of the finite statemachine, which normally describe the behaviour of a CA, to a new continuous transition function, from and to a normalised continuous domain $s(m,n) : [0,1]x[0,1]->[0,1]$, to model the behaviour of a cellular automaton in the neighbourhood of x. The author gives the full definition of the transition function in the paper. The left image in Figure 6 illustrates a SmoothLife glider.

5.2. Lenia

Lenia [Cha19] by Chan expands upon Smoothlife [Raf11] by introducing the kernel K, a composite function, which determines the "texture" and "skeleton" of a creature. The article defines Lenia both in a continuous world suitable for mathematics, as well as in a grid-based world suitable for computations. Unlike GoL, in Lenia's grid-based model, cells take on normalised floating-point values. The authors call this digital (GoL) versus analogue (Lenia). The basic idea is as follows. Define a unimodal, nonmonotonic function $G : [0:1]->[-1:1]$. Assume that at time t, we have the outline of a creature A^t in the Moore neighbourhood N of a cell x. Then we apply G to the result of applying K to A^t, in order to calculate the next generation $A^{t+\Delta t}$ as shown in equation 3 (derived from Equation 7, 8 and 9 in [Cha19]):

$$A^{t+\Delta t}(x) = [A^t(x) + \Delta t G(K * A^t(x))]_0^1 \tag{3}$$

The term $G(K*A^t)(x)$ expands to an integral over the Moore neighbourhood of A, which due to the discrete nature of a computer, is approximated by a summation over the cells. Let N be the Moore neighbourhood of a cell x, then $K*A^t(x)$ is calculated inside a computer as shown in equation 4 (part of Equation 7 and 8 in [Cha19]):

$$K * A^t(x) = \sum_{n \in N} K(n) * A^t(x+n)\Delta x^2 \tag{4}$$

According to the author, the animats (or life forms as they are referred to) turn out to be rather resilient to collisions and deformations. This is in contrast to GoL, where the eco-system of animats often seem to be a bit brittle and easy to disrupt. The right image in Figure 6 illustrates a glider from Lenia. In the next work [Cha20a],

Figure 6: *Left: a screenshot of a SmoothLife glider [Mur18, Raf11]. Right: a screenshot of a glider in Lenia [Cha19, Cha20b]*

the author follows up with further analyses of the creatures with identifications of self-replicating and pattern emitting creatures, as well as generally identifying creatures in 3D, and concludes, somewhat surprisingly, that there seem to be fewer stable creatures in 3D, and that they tend to be more stationary and rotate less.

6. Gene Regulatory Networks

Gene Regular Networks (GRNs), like cellular automata, are modelled with cell assemblies, but instead of having a set number of cells, GRNs often start with a single or very few cells and then grow this number over time through a diffusion process. Beginning with the work of Eggenberger [Egg97], where 3D structures based on GRNs are developed, Bongard and Pfeifer [BP03] evolved virtual creatures using a similar technique. The agents in their simulation evolve through structural division: they start with a single sphere that, upon doubling its size, splits into two default sized-spheres; then the growth-split process iterates. Even though the creatures are modelled with spheres, it is easiest to think of these as being modelled in a local voxel space. Each voxel has the six voxels along the three primary axes as neighbours. During development, genes diffuse to these neighbours, so when a sphere splits, the copy diffuses to the neighbouring voxel with the highest concentration of growth genes. Adjacent voxels are connected through a 1D rotational joint, though it will be rigid if the local structure does not have a motor neuron. Other forms of neurons include sensors (touch, proprioceptive and light) as well as output signals (constant and harmonic). Neurons are connected through a synaptic structure. Both neural networks and morphology are modelled as part of this technique. The authors evolve animats both for locomotion and block pushing, and used the Math Engine [PLC03] for simulating the phenotypes.

6.1. Soft-bodied Multicellular animats

Joachimczak et al. bridged GRNs and neural networks with their exploration and simulation of soft-bodied animats [JW11, JW12, JKDW13a, JKDW13b, JSA14, JKSA15]. In the original work [JW11], an artificial GRN (or AGRN) is developed to create 3D cellular structures to solve the French-flag test [Wol69], which is a challenge to create a cellular pattern separated into red, blue and white zones.
In their next work [JW12], the authors modify their initial solution by limiting it to 2D and adding cellular functions to represent springs. When the genotype is mapped to a phenotype, the authors start by building a cell structure. A border is then created around the structure by adding edges between neighbouring cells sitting at the limit of the structure. The rest of the structure is then triangulated by selecting the centre of each cell as a vertex and polygonized using the Delaunay triangulation [Nik34]. As a final step, the Gabriel subgraph [GS69] of the triangulation is found and used as the final shape structure. For physical simulation, edges between two points in the triangle mesh also function like springs in a spring-mass system where the vertices all have equal mass. As the initial implementation only simulates swimming, forces for fluid drag are computed and applied directly before the integration step is calculated using the Bullet physics engine [Cou]. Just like Sims [Sim94], an individual's fitness is measured as the distance travelled by the centre of mass of the animat. The authors later expand this work with walking and swimming towards a target [JKDW13a, JKDW13b].

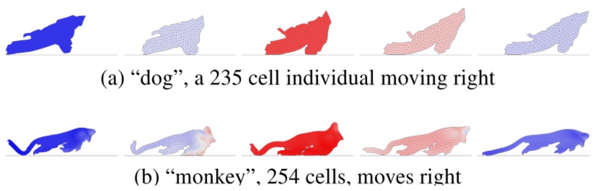
(a) "dog", a 235 cell individual moving right
(b) "monkey", 254 cells, moves right

Figure 7: *Adapted reproduction of Figure 5 from [JSA14] showing the movement pattern of a few of the evolved animats*

Building upon Joachimczak et al.'s earlier works, the authors present a simplified and more performant GRN model [JSA14]. The basics of the model are: (a) progressive growth of the cellular data structure via cell division, (b) all cells are controlled by the same GRN, (3) cells have individual behaviour, as part of the decision process is based on the local neighbourhood. The authors' do not use the term CPPN [Sta07], but the function in the nodes of the GRN can vary, as they are chosen from a pre-selected set of possible functions. The architecture of the GRN is based on NEAT [SM02] (NeuroEvolution of Augmenting Topologies). As in earlier works, the resulting phenotype is a triangulated spring-mass model.
Joachimczak et al. [JKSA15] finalise the series of works by using cell growth to expand inside and fill existing structures, like the outline of an animal or character, and then using the physical simulation to make the drawing move. This work is really much more like a behaviour controller than focused on morphology but is mentioned here for completeness.

7. Artificial Neural Networks

ANNs are often used for simulating the behaviour of animats. This can be traced back to Sims [Sim94] and has continued to the present day. In 2007 Stanley [Sta07] defined the term Compositional Pattern Producing Network (CPPN), which is an ANN where signalling can be modelled with a number of different functions instead of only sigmoids. As discussed earlier, § 2.6, Sims [Sim94] had already used such networks for behavioural control, though Stanley was the first to define the term and use them for modelling morphology. In the initial work, Stanley used CPPNs for generating 2D images.
From 2010 and onwards, Auerbach & Bongard published a series of works on using CPPNs for generating animats. The original work [AB10b] used CPPNs for generating 3D structures made of spheres, where all structures were rigidly attached to each other, and the only movement consisted of falling using gravity. In the next update [AB10a], the structures were able to move on their own accord via 1D rotational joints sitting between selected sphere elements. Whether a joint is created or not depends entirely on one of the outputs of the CPPN. Neural networks are not used for the brain, instead movement is generated with a periodic sine function. Each joint has an associated constant that offsets the central periodic sine function's local output.
Along with ANN-based controllers, Auerbach & Bongard made several improvements in their next publication [AB11]. Instead of using a period function for controlling the joints, each sphere

(node) in an animat has sensors, and a continuous-time recurrent neural network (CTRNN) [Bee06] for behavioural control. There are four types of sensors: distance to target, touch, proprioceptive and a time-based signal outputting sensor. Additionally, and unlike earlier works [AB10b, AB10a], the CPPNs for generating the morphology are allowed to be recurrent. Specifically testing the effect of recurrent CPPNs, the authors find that animats that have recurrent networks outperform those which do not.

In the last of the series of works [AB12], the authors reduce complexity by using oscillating hinge-joint controllers with two degrees of freedom while focusing on the possible evolution and growth of complex morphology. The authors want to test if a more complex morphology evolves as the complexity of the simulation environment increases. CPPNs are still used for generating the morphology, but instead of growing spheres, Marching Cubes [LC87] is used for triangulating a point cloud. Their results show that their assumption is indeed valid, though they also argue more work is needed to check if a complex controller can compensate for needing more complex morphology, or if specific environments encourage the evolution of complex morphologies, while other environments might instead encourage complex controllers.

All of the aforementioned works by Auerbach and Bongard [AB10a, AB10b, AB11, AB12] use the physics engine ODE [Smi] for simulation. As we will see in the following, except for the works mentioned here and those of Veenstra & Glette [VG20] (details in § 14), most uses of CPPNs for generating morphology has been for soft-body simulations.

8. Soft-body simulations

A large group of works focus on soft-body simulations. This started around 2010 with Hiller et al. [HL10] though the research area has taken on a life of its own since then. We have chosen a selection of works relevant to the morphology of virtual creatures. In Corucci et al.'s overview on soft robotics development [CCK*17], they introduce a number of concepts from biology into soft-body simulations, such as *morphological computation*, *morphosis*, *embodied intelligence* and *development plasticity*. While their survey is focused on robotics, the terms are just as useful in a purely virtual world.

8.1. Evolving Amorphous Robots

Hiller et al. [HL10] focused on the software generation of soft-body creatures, which given a few more advances in physical materials, can potentially be 3D printed. They describe two different contributions: (i) the generation of soft-body virtual creatures using density functions and (ii) a physics systems that can simulate such creatures. The authors denote their soft-body virtual creatures as continuous amorphous robots. A single creature consists of a C^0 continuous 3D surface without any holes. Comparing to Sims, the animats can be conceptually mapped to continuous amorphous robots by replacing the genotype node link graph with a material distribution and the joints/actuators with a material. The authors use two different materials in their work, a passive one and one that can expand and contract, causing the creature to move when changing volume. Each density function represents one material. A surface is created by taking the maximum of all density functions at a given point in a lattice. The material represented by the density function with the highest value at a given point is the material instantiated at that point.

Any method that can represent a density function can be used for generating a phenotype. The authors examine three different methods: the Discrete Cosine Transform (DCT), Compositional Pattern Producing Network [Sta07] (CPPN), as well as a Gaussian Mixtures representation (GMX). The DCT is a special case of the Fourier Transform, where the genotype consists of a 3D matrix of frequency amplitude weights. These weights are inversely mapped and summed through an Inverse Fourier Transform. In the CPPN case, a network with weights is evolved, which takes a 3D position as input and outputs a density weight as its singular value. The GMX is a composition of Gaussian functions with a centre in the lattice and different falloffs, as well as a weight than can be either negative or positive. The authors measured the performance of these three representations and found "the GMX representation outperformed the other representations consistently" [HL10].

As mentioned earlier, to test their soft-body phenotypes, the authors developed a custom physics and collision system. The physics system works on a voxelized version of the phenotypes and handles rotational and translational effects, as well as friction, any number of materials and self-intersection.

Hiller & Lipson [HL14] describe VoxCAD [Hila] and Voxelyze [Hilb], which together form an open-source soft-body simulator. Several papers [CMCL13,C*16,CCGS*18,KCB18,KBLB20] have used VoxCAD/Voxelyze for simulation. The simulator uses a custom physics engine where soft-bodies are represented as voxels in a big lattice connected through a mass-spring system. Springs are simulated as beams that have rotational and translational stiffness.

8.2. Unshackling Evolution

Like Veenstra et al. [VFRS17] and Hiller et al. [HL10], Cheney et al. [CMCL13] look at the effect different genotype encodings have on the performance of evolutionary development, i.e. how quickly does the fitness score improve after each generation. Rather than use DCT, CPPN or GMX, as Hiller et al. [HL10] did, Cheney et al. test a direct representation and a CPPN-NEAT [Sta07, SM02] representation. They find that CPPN-NEAT outperforms the direct encoding. Note that Auerbach & Bogard [AB10b] had already shown that CPPN-NEAT is useful for evolving virtual 3D structures. We also note that it would have been useful that they also test GMX in order to compare their best results to those of Hiller et al. [HL10]. As future work, Cheney et al. suggest also trying out the HyperNEAT algorithm [CBOP09] as it has shown itself to be useful for evolving ANN-based behaviour controllers for robots.

Cheney et al. [CMCL13] also try out the following four different penalty functions for the fitness to see if it would have an impact on evolution: amount of voxels, amount of actuable material, adjoining faces between voxels and none. Generally, the authors found that, while the penalty function does drive morphological evolution, it doesn't influence the overall performance of the algorithm.

The voxel-based soft-body creatures consist of two fundamentally different types of voxels— active and passive. Active voxels contract and expand at a specified frequency, while passive voxels are either soft or hard. All in all, voxels can belong to one of four materials (colours): (i) greens have a period actuation of 20%, (ii) reds

behave like greens but with the opposite phase, (iii) light blues are passive, soft and bend easily, while (iv) blues are also passive but stiffer and more resistant to pressure from neighbours.
The authors do not give much information on the direct representation, except that it is implemented using GAlib [Wal12]. The CPPN-NEAT network is constructed in such a way that it has four inputs and five outputs per voxel. The inputs are the (x,y,z) coordinated and the distance to centre. One output determines if the material is within the voxel, while the maximum of the remaining four outputs specifies the material type.
The implementation was tested in the VoxCAD [Hila] test suite simulator. As far as we have been able to find out, the VoxCAD repo was made public after Cheney et al. [CMCL13] and first described by Hiller et al. [HL14]; it is thus likely that Cheney et al. had direct access to an earlier version of the repository and they do thank Hiller for help with VoxCAD.

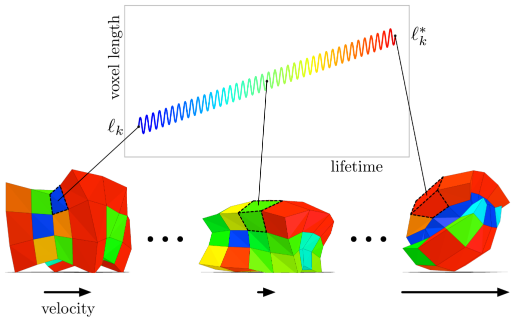

Figure 8: *Adapted reproduction of Figure 1 from [KCB18], showing how a soft-robot can change its shape during its life time, as the size of voxels varies. Blue voxels are small, green medium and red big. The Figure is licensed under CC by 4.0 [CCB]*

8.3. Swimming

Nearly all simulations involving swimming animats have used a simple drag model to simulate a fluid environment [C*16, Sim94, Tu96, TT00]. This completely ignores other important aspects, such as the pressure of a fluid on a creature's body. Using the physics engine LiquidFun [Goo13], Johnson et al. [JPH19] tested swimming pseudo-soft body agents in two types of simulated liquids, one created with drag-only, like earlier simulations, and one simulating the liquid using particles. The swimmers are pseudo soft-body agents, as the agents are created as small spring-mass system models with a hard polygonal *skin*, which is used for a rigid body physics simulation with the particle-based liquid. The authors created two types of agents; one is a small triangle with connected springs with prismatic joints in between to drive the motion, while the other type has two chambers that can be compressed to push out particles and drive the movement forward. The authors found that the triangular swimmer did not move at all in the drag model if the rate of contraction of the springs matched the rate of expansion, however turning worked well. It could move in the particle-based liquid, but only if the actuation frequency exceeded a certain threshold. The agent with the compressed chambers was only tested in the particle-liquid and moved well, but with a high turning radius. It is also noted that turbulence occurred in the particle-based liquid, which an agent could potentially take advantage of.
As part of their work on modelling both the graphical and physical aspects of an eco-system for fish, Tu & Terzopoulos [TT00] created a technical model for swimming. However, to be able to present the entire series of work from a more complete holistic view, their work on swimming and fish behaviour is presented together with the graphical aspects in section 12.

8.4. How morphological development can guide evolution

Kriegman et al. [KCB18] use the Voxelyze physics engine [Hilb, HL14] to test out continuous development during the lifetime of an individual, part of the "evo-devo" family of algorithms. They conclude their model results in more robust behaviour controllers. They test out their 4x4x3 voxel soft-robots and compare them with an evo model. Each voxel is interpolated linearly in size during its lifetime from a start size a to the end size b, which implies $a = b$ in the evo model. This interpolation, causing a change of shape in the phenotype is exemplified in Figure 8. The model uses a separate representation for morphology and control. For the evo-devo model, both morphology and control parameters are interpolated across the lifetime of the robot. The authors find that the evo-devo model always converges significantly faster than the evo model unless the mutation rate is increased to a level where it results in random search. In particular, the evo-devo model loses very little fitness if the final generation is rerun without development enabled, and it still outperforms the evo model. It is also noted that behaviour controllers are quite brittle in that they are susceptible to changes in morphology. In comparison, by using evo-devo, behaviour controllers are evolved which are more robust to changes in morphology.
Inspired by the work by Hiller et al. [HL14] on the VoxCAD soft-body simulator, Francesco et al. [C*16] simulated swimmers by adding drag to an existing model. The simulated robots are evolved using two CPPN-NEAT networks, one for the evolution of the morphology and one for the behaviour. The robots are evolved using a multi-objective algorithm which maximizes distance travelled, minimizes the amount of voxels actuated, minimizes the number of voxels in the model, and minimizes the age of each individual [SL10].
Corucci et al. [CCGS*18] later expanded this work by trying out five different materials with different stiffness for movements on land or in water. Additionally, they test for exaptation by moving soft robots developed on land to water, and vice versa. This work uses the same multi-objective algorithm described in [C*16], except for the Age-Fitness Pareto Optimization [SL10], which they only mention and may add in a future version. They find that stiff limbs are suitable for walking on land, while the opposite is true in water. The softer a creature is, the better a swimmer it becomes. For testing land->water and water->land, the authors ran the first half of the generations in the original environment and then the last half in the other environment they were transferred to. They find that moving soft robots evolved on land to water is always detrimental to evolutionary development. On the other hand, they found indicators that evolving a robot in water before moving it on land was beneficial for the development, as opposed to developing on land

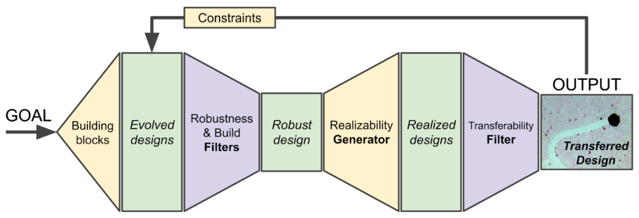

Figure 9: *Adapted version of Figure S1 from the appendix of [KBLB20] showing the pipeline for creating* in vivo *soft-body robots. The Figure is licensed under CC by 4.0 [CCB]*

only. Unfortunately, the authors did not manage to reject the null hypothesis by achieving statistical significance ($p < 0.05$) on this last test, so this needs more work.

8.5. Living Soft-Body Robots

Kriegman et al. [KBLB20] used the VoxCAD [Hila] simulator as part of a pipeline for creating living *in vivo* soft-body robots. CPPNs are used for describing the genotype, where the networks map a 3D coordinate in local genotype space to two values: (i) a boolean indicating the presence of a material, and (ii) a descriptor of the material as contractive or passive. In some of their experiments, they used a second CPPN to set phase-offsets for actuation. This way of using CPPNs is similar to Kriegman et al. [KCB18], though, in this work, they use the Age-Fitness Pareto Optimisation algorithm [SL10] for simulating evolution. The complete pipeline for *in vivo* robots, can be seen in Figure 9. In this model, they still have only one representation of the genotype but two versions of the phenotype, one *in silico* and one *in vivo*. The robot has its fitness measured at both stages. The *in vivo* phenotypes are built from cells extracted from embryos. The process of growing and shaping is a combination of using chemical processes and surgery using microcautery and surgical forceps and is described in depth in additional materials in [KBLB20].

9. Conceptual Blending

In 2003, Ribeiro et al [RPM*03] and Pereira & Cardoso [PC03] applied conceptual blending [FT] as a computational creativity model. Conceptual blending is a model for how concepts are grouped separately in the human consciousness, in what are called *mental spaces*. For example, the mental space Wolf could hold the concepts furry, four legs, pointy teeth, tail, carnivorous, and more. Mental spaces and mappings between the concepts they contain can be used to create new mental spaces. For example, imagine two mental spaces, Human and Wolf, which can be blended to create the space Werewolf by mapping between similar concepts in each space, such as the number of legs, teeth style, diet, physiology. Conceptual blending requires a minimum of four mental spaces; two input or source spaces to be blended, a target or blend space, and a general space that functions as a knowledge base.

Building on such notions, Pereira & Cardoso [PC03] describe Divago, an implementation of conceptual blending as a computational creativity framework. They focus on the implementation and use of *Optimality Pressures* as introduced by Fauconnier & Turner [FT] to generate blends between a horse and a bird. In essence, the different *optimality pressures* end up as weights for an evolutionary algorithm when searching for appropriate blends.

Ribeiro et al. [RPM*03], provide a Divago-inspired implementation consisting of four submodules: input, creature builder, game knowledge base, and output. One of their goals is to make a practical system, so for example, when the creature builder starts to build the concept mapping, user-controlled restrictions can be applied to the mappings. For creating the blend itself, alike Pereira & Cardoso [PC03], they implement optimality pressures as weighted constraints to assemble the fitness function. After the GA has converged on a result, the blended model goes through two additional phases in the creature builder: *elaboration* and *validation*. Elaboration uses the knowledge base to add extra features. For example, to add feathers to a bird or apply game-specific rules, such as if a blended character ends up only on one leg, a wooden leg is added. The output module takes care of assembling the final 3D model from the blended description, including colouring and merging 3D models, as well as export functionality to make sure the model can be used in a 3rd party program. It is not only visual features that are blended: game-specific properties, such as strength, chattiness and height, are also included.

10. Self-Assembling Morphologies

Pathak et al. [PLD*19] produced an original platform, where virtual robots can assemble and disassemble dynamically to reconfigure themselves during simulation. According to the authors, monolithic controllers tend to perform poorly as the numbers of limbs grow, and so they set out to see if modular controllers generalise better than monolithic ones.

In their work, all robots start out as individuals, each consisting of one limb and one motor (Figure 10). All limbs exist as individual robots, and each is controlled by a neural network that outputs torque, designating how much the limb rotates. In order to create more advanced morphologies, the limbs can attach and detach themselves from each other's motor joints via magnetism. The limbs create a common policy network when attached. They call this mechanism a *Dynamic Graph Network* (DGN) since the topology can dynamically change during simulation when limbs attach and detach. Rewards are shared when limbs have attached together to form a more complex structure.

The authors create two baselines to compare against. Both are based on a monolithic policy network, where one has a static morphology and the other a dynamic one. Calculating the mean reward across the types of tasks; standing up with no wind, standing up with wind, and locomotion across a bumpy terrain, the authors show that the DGN performs better with a maximum of 2500 steps. One of the things that is missing from the authors' baseline test is that since a direct representation has been shown to underperform compared to an indirect one [CMCL13], including a monolithic indirect representation for comparison would have made the work more complete. Finally, the authors always trained with six limbs but show that the approach generalizes to robots with three to twelve limbs with no extra training, never performing worse than 62% of the original score.

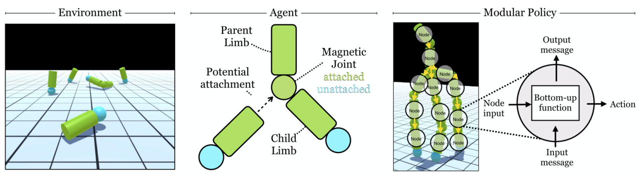

Figure 10: *Reproduction of Figure 1 from Pathak et al. [PLD*19] showing the structure of an assembled robot. Left: robot primitives. Middle: self-aggregation through a magnetic joint. Right: message and input flow during simulation.*

11. Spore

Many games, including Petz [MMU95], Black & White [Stu01], Spore [Max08] and No Man's Sky [Gam16] included procedurally generated or modified morphology for some or all of the creatures in the game. The methods that games apply for this effect are all hybrid, as they are composed of several different methods to get the desired visual look and behaviour. It is important to differentiate between games that generate the creatures almost from scratch or at least from basic primitives against games that procedurally modify existing human-made models. For example, the creatures in Black & White [Stu01] can grow up and modify their physical look based on their physical health and moral attitude, however, all modifications look like they are made to the same base model. Evolva [Art00] is another example of a game that is not covered in this study, as the morphological changes all look to be taking place to an existing model. The focus in this work is on algorithms with a more fundamental approach, where a template of the procedurally generated morphologies has not been modelled beforehand. This can mean either generating the creature from nothing or procedurally assembling it from basic primitives. Most games that qualify for our definition of procedurally generated morphology, such as Spore [Max08], No Man's Sky [Gam16] and Impossible Creatures [Ent03], use the latter method.

However, not much knowledge has been made publicly available on the implementation of the procedural techniques driving most of these games. One standout exception is Spore, for which Hecker [H*08, Hec10] has made a lot of information available, along with Compton (et al.) [C*07, COM13, Com16, Com17], Gingold [Gin03, Gin07], DeBry [DGH*07], Willmott [Wil07] and Smith [Smi14].

Spore [Max08] is a game featuring the evolution of creatures. The game spans five different evolutionary stages, all of which feature different gameplay and mechanics. Causing great debate, Spore ended up being more of a game featuring evolutionary elements than a game about evolution [Smi14].

With an overview of the creature pipeline in Spore, Compton [Com17] shows how implicit surfaces are generated around a 3D graph of a skeletal hierarchical structure. From this representation, uv maps are extracted for procedural painting, and animations are generated. Hecker [Hec10] reports about his contributions to Spore, where he describes this pipeline in more detail.

Since players can attach and detach body parts in the editor, a robust model that could change the creature morphology dynamically was needed. Hecker used implicit surfaces for modelling the body parts. At the time, the patent on the Marching Cubes [LC87] algorithm was still valid, so Hecker chose to use an ear clipping [Mei75] algorithm for the dynamic triangulation of the implicit surfaces. Already then, there were other methods than Marching Cubes published for polygonization of implicit surfaces, such as the Marching Triangles method [HI97]. The implicit function used for modelling the surfaces is a variation of the one given by Triquet et al. [FPC01]. For animation, bone weights for vertices are generated by identifying which part of the creature an implicit function describes. Spore's innovative animation system is described in Hecker et al. [H*08]. As the users can create their own creatures from simple parts, the central issue facing the developers was how to make an animation for a skeleton that wasn't known a priori. The developers came up with a semantics driven system, where general animations can specialise whenever they are applied to a specific skeleton and morphology. To help this specialisation along, semantic capabilities (tags) can be applied to body parts, e.g. "the body associated with a hand mesh has the grasper capability". The authors report that they have tested the system on hundreds of creatures and over a thousand different animations, with a pass rate of approximately 90%.

Spore's Art Director, Quigley, wrote a couple of blog posts on the texturing system of Spore. Quigley drew a set of creatures with example textures that the team wanted to be able to reproduce using their texturing system [Quib]. Quigley [Quia] goes on to describe the system, which ended up using particle systems, where particles act as virtual paintbrushes, painting the surface of a creature as the particles move. The system included a certain amount of artistic control, so particle systems could be directed, for example, to move along the spine of a character or towards specific features. This system is for general or ambient texturing and doesn't cover detail texturing. Eyes, teeth and the like were still painted by hand. For the user, the system appears to work directly on the surface of 3D creatures. In reality, behind the scenes, a 2D texture atlas is automatically generated and mapped onto the model.

12. Art

We have already covered what is most likely the most well-known uses of creating virtual creatures for artistic purposes; Kawaguchi's spirally shapes [Kaw82], Sims' evolved creatures [Sim94] and Latham & Todd's evolutionary art [TL92] (covered in detail in § 2). Some of the derived works, such as Ray's aesthetically evolved *pets* [Ray01] (covered in § 3) are artful projects. Other ALife works such as Chan's Lenia [Cha19, Cha20a] also possess a particular artistic look. However, in the following, we cover some of the more artistic virtual creatures projects that do not easily fall into any of the other categories.

Form [Row99] was a project by Rowbottom, inspired by Dawkins [Daw86] and Todd & Latham [TL92]. Just like the latter work, this project is built on basic geometric forms, which can have mathematical operations applied to them. Some of the shapes generated by Form can be clearly recognized as being inspired by Todd & Latham. However, mutations are inspired more by Dawkins' works, as the structure itself is left as is, but the genes modify parameters instead. Rowbottom presents five different iterations of the project, ending up with a final version that supports animations and interpolating between forms.

With a project placed somewhere between ALife and Art, Tu & Terzopoulos [TT00] set out to simulate a fishbowl, a complete ecosystem of autonomous fish, including morphology, motor control, sensor perception, simulation of hunger, breeding and hunting, as well as additional behaviours such as schooling. The model described here is more of a hybrid method, as often seen in art projects, where several disparate techniques combine to create convincing morphology and behaviour.

The movements of the fish themselves are modelled using a biomechanical spring-mass system with motors attached to specific muscle springs to facilitate turning. There are three motor controllers: turn left, turn right and go straight. Pectoral fins that can turn up and down, like the small tail wings on the back of an aeroplane, are used to create full 3D controls.

The graphical model of the fish is separate from the physics-driven spring-mass representation of the model. The fish are modelled using non-uniform rational B-splines (NURBS) in the Alias^{TM} 3D modelling package [Aut85]. For visualising the mesh, the authors created a custom tool for uv texturing coordinates to map photographs of real fish onto the NURBS mesh. The simulated biomechanical spring-mass model is subsequently mapped onto the NURBS surface, which gives the visual impression of a swimming fish.

For input to behaviours, the fish not only have a sense of touch through collision detection, but the authors have also added sight and sensitivity to temperature. Sight is modelled mostly through a combination of raycasts and occlusion tests. Fish also have variables that help determine behaviour, such as *hunger*, *fear* and *libido*, as well as a type which can be *predator* or *pacifist*. Together with a few designer determined attributes such as preferences for light, darkness, warmth, cold, schooling and gender, this completely determines the fishes' general behaviour. Together the perception system and attribute variables combine into a dynamic brain that results in behaviour in the simulation.

Lomas' cellular art [Lom14] has most in common with Gene Regular Networks (§6) as it is based on the idea of cell splitting and diffusion. Kozinarium [KK18] is a virtual creature art project, run by Kozlov & Kozlova. The virtual creatures are generated entirely through node-based systems using Houdini [Inc96] and Fusion [Des96]. Kozinarium's creatures are all imaginary, yet they are realistic-looking with a convincing animation in natural-looking surroundings. The creatures even get a little matching description, lending to the idea that they could be real. According to the project, they released 26 creatures in 2019. A few new ones have also been published during 2020. Kozlov documented some of the techniques behind Kozinarium in a blog post [Koz18]. The main setup for designing creatures is based on Channel Nodes (CHOP) in Houdini. By chaining CHOPs together, data can flow from node to node along designated channels. Each node has a number of inputs that work as parameters for a function, and the result is passed on as output from the node. The node graph for Kozinarium consists of approximately 1700 nodes. The primary way for a user to generate a new creature is through changing a set of seed values. Creatures are simulated using Finite Element Methods and the skeletal rigging is automatic. Once the creatures are finished, animations are rendered out in Mantra, the built-in renderer for Houdini and post-processing done in Fusion [Des96].

Toor and Bertsch [TB20] detail how GANs (Generative Adversarial Networks) have been trained and later used for automating the generation of images of virtual creatures. For this work, the artist makes a basic sketch of a creature, and the algorithm fills out the details and creates the final image of a creature. A web program, *Chimera Painter* is available for public use, where creature designers can sketch their own creatures and let the machine learning algorithm do the rest (the link is given in [TB20]).

13. Simulations and Ecosystems

Over time, several simulations featuring virtual creatures have been made. Some of those have already been described earlier such as Framsticks [KUa] in §4.1, or the cellular automata described in §2.1 and §5. Simulations are often harder to categorise as art, ALife, games or something else. Most of the ones described here overlap several categories. For example, Gene Pool [Ven, Ven05] is a simulation aiming to compare natural and sexual selection, so it is an ALife application, but it also has the looks and playfulness of a game. In fact, the direct predecessor of Gene Pool, Darwin Pond [Roc96], was developed by a game company. The simulation is interesting enough to look at that it would not feel out of place as an interactive installation in an art gallery.

Yaeger [Yae93] created PolyWorld, one of the first ALife simulations where several co-existing creatures living in a 3D world have needs and develop over time as generations come and go. The creatures' morphology does not change over time, but the genes have properties that indicate morphological features such as size, strength, and maximum speed. Most of the other genes relate to neurons. Actions spend energy, so the creatures need to hunt for food. This can either be by eating other creatures or finding food. Eating is just one of seven behaviours: Eating, mating, fighting, moving, turning, focusing and lighting. The last two might require some explanation. Focusing relates to the vision system of the creatures. All creatures in PolyWorld have vision, consisting of a single row of pixels rendered from its view. The focusing behaviour refers to adjusting the horizontal viewing angle. Lighting refers to creatures being able to change the brightness of their body in the world. The author state that this could have been used as a form of communication, but there is no evidence the creatures did that. The simulation took place on a flat plane with barriers that could be put up in arbitrary locations. Even this simple model created a world with species with very different behaviours and survival strategies.

Darwin Pond [Roc96] is still accessible online. However, the application can seem a bit erratic due to Microsoft Windows OS interoperability issues with applications designed for running on a previous version of the operating system. The manual is still available for download. According to it, the swimmers in the game *need* to eat and *want* to breed. These two goals can be conflicting. According to the state diagram in the manual, hunger always dominates mating. Swimmers have 15 genes each, which control morphology and behaviour. The movement is controlled through a harmonic function, whose frequency and radius is manipulated through genes. The app comes with an impressive user interface, so one can follow and inspect swimmers, save preferred versions, change growth and distribution of food, add and remove swimmers, etc. There are also graphs showing such info as the amount of food and swimmers over time and the distribution of the six body colours (limbs have a single colour, but swimmers can consist of limbs with different

Year	*Section*	*Authors*	*Dim*	*Phenotype*	*Main Techniques*	*Simulation Framework*
1948	2.1	Neuman [VN66]			Cellular Automata	
1954	2.1	Barricelli [Fog06]			Cellular Automata	
1968	2.2	Lindenmayer [Lin68]			L-systems	
1970	2.1	Conway & Gardner [Gar70]	2D		Cellular Automata	
1983	2.1	Wolfram [Wol83]	1D		Cellular Automata	
1986	2.4	Dawkins [Daw86]	2D			
1986	2.2	Prusinkiewicz [Pru86]	2D		L-systems	
2001	4	Hornby et al. [HP01, HLP01]	3D	Rigid-body	L-systems, ANNs	
2001	4.1	Komosiński & Rotaru-Varga [KRV01]	3D	Rigid-body	L-systems, open-ended genotype representation, muscles, ANNs	
2003	6	Bongard & Pfeifer [BP03]	3D	Rigid-body	GRNs, ANNs	Math Engine [PLC03]
2003	3.1	Shim et al. [SK03, SCH04]	3D	Rigid-body	Sims' ANN, Flying	ODE [Smi]
2007	3	Lassabe et al. [LLD07]	3D	Rigid-body	Sims' variant of ANN & Custom classifiers	Breve [Kle03]
2010	8.1	Hiller et al. [HL10]	3D	Soft-body	CPPN, DCT, GMX	VoxCAD / Voxelyze [Hila, Hilb]
2010	7	Auerbach & Bongard [AB10a, AB10b, AB11, AB12]	3D	Rigid-body	CPPNs	ODE [Smi]
2011	6.1	Michał Joachimczak et al. [JW11, JW12, JKDW13a, JKDW13b, JSA14, JKSA15]	3D	Soft-body	GRNs	Bullet [Cou]
2011	5.1	Rafler [Raf11]	2D		Cellular Automata	
2013	8.2	Cheney et al. [CMCL13]	3D	Soft-body	CPPN	VoxCAD / Voxelyze [Hila, Hilb]
2013	3.2	Lessin et al. [LFM13, LFM14a, LFM14b, LFMR15]	3D	Rigid-body	ESP, muscles, Sims' variant of ANN	PhysX [Cor]
2016	8.4	Francesco et al. [C*16]	3D	Soft-body	CPPN, swimming	VoxCAD / Voxelyze [Hila, Hilb]
2017	4.2	Veenstra et al. [VFRS17]	3D	Rigid-body	L-systems	V-REP [RSF13]
2018	8.4	Corucci et al. [CCGS*18]	3D	Soft-body	CPPN, swimming, exaptation	VoxCAD / Voxelyze [Hila, Hilb]
2018	8.4	Kriegman et al. [KCB18, KBLB20]	3D	Soft-body	CPPN, evo vs. evo-devo	VoxCAD / Voxelyze [Hila, Hilb]
2019	5.2	Chan [Cha19, Cha20a]			Cellular Automata	
2019	8	Johnson et al. [JPH19]	2D	Soft-body	Swimming, particle physics	Liquid Fun [Goo13]
2019	10	Pathak et al. [PLD*19]	3D	Rigid-body	Dynamic Graph Network	Unity ML [JBT*18]
2020	8.5	Kriegman et al. [KBLB20]	3D	Soft-body	CPPN, in-vivo soft-robots	VoxCAD / Voxelyze [Hila, Hilb]
2020	14	Veenstra & Glette [VG20]	2D	Rigid-body	CPPN, L-systems	Box2D [Cat06]

Table 2: *Overview of ALife inspired works presented in this paper. Acronyms used in the Main Techniques columns include ANN (Artificial Neural Network), CPPN (Compositional Pattern Producing Network), DCT (Discrete Cosine Transform), ESP (Encapsulation, Syllabus & Pandemonium), GMX (Gaussian Mixtures) and GRN (Gene Regulatory Network)*

Year	*Section*	*Title / Method*	*Authors*	*Main Techniques*
2003	9	Conceptual Blending	Ribeiro et al [RPM*03], Pereira & Cardoso [PC03]	Conceptual Blending
2008	11	Spore	Maxis [Max08]	Implicit Surfaces & Mesh generation [Hec10, Com17], procedural texturing [Quib, Quia], animation [H*08]
2015	3.2.1	Darwin's Avatars	Lessin & Risi [LR15]	ESP, muscles, Sims' variant of ANN

Table 3: *Overview of games and game technologies presented in this paper*

colours).
Ventrella's [Ven, Ven05] Gene Pool builds on the ideas and looks of Darwin Pond [Roc96]. In Gene Pool, the world is inhabited by simulated swimbots, that interbreed and compete for food. The main goal is to investigate any tension and interaction between natural and sexual selection. In the simulation, swimbots spend energy when they move. Once its energy level drops below a certain threshold, a swimbot starts looking for food. A swimbot with zero

Year	*Section*	*Authors*	*Dim*	*Phenotype*	*Main Techniques*
1982	2.3	Kawaguchi [Kaw82]	3D		
1992	2.5	Latham & Todd [TL92]	3D		
1994	2.6	Sims [Sim94]	3D	Rigid-body	Sims' variant of ANN
1996	12	Tu & Terzopoulos [Tu96, TT00]	3D	Hybrid	Mass-Spring, Computer-vision [Ter95, TR97], texturing
1999	12	Rowbottom [Row99]	3D		
2002	3	Ray [Ray01]	3D	Rigid-body	Sims' variant of ANN
2014	12	Lomas [Lom14]	3D		Gene Regular Networks
2018	12	Kozlov & Kozlova [Koz15, Koz17, KK18]	3D		Houdini [Inc96]
2019	5.2	Chan [Cha19, Cha20a]	2&3D		Cellular Automata
2020	12	Toor & Bertsch [TB20]	2D		GANs

Table 4: *Overview of artworks presented in this paper*

Year	*Title*	*Authors*
1993	PolyWorld	Yaeger [Yae93]
1996	Darwin Pond	Rocket Science Games [Roc96]
2000	Soda Constructor	Burton [Bur00]
2005	Gene Pool	Ventrella [Ven, Ven05]
2008	Creature Academy	Pilat & Jacob [PJ08]
2010	Sticky Feet	Turk [Tur10]
2011		Lehman & Stanley [LS11]
2013		Ito et al. [IPSA13]
2014	Running Star	Wan [Wan14]
2014		Taylor [Tay14]
2016 (2020)		Chiba et al. [CSA16, CSA20]
2017	Open Soda Constructor	Fidelman [Fid17]

Table 5: *Overview of papers discussed in the Simulations and Ecosystems section*

energy dies. If the energy level is above a certain level, it starts looking for a mate. Mating criteria can be set interactively to allow for easy experimentation. Morphologically a swimbot can have from two to ten parts, and each part can have individual length, thickness, colour and resting angle. These attributes are genetically determined. Likewise, for behaviour, swimbots move via a harmonic function, which is controlled by phase and amplitude arguments, which are also represented as genes.

Sadly no longer available online, Soda Constructor [Bur00] was a playful web-based tool where the user could create a 2D spring-based model inside of a small, constrained area. The model consisted solely of vertices with springs between them. The vertices collide with the environment, giving the creature the ability to move. The simulator also featured gravity. A single harmonic function was used for controlling the movement. The function and its impact on individual vertices could be customised. Several simulators inspired by the Soda Constructor, such as the Open Constructor [Fid17] later followed.

In the Sticky Feet [Tur10] simulation created by Turk, a 2D world is inhabited by spring-mass based creatures. The creatures move through oscillating the resting length of springs and changing the friction of the endpoints of the springs. Endpoints can be connected to any number of springs, and there is no internal collision between springs, points or springs and points. Points have a radius, so they are, in fact, small discs. Animats in Sticky Feet have two special points; a heart and a mouth. When the mouth of a creature *A* comes within a specific radius of the heart of another creature *B*, *B* dies, and another creature *C*, created with a single ancestor *A*, is born with the chance of having one or more mutations. In this way, there is a constant sized population of creatures, and the fitness function is simply how long an animat survives in the simulation. Each spring may also have a sensor attached, which can affect the modulation of the spring and is pre-configured to detect either hearts or mouths. Sensors add behaviour to the creatures so they can detect when they are close to prey or start evading when they detect another creature is near. The world is seen from above as if looking down on creatures moving around on a piece of paper.

Running Star [Wan14] is a web application for teaching understanding of evolution using artificial creatures. In it, creatures composed of boxes and wheels evolve to traverse an obstacle course. As the morphology of the creatures evolves, the creatures make it further and further in the level. Everything is shown in real-time. The user can add extra obstacles in the form of boxes and configure additional parameters such as simulation time.

Taylor [Tay14] is an inspiring read for anyone who is into simulating virtual creatures, as the author questions why no-one has managed to reach a more open-ended evolutionary system with a more diverse complexity. The author argues that it is not simply the lack of scale of current simulations that is the problem (Taylor [Tay14] mentions the number 5000 as the average population size for a viable MVP). While an insufficient population size will cause problems such as inbreeding and lack of diversity, the author argues that the logic of current simulations might simply not be sufficient. For example, most simulations focus only on vertical gene transfer and an evolutionary hierarchy. Horizontal gene transfer and ecological hierarchies are rarely simulated. However, the larger issue is that simulations can only model what has been programmed in by the designer. It is not possible to break the boundaries of the program.

Chiba et al. [CSA16] is one of the few papers that we have found that delves into the interplay between ecological and organism inheritance; both the organism and the environment evolve over time, but there is also an interplay between the two at each generational step. Ecological inheritance will also affect morphology. In that work, organisms are able to create boxes and planks that can help traverse ravines and valleys, by for example, using those as stepping stones or ramps. The authors set up three experiments: one where there is no ecological inheritance (all planks and boxes stay), almost complete ecological inheritance (0.01 chance of individual planks and boxes disappearing due to "weathering" effects), and one with unstable ecological inheritance (0.1 chance of elements disappearing). The highest fitness was seen with stable ecological inheritance, while the lowest one was in the unstable environment, as it was difficult for the organisms to develop a strategy. In the stable environment, organisms would place a few new objects to replace the old elements, while in the environment with no ecological inheritance, for each generation, organisms place a lot of objects to build bridges and ramps. Chiba et al. [CSA20] expands on this work, using a deep auto-encoder to extract the features of adaptive structures.

Creating interesting and potentially surprising morphological ideas needs to combine the often opposing goals of a high degree of diversity, as well as usable high fitness solutions. Lehman & Stanley [LS11] tackles the problem of early convergence by focusing on how to increase and maintain diversity using a Pareto-based Multi-objective Evolutionary Algorithm (MOEA). The basic inspiration is that in nature, different niches are often not competing. For example, bears and bacteria do not compete directly against each other. Therefore to maintain diversity, the focus should be on local competition within each niche, as global competition will ultimately lead to the destruction of morphological and behavioural niches during convergence. The authors suggest creating a 3D morphological space through creature height, mass and number of active joints. In this space, the results show that while local competition gives a lower overall fitness than global competition, novelty search exploits morphological space better.

Based on the Sims [Sim94] inspired Creature Academy framework [PJ08], Ito et al. [IPSA13] answer the question: what comes first, morphology or behaviour? The authors show that morphological changes tend to precede behavioural changes when introducing a new strategy, while the reverse applies as a response to strategy changes. They then conjecture this is because morphological changes tend to be more drastic than behavioural changes, so a change in morphology will also require a following change in behaviour.

14. Comparisons between Genotype Representations

Several works compare efficiency between different genotype representations, the latest we have studied being by Veenstra & Glette [VG20]. They compared four different genotype encodings as they optimised the locomotion of 2D virtual creatures. The chosen representations were a direct encoding, L-systems (§2.2 and §4), and two neural network-based representations: CPPNs (§2.6) and Cellular Encoding (CE) [Gru93]. The authors used the Distributed Evolutionary Algorithms in the Python framework (DEAP) [F*12] for evolving their population, regardless of their genotype representation. They also use DEAP for evolving their CPPNs, which is contrary to most other uses of CPPN studied in this review, as often NEAT [SM02] is combined with CPPN for evolving the networks. The controller is based on a harmonic wave function. Box2D [Cat06] was used for the simulation environment. The authors find that the direct and the L-system representations converge faster, while L-systems and the network-based representations made larger leaps across the search space.

Veenstra et al. (§4.2) made a comparison between two representations: a direct and indirect L-system. They found no significant difference in performance between the two, except for smaller robots with five modules. For bigger robots, the indirect representation only had initial faster convergence, but the direct representation would catch up in later generations.

In Framsticks (§4.1), Komosiński & Rotaru-Varga compared the performance of three different genotype representations [KRV01]: a direct representation, a direct recursive representation and an indirect representation. They found no significant difference in performance between these, except for a single test, where the direct recursive and indirect representations did better than the direct representation (details in §4.1). In other tests, indirect representations do better than direct ones. Works comparing the performance of genotype representations is shown in Table 6. Overall, it does not seem to be possible to conclude which representation is best. CPPN has generally done well for 3D soft-body simulations, while Veenstra & Glette [VG20] show that a direct representation outperforms the CPPN network in a 2D rigid-body simulation.

15. Conclusion

We have presented a swathe of different works, stemming from a diverse set of disciplines, all with the commonality that they focus on the generation of virtual creatures (other than human forms which would require their own survey. §1.1 describes the framework used for this survey.). In an attempt to give a more cohesive presentation, these works have been divided into three main groups: Art, Games and ALife. Games and ALife works are often at opposite ends of a continuum, where games tend to focus on the outcome of algorithms and the desires of the creating agent. In ALife, the focus is on the process of evolution and development — the aesthetics of the final creation is less important as long as the fitness function has been optimised. Artworks tend to exist in a continuum between the two, as many artists are interested both in the process and having some influence on the aesthetics of their creation.

Most works focus on the development of individual animats, with crossover being the only social element (assuming the authors don't only use mutation). However, social interaction and competition also affect evolution, so some works take a more open-ended evolutionary approach by modelling environments where creatures have to compete for resources and where they can influence each other and the environment. We have therefore included a section (§13) on Simulations and Ecosystems relevant for the generation of virtual creatures.

Looking towards the future of virtual creature development, the clearest one seems to be in art, games and entertainment, as recent ALife methods involving animat morphology instead tend to focus on applications in physical robotics. On the games industry side, applications such as Houdini [Inc96] from SideFX are empowering creators with easy access to generative methods. Additionally, looking at machine learning, there is a possible trajectory where a computational agent can become either the sole creator or a col-

Year	*Section*	*Authors*	*Tested genotype representations*	*Best representations*	*Dim*	*Phenotype*
2001	4.1	Komosiński & Rotaru-Varga. [KRV01]	Direct, direct recursive, L-system	Direct recursive and indirect in one test, otherwise no significant difference.	3D	Rigid-body
2010	8.1	Hiller et al. [HL10]	DCT, CPPN & GMX	GMX	3D	Soft-body
2011	7	Auerbach & Bongard [AB11]	Non-recurrent CPPN & Recurrent CPPN	Recurrent CPPN	3D	Rigid-body
2013	8.2	Cheney et al. [CMCL13]	Direct & CPPN-NEAT	CPPN-NEAT	3D	Soft-body
2017	4.2	Veenstra et al. [VFRS17]	Direct, L-system	L-system for smaller robots and few generations.	3D	Rigid-body
2020	14	Veenstra & Glette [VG20]	Direct, L-system, CPPN, CE	Direct and L-system	2D	Rigid-body

Table 6: *Works that compare the performance of genotype representations*

league when creating virtual creatures for video games. Whether more advanced methods from ALife, such as CPPNs, will be experimented with by games developers will depend on designers willingness to give up authorial control in exchange for a variety of options. Alternatively, it may be that computational power and algorithmic improvements increase to a point where designers can use these advanced techniques to directly create or tweak their creations without waiting or losing authorial control [LLL20].

One domain that could offer interesting new perspectives is from developmental biology itself. Up until now, the main influence in computing has been somewhat simple approximations of how genes may encode diversity and provide metaphors for simulating evolutionary differentiation — with computational methods such as provided by L-systems and genetic programming. Recent works in biology point into the direction of other relevant encodings and environmental influences. This calls for a study of recent understanding in epigenetics — how phenotype modifications can be carried on by environmental conditions without involving genes per se. In particular, there has been continuous progress in recent years in the study of how various environmental conditions within (molecular level) and outside cells (assemblies), such as provided by (non-neural) bioelectric force fields and gradients, directly modulate the development of phenotypes – a line of work which has its origin in the last paper by Turing [Tur52]. Such studies in biology have also recognised the need to integrate recent progress in computing within their own discipline, including deep (learning) generative methods, predictive coding or information theory [PL15,NMRL21]. We thus expect a greater exchange of ideas and principles between biology and computing to take place in the near future and having a strong influence on the further development of the field of virtual creature morphology.

Moving into a more high-level perspective, Kriegman [Kri19] writes in Nature's Machine Intelligence that virtual creatures "are interesting objects of scientific investigation in their own right. More than that, they have the potential to be as beautiful and complex as life itself."

Acknowledgements

This work was funded in part by the EPSRC Centre for Doctoral Training in Intelligent Games & Game Intelligence (IGGI) EP/L015846/1.

References

[AB10a] AUERBACH J. E., BONGARD J. C.: Dynamic Resolution in the Co-Evolution of Morphology and Control. In *Artificial Life XII. Proceedings of the 12th International Conference on the Synthesis and Simulation of Living Systems* (May 2010), pp. 451–458. 9, 10, 15

[AB10b] AUERBACH J. E., BONGARD J. C.: *Evolving CPPNs to grow three-dimensional physical structures.* In *Proceedings of the 12th annual conference on Genetic and evolutionary computation - GECCO '10* (New York, New York, USA, 2010), ACM Press, p. 627. doi:10.1145/1830483.1830597. 9, 10, 15

[AB11] AUERBACH J. E., BONGARD J. C.: Evolving complete robots with CPPN-NEAT. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation - GECCO '11* (2011), ACM Press, p. 1475. doi:10.1145/2001576.2001775. 9, 10, 15, 18

[AB12] AUERBACH J. E., BONGARD J. C.: On the relationship between environmental and morphological complexity in evolved robots. In *Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference - GECCO '12* (New York, New York, USA, 2012), ACM Press, p. 521. doi:10.1145/2330163.2330238. 10, 15

[ACBS*17] AKROUR D., CUSSAT-BLANC S., SANCHEZ S., DJEDI N., LUGA H.: Joint evolution of morphologies and controllers for realistic modular robots. In *22nd Symposium on Artificial Life And Robotics (AROB 2017)* (2017), Springer Japan, pp. 57–62. URL: https://oatao.univ-toulouse.fr/18944/. 2

[AJI*16] ARITA T., JOACHIMCZAK M., ITO T., ASAKURA A., SUZUKI R.: ALife approach to eco-evo-devo using evolution of virtual creatures. *Artificial Life and Robotics 21*, 2 (2016), 141–148. doi:10.1007/s10015-016-0278-5. 2

[Art00] ARTWORKS C.: *Evolva*. Game [Microsoft Windows], June 2000. Virgin Interactive. 13

[ASA15] ASAKURA A., SUZUKI R., ARITA T.: Evolving 3D virtual creatures through exaptation triggered by environmental change. *Artificial Life and Robotics 20*, 3 (2015), 244–250. doi:10.1007/s10015-015-0222-0. 2, 6

[Aut85] AUTODESK: *StudioTools*, 1985. 14

[AvdP16] AGRAWAL S., VAN DE PANNE M.: Task-based locomotion. *ACM Transactions on Graphics 35*, 4 (2016), 1–11. doi:10.1145/2897824.2925893. 2

[B*89] BURRIDGE J. M., ET AL.: The WINSOM solid modeller and its application to data visualization. *IBM Systems Journal 28*, 4 (1989), 548–568. doi:10.1147/sj.284.0548. 4

[BC05] BRUDERLIN A., CALVERT T. W.: Goal-directed, dynamic animation of human walking. *ACM SIGGRAPH Computer Graphics 23*, 3 (2005), 233–242. doi:10.1145/74334.74357. 2

[BD91] BBC, DAWKINS R.: Richard Dawkins discusses biomorphs in 1991, 1991. Accessed: 2020-02-14. URL: https://www.bbc.co.uk/news/av/science-environment-36367745/richard-dawkins-discusses-biomorphs-in-1991. 3

[Bee06] BEER R. D.: Parameter space structure of continuous-time recurrent neural networks. *Neural Computation 18*, 12 (2006), 3009–3051. doi:10.1162/neco.2006.18.12.3009. 10

[BP03] BONGARD J. C., PFEIFER R.: Evolving Complete Agents using Artificial Ontogeny. In *Morpho-functional Machines: The New Species*. Springer Japan, Tokyo, 2003, pp. 237–258. doi:10.1007/978-4-431-67869-4_12. 9, 15

[Bra84] BRAITENBERG V.: *Vehicles: Experiments in Synthetic Psychology*. MIT Press, Cambridge, MA, 1984. 2

[Bur00] BURTON E.: Soda constructor, april 2000. Accessed: 2020-06-05. URL: https://en.wikipedia.org/wiki/Soda_Constructor. 16

[C*07] COMPTON K., ET AL.: Creating spherical worlds. In *ACM SIGGRAPH 2007 sketches on - SIGGRAPH '07* (2007), no. sap 0251, ACM Press, pp. 82–es. doi:10.1145/1278780.1278879. 13

[C*16] CORUCCI F., ET AL.: Evolving swimming soft-bodied creatures. In *ALIFE XV, The Fifteenth International Conference on the Synthesis and Simulation of Living Systems, Late Breaking Proceedings* (apr 2016). 2, 10, 11, 15

[Cat06] CATTO E.: Box2D, 2006. Accessed: 2020-06-16. URL: https://box2d.org/. 15, 17

[CBOP09] CLUNE J., BECKMANN B. E., OFRIA C., PENNOCK R. T.: Evolving coordinated quadruped gaits with the HyperNEAT generative encoding. In *2009 IEEE Congress on Evolutionary Computation* (may 2009), IEEE, pp. 2764–2771. doi:10.1109/CEC.2009.4983289. 10

[CCB] Attribution 4.0 international (cc by 4.0). Accessed: 2020-03-26. URL: https://creativecommons.org/licenses/by/4.0/. 11, 12

[CCGS*18] CORUCCI F., CHENEY N., GIORGIO-SERCHI F., BONGARD J., LASCHI C.: Evolving Soft Locomotion in Aquatic and Terrestrial Environments: Effects of Material Properties and Environmental Transitions. *Soft Robotics 5*, 4 (aug 2018), 475–495. arXiv:1711.06605, doi:10.1089/soro.2017.0055. 2, 6, 10, 11, 15

[CCK*17] CORUCCI F., CHENEY N., KRIEGMAN S., BONGARD J., LASCHI C.: Evolutionary Developmental Soft Robotics As a Framework to Study Intelligence and Adaptive Behavior in Animals and Plants. *Frontiers in Robotics and AI 4* (July 2017), 34. doi:10.3389/frobt.2017.00034. 10

[CEA95] CHAUMONT N., EGLI R., ADAMI C.: Evolution of Virtual Catapults. 1–6. URL: https://www.researchgate.net/publication/228685505. 2

[CEA07] CHAUMONT N., EGLI R., ADAMI C.: Evolving virtual creatures and catapults. *Artificial Life 13*, 2 (2007), 139–157. doi:10.1162/artl.2007.13.2.139. 2

[CGKR21] CHENEY N., GRASSO C., KRIEGMAN S., RISI S.: The virtual creatures competition, 2021. Accessed: 2021-02-22. URL: https://virtualcreatures.github.io/. 1

[Cha19] CHAN B. W.-C.: Lenia: Biology of Artificial Life. *Complex Systems 28*, 3 (oct 2019), 251–286. arXiv:arXiv:1812.05433v3, doi:10.25088/ComplexSystems.28.3.251. 8, 13, 15, 16

[Cha20a] CHAN B. W.-C.: Lenia and Expanded Universe. In *The 2020 Conference on Artificial Life* (Cambridge, MA, 2020), MIT Press, pp. 221–229. arXiv:2005.03742, doi:10.1162/isal_a_00297. 8, 13, 15, 16

[Cha20b] CHAN B. W.-C.: Original lenia, Oct 2020. Accessed: 2021-03-26. URL: https://github.com/Chakazul/Lenia/releases/tag/v3.0. 8

[Cho56] CHOMSKY N.: Three models for the description of language. *IEEE Transactions on Information Theory 2*, 3 (sep 1956), 113–124. doi:10.1109/TIT.1956.1056813. 3

[CM16] CULLY A., MOURET J.-B.: Evolving a Behavioral Repertoire for a Walking Robot. *Evolutionary Computation 24*, 1 (mar 2016), 59–88. URL: http://www.mitpressjournals.org/doi/10.1162/EVCO_a_00143, doi:10.1162/EVCO_a_00143. 2

[CMCL13] CHENEY N., MACCURDY R., CLUNE J., LIPSON H.: Unshackling evolution. In *Proceeding of the fifteenth annual conference on Genetic and evolutionary computation conference - GECCO '13* (2013), pp. 167–174. doi:10.1145/2463372.2463404. 2, 10, 11, 12, 15, 18

[COM13] COMPTON K., OSBORN J., MATEAS M.: Generative methods. In *Proceedings of the 4th Workshop on Procedural Content Generation in Games* (2013). URL: http://www.fdg2013.org/program/workshops/papers/PCG2013/pcg2013_6.pdf. 13

[Com15] COMFORT N.: Genetics: Dawkins, redux. *Nature 525*, 7568 (2015), 184–185. doi:10.1038/525184a. 4

[Com16] COMPTON K.: So you want to build a generator..., February 2016. Accessed: 2019-07-10. URL: https://galaxykate0.tumblr.com/post/139774965871/so-you-want-to-build-a-generator. 13

[Com17] COMPTON K.: Practical procedural generation for everyone, 2017. Accessed: 2020-05-19. URL: https://www.youtube.com/watch?v=WumyfLEa6bU&t=1095. 13, 15

[Cor] CORPORATION N.: Physx. Accessed: 2019-07-19. URL: https://developer.nvidia.com/physx-sdk. 6, 15

[Cou] COUMANS E. E. A.: Bullet real-time physics simulation. Accessed: 2019-07-03. URL: http://bulletphysics.org. 9, 15

[CSA16] CHIBA N., SUZUKI R., ARITA T.: How ecological inheritance can affect the evolution of complex niche construction in a 2D physical simulation. In *Proceedings of the Artificial Life Conference 2016* (Cambridge, MA, 2016), MIT Press, pp. 426–433. doi:10.7551/978-0-262-33936-0-ch070. 16

[CSA20] CHIBA N., SUZUKI R., ARITA T.: Evolution of Complex Niche-Constructing Behaviors and Ecological Inheritance of Adaptive Structures in a Physically Grounded Environment. *Frontiers in Robotics and AI 7*, April (2020), 1–13. doi:10.3389/frobt.2020.00045. 16, 17

[Dar08] DARWIN C.: *On the Origin of Species*. OUP Oxford; Revised edition, November 2008. 2

[Daw] DAWKINS R.: Watchmaker. Accessed: 2019-10-01. URL: https://aronnax9000.github.io/WatchmakerSuite/. 3, 4

[Daw86] DAWKINS R.: *The Blind Watchmaker*. Norton & Company Inc, 1986. 2, 3, 4, 13, 15

[Daw15] DAWKINS R.: *Brief Candle in the Dark: My Life in Science*. Bantam Press, 2015. 4

[Des96] DESIGN B.: *Fusion*, November 1996. 14

[DGH*07] DEBRY D. G., GOFFIN H., HECKER C., QUIGLEY O., SHODHAN S., WILLMOTT A.: Player-driven procedural texturing. In *ACM SIGGRAPH 2007 sketches* (2007), no. sketches 0311, pp. 81–es. doi:10.1145/1278780.1278878. 13

[Egg97] EGGENBERGER P.: Evolving Morphologies of Simulated 3d Organisms Based on Differential Gene Expression. In *Proceedings of the 4th European Conference on Artificial Life (ECAL97)* (1997), pp. 205–213. 9

[Ent03] ENTERTAINMENT R.: *Impossible Creatures*. Game [Microsoft Windows], January 2003. Microsoft Game Studios. 13

[F*12] FORTIN F. A., ET AL.: DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research 13*, 1 (2012), 2171–2175. 17

[FBLPD13] FAÍÑA A., BELLAS F., LÓPEZ-PEÑA F., DURO R. J.: Edhmor: Evolutionary designer of heterogeneous modular robots. *Engineering Applications of Artificial Intelligence 26*, 10 (2013), 2408–2423. doi:10.1016/j.engappai.2013.09.009. 8

[Fid17] FIDELMAN P.: Open constructor, 2017. Accessed: 2020-06-05. URL: https://github.com/OpenConstructor/OpenConstructor. 16

[FMN94] FUKUNAGA A., MARKS J., NGO J. T.: Automatic Control of Physically Realistic Animated Figures Using Evolutionary Programming. *Proceedings 3rd Annual Conference on Evolutionary Programming* (1994), 76–83. 2

[Fod08] FODDY B.: *QWOP*. Game [Web, iOS, Android], 2008. Foddy, Bennett. 6

[Fog06] FOGEL D.: Historic perspective - Nils Barricelli-artificial life, coevolution, self-adaptation. *IEEE Computational Intelligence Magazine 1*, 1 (feb 2006), 41–45. doi:10.1109/MCI.2006.1597062. 3, 15

[FPC01] FREDERIC T., PHILIPPE M., CHRISTOPHE C.: Fast Polygonization of Implicit Surfaces. In *Proceedings of the 9th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision* (2001). 13

[FT] FAUCONNIER G., TURNER M.: Conceptual Integration Networks. *Cognitive Science 22*, 2, 133–187. URL: http://doi.wiley.com/10.1207/s15516709cog2202_1, doi:10.1207/s15516709cog2202_1. 12

[Gam12] GAMES N.: *Incredipede*. Game [Microsoft Windows, Mac OSX, iOS, Android], 2012. Northway Games. 6

[Gam16] GAMES H.: *No Man's Sky*. Game [PlayStation 4, Microsoft Windows, Xbox One], August 2016. Sony Entertainment and Hello Games. 13

[Gar70] GARDNER M.: The fantastic combinations of John Conway's new solitaire game. *Scientific American 223* (1970), 120–123. 3, 8, 15

[GCH*16] GEHRING C., COROS S., HUTLER M., DARIO BELLICOSO C., HEIJNEN H., DIETHELM R., BLOESCH M., FANKHAUSER P., HWANGBO J., HOEPFLINGER M., SIEGWART R.: Practice Makes Perfect: An Optimization-Based Approach to Controlling Agile Motions for a Quadruped Robot. *IEEE Robotics and Automation Magazine 23*, 1 (2016), 34–43. doi:10.1109/MRA.2015.2505910. 2

[Gin03] GINGOLD C.: Miniature Gardens & Magic Crayons : Games , Spaces , & Worlds, 2003. URL: http://levitylab.com/cog/writing/thesis/. 13

[Gin07] GINGOLD C.: SPORE's Magic Crayons, 2007. Accessed: 2019-07-10. URL: https://www.gdcvault.com/play/527/SPORE-s-Magic. 13

[Goo13] GOOGLE: Liquidfun, 2013. Accessed: 2019-10-11. URL: https://google.github.io/liquidfun. 11, 15

[GP12] GEIJTENBEEK T., PRONOST N.: Interactive Character Animation Using Simulated Physics: A State-of-the-Art Review. *Computer Graphics Forum 31*, 8 (dec 2012), 2492–2515. doi:10.1111/j.1467-8659.2012.03189.x. 2

[Gru93] GRUAU F.: Cellular encoding as a graph grammar. *IEEE Colloquium on Grammatical Inference: Theory, Applications and Alternatives* (1993), 1701–1710. 17

[GS69] GABRIEL K. R., SOKAL R. R.: A New Statistical Approach to Geographic Variation Analysis. *Systematic Zoology 18*, 3 (sep 1969), 259. doi:10.2307/2412323. 9

[GvdPvdS13] GEIJTENBEEK T., VAN DE PANNE M., VAN DER STAPPEN A. F.: Flexible muscle-based locomotion for bipedal creatures. *ACM Transactions on Graphics 32*, 6 (2013), 1–11. URL: http://dl.acm.org/citation.cfm?doid=2508363.2508399, doi:10.1145/2508363.2508399. 2

[H*08] HECKER C., ET AL.: Real-time motion retargeting to highly varied user-created morphologies. *ACM Transactions on Graphics 27*, 3 (2008), 1. doi:10.1145/1360612.1360626. 13, 15

[Hag91] HAGGERTY M.: Evolution by Esthetics, an Interview with W. Latham and S. Todd. *IEEE Computer Graphics and Applications 11*, 2 (1991), 5–9. 4

[Hay99] HAYLES N. K.: Simulating Narratives: What Virtual Creatures Can Teach Us. *Critical Inquiry 26*, 1 (oct 1999), 1–26. doi:10.1086/448950. 1

[Hec10] HECKER C.: My Liner Notes for Spore, 2010. Accessed: 2019-07-05. URL: http://chrishecker.com/My_Liner_Notes_for_Spore. 13, 15

[HI97] HILTON A., ILLINGWORTH J.: Marching Triangles: Delunay Implicit Surface Triangulation, 1997. 13

[Hila] HILLER J.: Voxcad. Accessed: 2019-11-21. URL: https://github.com/jonhiller/VoxCAD. 10, 11, 12, 15

[Hilb] HILLER J.: Voxelyze. Accessed: 2019-12-30. URL: https://github.com/jonhiller/Voxelyze. 10, 11, 15

[Hil90] HILLIS W.: Co-evolving parasites improve simulated evolution as an optimization procedure. *Physica D: Nonlinear Phenomena 42*, 1-3 (jun 1990), 228–234. URL: https://linkinghub.elsevier.com/retrieve/pii/016727899090076 2, doi:10.1016/0167-2789(90)90076-2. 1

[HKS17] HOLDEN D., KOMURA T., SAITO J.: Phase-functioned neural networks for character control. *ACM Transactions on Graphics 36*, 4 (jul 2017), 1–13. doi:10.1145/3072959.3073663. 2

[HL10] HILLER J. D., LIPSON H.: Evolving Amorphous Robots. In *Alife 12* (sep 2010), Springer, pp. 717—-724. 2, 10, 15, 18

[HL14] HILLER J., LIPSON H.: Dynamic Simulation of Soft Multimaterial 3D-Printed Objects. *Soft Robotics 1*, 1 (mar 2014), 88–101. doi:10.1089/soro.2013.0010. 10, 11

[HLP01] HORNBY G. S., LIPSON H., POLLACK J.: Evolution of generative design systems for modular physical robots. In *Proceedings IEEE International Conference on Robotics and Automation (ICRA)* (2001), vol. 4, IEEE, pp. 4146–4151. doi:10.1109/ROBOT.2001.933266. 7, 15

[HP01] HORNBY G. S., POLLACK J.: Body-Brain Co-evolution Using L-systems as a Generative Encoding. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001)* (2001), no. December, pp. 868–875. 7, 15

[Inc96] INC. S. E. S.: *Houdini*, 1996. Accessed: 2019-07-25. URL: https://www.sidefx.com/products/houdini/. 14, 16, 17

[IPSA13] ITO T., PILAT M. L., SUZUKI R., ARITA T.: ALife approach for body-behavior predator-prey coevolution: Body first or behavior first? *Artificial Life and Robotics 18*, 1-2 (2013), 36–40. doi:10.1007/s10015-013-0096-y. 1, 6, 16, 17

[JBT*18] JULIANI A., BERGES V.-P., TENG E., COHEN A., HARPER J., ELION C., GOY C., GAO Y., HENRY H., MATTAR M., LANGE D.: Unity: A General Platform for Intelligent Agents. *arXiv* (sep 2018), 1–28. URL: http://arxiv.org/abs/1809.02627, arXiv:1809.02627. 15

[JKDW13a] JOACHIMCZAK M., KOWALIW T., DOURSAT R., WROBEL B.: Controlling development and chemotaxis of soft-bodied multicellular animats with the same gene regulatory network. In *Advances in Artificial Life, ECAL 2013* (sep 2013), MIT Press, pp. 454–461. doi:10.7551/978-0-262-31709-2-ch065. 9, 15

[JKDW13b] JOACHIMCZAK M., KOWALIW T., DOURSAT R., WRÓBEL B.: Evolutionary design of soft-bodied animats with decentralized control. *Artificial Life and Robotics 18*, 3-4 (2013), 152–160. doi:10.1007/s10015-013-0121-1. 2, 9, 15

[JKSA15] JOACHIMCZAK M., KAUR R., SUZUKI R., ARITA T.: Bringing drawings to life: Evolving distributed controllers for hand-drawn soft-bodied robots. In *International Symposium on Swarm Behavior and Bio-Inspired Robotics* (2015), vol. 8, pp. 381–388. doi:10.13140/RG.2.1.4154.0085. 9, 15

[JPH19] JOHNSON C., PHILIPPIDES A., HUSBANDS P.: Simulating Soft-Bodied Swimmers with Particle-Based Physics. *Soft Robotics 6*, 2 (2019), 263–275. doi:10.1089/soro.2018.0027. 2, 11, 15

[JSA14] JOACHIMCZAK M., SUZUKI R., ARITA T.: Fine Grained Artificial Development for Body-Controller Coevolution of Soft-Bodied Animats. In *Artificial Life 14: Proceedings of the Fourteenth International Conference on the Synthesis and Simulation of Living Systems* (jul 2014), The MIT Press, pp. 239–246. doi:10.7551/978-0-262-32621-6-ch040. 9, 15

[JW11] JOACHIMCZAK M., WRÓBEL B.: Evolution of the morphology and patterning of artificial embryos: Scaling the tricolour problem to the third dimension. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 5777 LNAI*, PART 1 (2011), 35–43. doi:10.1007/978-3-642-21283-3_5. 9, 15

[JW12] JOACHIMCZAK M., WRÓBEL B.: Co-evolution of morphology and control of soft-bodied multicellular animats. In *Proceedings of the 14th ACM Annual Conference on Genetic and Evolutionary Computation* (2012), no. July, pp. 561–568. doi:10.1145/2330163.2330243. 2, 9, 15

[Kaw82] KAWAGUCHI Y.: A morphological study of the form of nature. In *Proceedings of the 9th annual conference on Computer graphics and interactive techniques - SIGGRAPH '82* (1982), ACM Press, pp. 223–232. doi:10.1145/800064.801284. 3, 13, 16

[KBLB20] KRIEGMAN S., BLACKISTON D., LEVIN M., BONGARD J.: A scalable pipeline for designing reconfigurable organisms. *Proceedings of the National Academy of Sciences 117*, 4 (2020), 1853–1859. URL: https://www.pnas.org/content/117/4/1853, arXiv:https://www.pnas.org/content/117/4/1853.full.pdf, doi:10.1073/pnas.1910837117. 10, 12, 15

[KCB18] KRIEGMAN S., CHENEY N., BONGARD J.: How morphological development can guide evolution. *Scientific Reports 8*, 1 (dec 2018), 13934. URL: http://www.nature.com/articles/s41598-018-31868-7, arXiv:1711.07387, doi:10.1038/s41598-018-31868-7. 10, 11, 12, 15

[KGV83] KIRKPATRICK S., GELATT C. D., VECCHI M. P.: Optimization by Simulated Annealing. *Science 220*, 4598 (may 1983), 671–680. doi:10.1126/science.220.4598.671. 4

[KK18] KOZLOVA E., KOZLOV D.: Twitter account, march 2018. Accessed: 2020-02-05. URL: https://twitter.com/Kozinarium. 14, 16

[Kle03] KLEIN J.: BREVE: A 3D Environment for the Simulation of Decentralized Systems and Artificial Life. In *Proceedings of the Eighth International Conference on Artificial Life* (2003), pp. 329–334. doi:10.1.1.4.9943. 5, 15

[Koz15] KOZLOV D.: On wings, tails and procedural modeling, april 2015. Accessed: 2020-02-05. URL: http://www.the-working-man.org/2015/04/on-wings-tails-and-procedural-modeling.html. 16

[Koz17] KOZLOV D.: Procedural content creation f.a.q. - project aero, houdini and beyond, april 2017. Accessed: 2020-02-05. URL: http://www.the-working-man.org/2017/04/procedural-content-creation-faq-project.html. 16

[Koz18] KOZLOV D.: Procedural content creation f.a.q. - project aero, houdini and beyond, april 2018. Accessed: 2020-02-05. URL: http://www.the-working-man.org/2018/04/procedural-bestiary-and-next-generation.html. 14

[KP11] KIM J., POLLARD N. S.: Direct control of simulated nonhuman characters. *IEEE Computer Graphics and Applications 31*, 4 (2011), 56–65. doi:10.1109/MCG.2011.58. 2

[Kra08] KRAH P.: Towards efficient evolution of morphology and control. In *Proceedings of the 10th ACM Genetic and Evolutionary Computation Conference (GECCO)* (2008), p. 287. doi:10.1145/1389095.1389142. 2

[Krč07] KRČAH P.: Evolving virtual creatures revisited. In *Proceedings of the 9th ACM Genetic and Evolutionary Computation Conference (GECCO)* (2007). doi:10.1145/1276958.1284700. 2

[Kri19] KRIEGMAN S.: Why virtual creatures matter. *Nature Machine Intelligence 1*, 10 (oct 2019), 492–492. doi:10.1038/s42256-019-0102-8. 1, 18

[KRV01] KOMOSIŃSKI M., ROTARU-VARGA A.: Comparison of Different Genotype Encodings for Simulated Three-Dimensional Agents. *Artificial Life 7*, 4 (oct 2001), 395–418. doi:10.1162/106454601317297022. 7, 15, 17, 18

[KUa] KOMOSIŃSKI M., ULATOWSKI S.: Framsticks. Accessed: 2019-07-23. URL: http://www.framsticks.com/. 7, 14

[KUb] KOMOSIŃSKI M., ULATOWSKI S.: Framsticks Manual. Accessed: 2019-07-24. URL: http://www.framsticks.com/files/common/Framsticks_Manual.pdf. 7

[L*20] LEHMAN J., ET AL.: The Surprising Creativity of Digital Evolution: A Collection of Anecdotes from the Evolutionary Computation and Artificial Life Research Communities. *Artificial Life 26*, 2 (may 2020), 274–306. arXiv:1803.03453, doi:10.1162/artl_a_00319. 1

[Lat89] LATHAM W.: *Form Synth: The Rule-based Evolution of Complex Forms from Geometric Primitives*. Springer New York, New York, NY, 1989, pp. 80–108. doi:10.1007/978-1-4612-4538-4_7. 4

[LC87] LORENSEN W. E., CLINE H. E.: Marching cubes: A high resolution 3D surface construction algorithm. In *Proceedings of the 14th ACM Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)* (1987), vol. 21, pp. 163–169. doi:10.1145/37401.37422. 10, 13

[LFM13] LESSIN D., FUSSELL D., MIIKKULAINEN R.: Open-ended behavioral complexity for evolved virtual creatures. In *Proceeding of the fifteenth annual conference on Genetic and evolutionary computation conference - GECCO '13* (New York, New York, USA, 2013), ACM Press, p. 335. doi:10.1145/2463372.2463411. 2, 6, 7, 15

[LFM14a] LESSIN D., FUSSELL D., MIIKKULAINEN R.: Adopting Morphology to Multiple Tasks in Evolved Virtual Creatures. In *Artificial Life 14: Proceedings of the Fourteenth International Conference on the Synthesis and Simulation of Living Systems* (jul 2014), The MIT Press, pp. 247–254. doi:10.7551/978-0-262-32621-6-ch041. 2, 6, 15

[LFM14b] LESSIN D., FUSSELL D., MIIKKULAINEN R.: Trading control intelligence for physical intelligence: Muscle drives in evolved virtual creatures. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)* (2014), pp. 705–712. doi:10.1145/2576768.2598290. 6, 15

[LFMR15] LESSIN D., FUSSELL D., MIIKKULAINEN R., RISI S.: Increasing Behavioral Complexity for Evolved Virtual Creatures with the ESP Method. URL: http://arxiv.org/abs/1510.07957, arXiv:1510.07957. 2, 6, 15

[Lin68] LINDENMAYER A.: Mathematical models for cellular interaction in development: Parts i and ii. *Journal of Theoretical Biology 18* (1968). 3, 15

[LLD07] LASSABE N., LUGA H., DUTHEN Y.: A new step for artificial creatures. In *Proceedings of the IEEE Symposium on Artificial Life (CI-ALife)* (2007), no. May, pp. 243–250. doi:10.1109/ALIFE.2007.367803. 2, 5, 15

[LLL20] Lai G., Latham W., Leymarie F. F.: Towards Friendly Mixed Initiative Procedural Content Generation: Three Pillars of Industry. In *International Conference on the Foundations of Digital Games (FDG)* (sep 2020), ACM, pp. 1–4. doi:10.1145/3402942.3402946. 18

[Lom14] Lomas A.: Cellular forms. In *ACM SIGGRAPH Studio* (2014), ACM Press. doi:10.1145/2619195.2656282. 14, 16

[LPKL14] Lee Y., Park M. S., Kwon T., Lee J.: Locomotion control for many-muscle humanoids. *ACM Transactions on Graphics 33*, 6 (2014), 1–11. doi:10.1145/2661229.2661233. 2

[LPY16] Liu L., Panne M. V. D., Yin K.: Guided Learning of Control Graphs for Physics-Based Characters. *ACM Transactions on Graphics 35*, 3 (2016), 1–14. doi:10.1145/2893476. 2

[LR15] Lessin D., Risi S.: Darwin's Avatars: A Novel Combination of Gameplay and Procedural Content Generation. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)* (2015), pp. 329–336. doi:10.1145/2739480.2754749. 1, 2, 6, 15

[LS11] Lehman J., Stanley K. O.: Evolving a diversity of virtual creatures through novelty search and local competition. In *Proceedings of the 13th Genetic and Evolutionary Computation Conference (GECCO)* (2011), p. 211. arXiv:0611062, doi:10.1145/2001576.2001606. 16, 17

[Max08] Maxis: *Spore*. Game [Microsoft Windows, Mac OS X], September 2008. Electronic Arts. 1, 13, 15

[MC05] Miconi T., Channon A.: A virtual creatures model for studies in artificial evolution. In *IEEE Congress on Evolutionary Computation* (February 2005), vol. 1, pp. 565–572. doi:10.1109/CEC.2005.1554733. 2

[McC93] McCormack J.: Interactive evolution of L-system grammars for computer graphics modelling. *Complex Systems: from biology to computation* (1993), 1–7. 3

[Mei75] Meisters G. H.: Polygons Have Ears. *The American Mathematical Monthly 82*, 6 (jun 1975), 648. doi:10.2307/2319703. 13

[Mic08] Miconi T.: In Silicon No One Can Hear You Scream: Evolving Fighting Creatures. In *Lecture Notes in Computer Science*, vol. 4971 LNCS. 2008, pp. 25–36. doi:10.1007/978-3-540-78671-9_3. 2

[MMK12] Mori M., MacDorman K., Kageki N.: The Uncanny Valley [From the Field]. *IEEE Robotics & Automation Magazine 19*, 2 (jun 2012), 98–100. doi:10.1109/MRA.2012.2192811. 2

[MMU95] Magic P., Mindscape, Ubisoft: *Petz (Dogz and Catz)*. Game [Windows], 1995. PF Magic, Mindscape and Ubisoft. 13

[MS07] McLaughlin T., Sumida S. S.: The morphology of digital creatures. In *ACM SIGGRAPH 2007 courses on - SIGGRAPH '07* (New York, New York, USA, 2007), no. 1, ACM Press, p. 1. URL: http://dl.acm.org/citation.cfm?doid=1281500.1281660, doi:10.1145/1281500.1281660. 1

[Mur18] Murphy S.: Smoothlife, 2018. Accessed: 2021-03-26. URL: https://github.com/duckythescientist/SmoothLife. 8

[MW90] Meyer J.-A., Wilson S. W. (Eds.):. *Proceedings of the First International Conference on Simulation of Adaptive Behavior on From Animals to Animats* (Cambridge, MA, USA, 1990), MIT Press. 5

[Nik34] Nikolaevich Delone B.: Sur la sphère vide. A la mémoire de Georges Voronoï. *Bulletin de l'Académie des Sciences de l'URSS. Classe des sciences mathématiques et na*, 6 (1934), 793–800. 9

[NMRL21] Niklasson E., Mordvintsev A., Randazzo E., Levin M.: Self-Organising Textures. *Distill 6*, 2 (feb 2021). doi:10.23915/distill.00027.003. 18

[PBvdP16] Peng X. B., Berseth G., van de Panne M.: Terrain-adaptive locomotion skills using deep reinforcement learning. *ACM Transactions on Graphics 35*, 4 (jul 2016), 1–12. doi:10.1145/2897824.2925881. 2

[PBYV17] Peng X. B., Berseth G., Yin K., Van De Panne M.: DeepLoco: Dynamic Locomotion Skills Using Hierarchical Deep Reinforcement Learning. *ACM Transactions on Graphics 36*, 4 (2017). doi:10.1145/3072959.3073602. 2

[PC03] Pereira F. C., Cardaso A.: The Horse-Bird Creature Generation Experiment. *The Journal of the Society for the Study of Artificial Intelligence and the Simulation of Behaviour 1*, 3 (2003), 257–280. 12, 15

[PHLF03] Pollack J. B., Hornby G. S., Lipson H., Funes P.: Computer creativity in the automatic design of robots. *Leonardo 36*, 2 (2003), 115–121. doi:10.1162/002409403321554170. 2

[PJ08] Pilat M. L., Jacob C.: Creature Academy: A system for virtual creature evolution. *2008 IEEE Congress on Evolutionary Computation, CEC 2008*, May (2008), 3289–3297. doi:10.1109/CEC.2008.4631243. 2, 16, 17

[PL15] Pezzulo G., Levin M.: Re-membering the body: Applications of computational neuroscience to the top-down control of regeneration of limbs and other complex organs. *Integrative Biology 7*, 12 (2015), 1487–1517. doi:10.1039/C5IB00221D. 18

[PLC03] PLC M.: Mathengine karma, 2003. Accessed: 2019-09-19. URL: https://docs.unrealengine.com/udk/Two/rsrc/Two/KarmaReference/KarmaUserGuide.pdf. 5, 9, 15

[PLD*19] Pathak D., Lu C., Darrell T., Isola P., Efros A. A.: Learning to Control Self-Assembling Morphologies: A Study of Generalization via Modularity. In *Advances in Neural Information Processing System* (2019), Wallach H., Larochelle H., Beygelzimer A., Alchè-Buc F., Fox E., Garnett R., (Eds.), Curran Associates, Inc. 12, 13, 15

[Pru86] Prusinkiewicz P.: Graphical Applications of L-Systems. In *Proceedings - Graphics Interface* (1986), pp. 247–253. 3, 15

[Quia] Quigley O.: Spore's creature skin painting. Accessed: 2020-05-25. URL: http://oceanquigley.blogspot.com/2009/04/spores-creature-skin-painting.html. 13, 15

[Quib] Quigley O.: Spore's creature skinpaint. Accessed: 2020-05-25. URL: http://oceanquigley.blogspot.com/2009/04/spores-creature-skinpaint.html. 13, 15

[Raf11] Rafler S.: Generalization of Conway's "Game of Life" to a continuous domain - SmoothLife. 1–4. URL: http://arxiv.org/abs/1111.1567, arXiv:1111.1567. 8, 15

[Ray01] Ray T. S.: Aesthetically Evolved Virtual Pets. *Leonardo 34*, 4 (aug 2001), 313–316. URL: http://www.mitpressjournals.org/doi/10.1162/00240940152549230, doi:10.1162/00240940152549230. 2, 5, 13, 16

[Roc96] RocketScienceGames: Darwin pond, 1996. Accessed: 2020-06-06. URL: http://www.ventrella.com/Darwin/darwin.html. 14, 15, 16

[Row99] Rowbottom A.: Evolutionary Art and Form. In *Evolutionary Design by Computers*. Morgan Kaufmann Publishers Inc., 1999, ch. 3.11, pp. 261–277. 13, 16

[RPM*03] Ribeiro P., Pereira F. C., Marques B., Leitão B., Cardoso A.: A model for creativity in creature generation. *Proceedings of the 4th annual European GAMEON Conference (2003)*, January (2003), 5. URL: http://pdf.aminer.org/000/222/675/a_model_for_creativity_in_creature_generation.pdf. 12, 15

[RSF13] Rohmer E., Singh S. P. N., Freese M.: V-REP: A versatile and scalable robot simulation framework. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (nov 2013), pp. 1321–1326. doi:10.1109/IROS.2013.6696520. 8, 15

[SCBN17] Sathiya V., Chinnadurai M., Babu R. A., Nagalakshmi K.: Evolution of Nature-inspired Intelligent Robots âĂŞ A Review. *International Journal of Computer Science and Information Security 15*, 10 (2017), 257–263. 2

[SCH04] SHIM Y., CHANG-HUN K.: Evolving Flying Creatures with Path Following Behaviors. In *The 9th International Conference on the Simulation and Synthesis of Living Systems (ALIFE IX)* (May 2004), pp. 125–132. 2, 5, 6, 15

[Sim87] SIMS K.: *Locomotion of Jointed Figures over Complex Terrain*. Master's thesis, Massachusetts Institute of Technology, 1987. 5

[Sim94] SIMS K.: Evolving virtual creatures. In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1994), SIGGRAPH '94, Association for Computing Machinery, p. 15âĂŞ22. URL: https://doi.org/10.1145/192161.192167, doi:10.1145/192161.192167. 1, 2, 3, 5, 6, 7, 8, 9, 11, 13, 16, 17

[SK03] SHIM Y., KIM C.-H.: Generating flying creatures using body-brain co-evolution. In *Proceedings of the ACM SIGGRAPH/Eurographics symposium on Computer animation (SCA)* (2003), pp. 276–285. 2, 5, 6, 15

[SL10] SCHMIDT M. D., LIPSON H.: Age-fitness pareto optimization. In *Proceedings of the 12th annual conference on Genetic and evolutionary computation - GECCO '10* (New York, New York, USA, 2010), no. 2, ACM Press, p. 543. doi:10.1145/1830483.1830584. 11, 12

[SM02] STANLEY K. O., MIIKKULAINEN R.: Evolving Neural Networks through Augmenting Topologies. *Evolutionary Computation 10*, 2 (jun 2002), 99–127. doi:10.1162/106365602320169811. 9, 10, 17

[Smi] SMITH R.: Open dynamics engine. Accessed: 2019-07-16. URL: http://ode.org. 6, 10, 15

[Smi14] SMITH G.: *Procedural Everything: Playing Procedural Content Generation in Spore*. Tech. rep., 2014. 13

[Sta07] STANLEY K. O.: Compositional pattern producing networks: A novel abstraction of development. *Genetic Programming and Evolvable Machines 8*, 2 (2007), 131–162. doi:10.1007/s10710-007-9028-8. 2, 5, 9, 10

[Stu01] STUDIOS L.: *Black & White*. Game [Windows, Mac], 2001. Electronic Arts & Feral Interactive. 13

[SYK*20] SHAH D., YANG B., KRIEGMAN S., LEVIN M., BONGARD J., KRAMER-BOTTIGLIO R.: Shape Changing Robots: Bioinspiration, Simulation, and Physical Realization. *Advanced Materials 2002882* (2020), 1–12. doi:10.1002/adma.202002882. 2

[Tay14] TAYLOR T.: Evolution in Virtual Worlds. *The Oxford Handbook of Virtuality*, July (2014), 526–548. doi:10.1093/oxfordhb/9780199826162.013.044. 16

[TB20] TOOR A. S., BERTSCH F.: Using GANs to Create Fantastical Creatures, November 2020. Accessed: 2021-01-25. URL: https://ai.googleblog.com/2020/11/using-gans-to-create-fantastical.html. 1, 14, 16

[Ter95] TERZOPOULOS D.: Modeling Living Systems for Computer Vision. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence* (1995), Morgan Kaufmann Publishers Inc., pp. 1003–1013. URL: http://dl.acm.org/citation.cfm?id=1625855.1625986. 16

[Tho42] THOMPSON D. W.: *On Growth and Form*. Cambridge University Press, 1942. 2

[TL91] TODD S., LATHAM W.: *Mutator, a Subjective Human Interface for Evolution of Computer Sculptures*. Tech. rep., IBM, 1991. 4

[TL92] TODD S., LATHAM W.: *Evolutionary Art and Computers*. Academic Press, Inc., 1992. 2, 3, 4, 13, 16

[Tod21] TODD S.:. Private Communication, February 2021. 4

[TR97] TERZOPOULOS D., RABIE T.: Animat vision: Active vision in artificial animals. In *Proceedings of IEEE International Conference on Computer Vision* (1997), vol. 1, IEEE Comput. Soc. Press, pp. 801–808. doi:10.1109/ICCV.1995.466856. 2, 16

[TT00] TU X., TERZOPOULOS D.: Artificial fishes. In *Proceedings of the 28th ACM Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)* (2000), pp. 43–50. doi:10.1145/192161.192170. 2, 11, 14, 16

[Tu96] TU X.: *Artificial Animals for Computer Animation: Biomechanics, Locomotion, Perception, and Behavior*. PhD thesis, University of Toronto, 1996. 2, 11, 16

[Tur52] TURING A. M.: The chemical basis of morphogenesis. *Philosophical Transactions of the Royal Society of London. Series B, Biological Sciences 237*, 641 (aug 1952), 37–72. doi:10.1098/rstb.1952.0012. 18

[Tur10] TURK G.: Sticky feet: Evolution in a multi-creature physical simulation. In *Artificial Life XII: Proceedings of the 12th International Conference on the Synthesis and Simulation of Living Systems, ALIFE 2010* (2010), pp. 496–503. 16

[Ven] VENTRELLA J.: Gene pool. Accessed: 2020-06-05. URL: http://swimbots.com/index.html. 14, 15, 16

[Ven05] VENTRELLA J.: GenePool: Exploring the Interaction Between Natural Selection and Sexual Selection. In *Artificial Life Models in Software*. Springer-Verlag, London, 2005, pp. 81–96. doi:10.1007/1-84628-214-4_4. 14, 15, 16

[VFRS17] VEENSTRA F., FAINA A., RISI S., STOY K.: Evolution and morphogenesis of simulated modular robots: A comparison between a direct and generative encoding. In *Applications of Evolutionary Computation* (Cham, 2017), Squillero G., Sim K., (Eds.), Springer International Publishing, pp. 870–885. 2, 7, 8, 10, 15, 18

[VG20] VEENSTRA F., GLETTE K.: How Different Encodings Affect Performance and Diversification when Evolving the Morphology and Control of 2D Virtual Creatures. In *The 2020 Conference on Artificial Life* (2020), no. 1996, MIT Press, pp. 592–601. doi:10.1162/isal_a_00295. 10, 15, 17, 18

[VN66] VON NEUMANN J.: *Theory of Self-reproducing automata*. University of Illinois Press, 1966. URL: https://archive.org/details/theoryofselfrepr00vonn_0/mode/2up. 3, 15

[Wai74] WAINWRIGHT R. T.: Life is Universal! In *Proceedings of the Winter Simulation Conference* (1974), pp. 448–458. 3

[Wal12] WALL M. M.: Galib, 2012. Accessed: 2019-11-20. URL: http://lancet.mit.edu/ga/. 11

[Wan14] WAN C.: Running star, 2014. Accessed: 2020-05-28. URL: http://www.alife.cs.is.nagoya-u.ac.jp/~reiji/ban/. 16

[Wil07] WILLMOTT A.: Fast object distribution. In *ACM SIGGRAPH Sketches* (2007), no. sap 0312, pp. 80–es. doi:10.1145/1278780.1278877. 13

[Wol69] WOLPERT L.: Positional information and the spatial pattern of cellular differentiation. *Journal of Theoretical Biology 25*, 1 (oct 1969), 1–47. doi:10.1016/S0022-5193(69)80016-0. 9

[Wol83] WOLFRAM S.: Statistical mechanics of cellular automata. *Reviews of Modern Physics 55*, 3 (jul 1983), 601–644. doi:10.1103/RevModPhys.55.601. 3, 15

[Yae93] YAEGER L.: Computational Genetics, Physiology, Metabolism, Neural Systems, Learning, Vision and Behavior or PolyWorld: Life in a New Context. *Artificial Life III, Vol. XVII of SFI Studies in the Sciences of Complexity, Santa Fe Institute* (1993), 263–298. 14, 16

[ZSKS18] ZHANG H., STARKE S., KOMURA T., SAITO J.: Mode-adaptive neural networks for quadruped motion control. *ACM Transactions on Graphics 37*, 4 (jul 2018), 1–11. doi:10.1145/3197517.3201366. 2