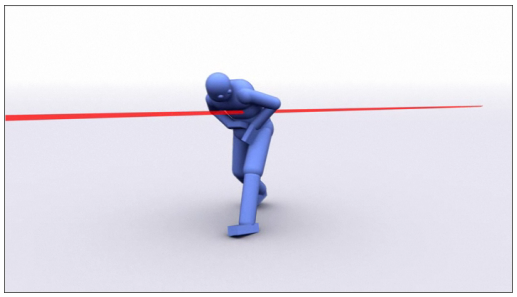
1
Introduction

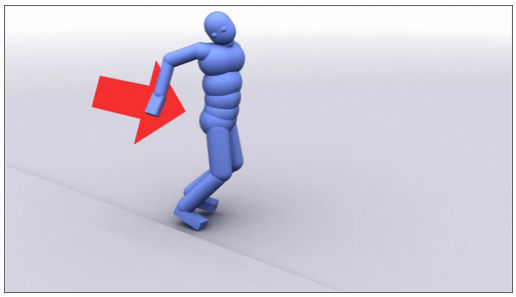
Character animation is a fundamental element in the entertainment industry. Electronic games, in particular, depend heavily on the ability of its artists to create virtual characters that move in a realistic manner, but this is a costly and time-consuming process. Procedurally-generated content has become important in the past years as a means of reducing artists' workload, producing games with smaller footprints, and reducing the overall cost of game development.

Some software developers have already begun tackling this problem. NaturalMotion's *Euphoria* A.I. engine (`www.naturalmotion.com/euphoria.htm`) stands out as a remarkable achievement in this field, being capable of generating animations for humanoid characters on-the-fly in response to environmental stimuli (see Figure 1.1). In systems like Euphoria, the animator instructs the 3D character to perform a sequence of behaviors instead of specifying movements frame-by-frame. Furthermore, animators can set end poses that the character will try to achieve using its own muscle power, which helps maintain continuity between animation sequences. These new animation processes are aligned with the goals of *behavioral animation*, where key-frame animators become directors, as firstly pointed out by Craig Reynolds in his seminal work about flocking and herding [Reynolds 1987]. Behavioral animations stand in contrast to traditional *canned animations*, painstakingly hand-crafted by human designers using animation software such as *3D Studio*, *Softimage*, and *Maya*.

NaturalMotion's software is the result of many years' worth of research and a significant financial investment, resources that the author of the present work and his research team do not possess. Rather than attempting to replicate NaturalMotion's results, our research takes a similar, but as-of-yet unexplored path: We propose a method for animating characters whose bodies are non-humanoid and, in fact, *arbitrarily-shaped*, focusing our goals on *horizontal displacement*. These virtual characters will not be represented by polygons or pixels, but by bones and muscles — the same components that make up real animals. These components can then be translated into a series of constraints



1.1(a): A humanoid character is shot in the chest.



1.1(b): A humanoid character is pushed from behind.

Figure 1.1: Virtual characters animated by the *Euphoria* engine react in a realistic manner to their surroundings and dynamically adapt their poses to these stimuli. (Image credit: `www.naturalmotion.com`)

and force actuators driven by physics simulation, which the character's "brain" will control. This way, the animation can be modeled as a set of instructions (a program) that is executed by the character's body.

Note that this is an inversion of traditional animation techniques, such as *key-frame animation* or *motion capture*. Those techniques rely on explicit positioning of the character's body by a key-frame animator or motion-captured actor, and the poses are then interpolated to generate each individual frame. Our proposed technique takes a static virtual character and generates each frame of its motion with physics simulation, after which it is possible to extract the key-frames.

The core of the problem, and the focus of our solution, is the aforementioned *program* that determines the character's motion. This program represents the character's intelligence, and characters with different body structures will need different programs to animate properly. It is possible, nevertheless, to establish very simple and broad goals to ensure the animations are generated as expected; for instance, a running animation might be described with the goal "maximize horizontal speed".

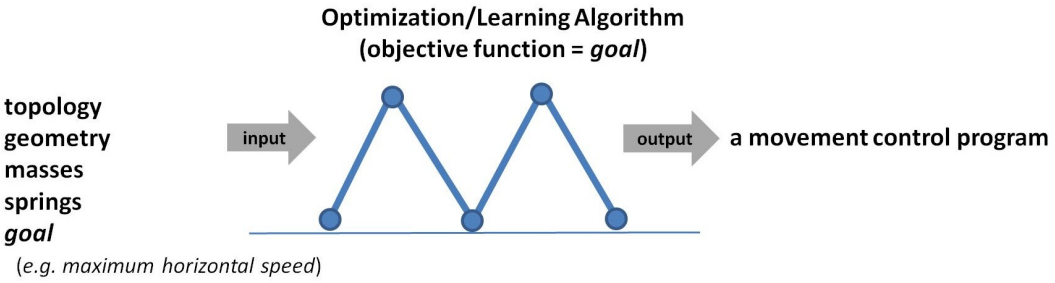


Figure 1.2: The process of motion synthesis for articulated figures.

Figure 1.2 illustrates the above-mentioned problem, where an optimization/learning algorithm (a genetic algorithm, in our case) synthesizes a motion controller for an articulated figure according to an animation goal. This motion controller is a program containing operations and values (e.g. "contract spring #1 by a factor of 0.5"). In the present dissertation we propose two types of programs: command sequences and expression trees.

Once an animation has been generated, it can be "played back" by running the physics simulation while following the instructions in the program. This approach has the following characteristics:

- The animation is simple to represent.
- Displaying the animation is straightforward.
- Creating the animation itself, however, is not.

In short, a solution to our problem (i.e., an animation program) is easy to represent, and we can tell a good solution from a bad one, but we do not know how to actually generate a good solution. These traits suggest the employment of *genetic algorithms*, an optimization technique well-suited for finding approximate solutions based on heuristics.

The main results of the present research work were already published in a paper by the dissertation's author [Luchini et al. 2008]. In relation to that previous work, this dissertation reports the following results: An extension of the animation program generated by the genetic algorithm, and an attempt of expanding the system to 3D articulated figures. The results for the 3D case were not considered robust enough to be fully presented, but we decided to report them because they can be useful for further research.

The present work is organized as follows: Chapter 2 discusses previous works and the current state-of-the-art of physics simulation, genetic algorithms, and procedural animation. Chapter 3 presents a few methods for describing the articulated bodies of virtual characters, while Chapter 4 presents two methods for describing the motion of those bodies. Chapter 5 details how the description of that motion can be converted into actual frames of animation for display. Chapter 6 describes the process that generates the animations in an automated manner. Chapter 7 presents our results, and from those results we extract a few conclusions in Chapter 8.