3
Modeling the Character

Since we are trying to simulate real-life biologic systems, it makes sense that we use the same components that are found in real-life animals to represent the virtual character: bones, muscles, joints, etc.

3.1
Passive Components

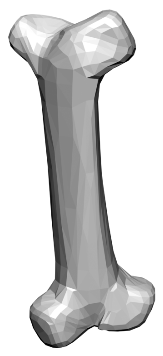
Passive components are those that cannot be controlled by the character's brain. For instance, animals are unable to change their bones' length; the virtual creature should, likewise, not be allowed to change his.

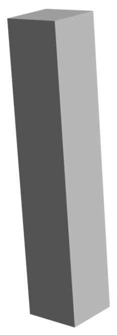
3.1.1
Bones

Bones are, perhaps, the most important component of the character, as they most easily define a recognizable, visible shape for the body. Artists are already used to working with skeletal animation in 3D modeling software, so their existing skills can be easily employed when modeling virtual creatures. Furthermore, when bones collide against the environment (ground, walls, etc.), the physics simulation system applies collision forces and friction to the rigid bodies, which become the main method of propulsion for the character.

Ideally, bones should be represented by *rigid bodies* (either meshes or geometric shapes) that are free to move and spin in space. Simulating the physical behavior of rigid bodies, however, is a computationally expensive process. As we will see in Chapter 7, physics simulation is the bottleneck in our algorithm and we should consider any possible measures to make it more efficient. One way to achieve faster physics simulation is to represent the bone with *particles* connected by a *bilateral constraint* — that is, the two particles will always be at a fixed distance from each other [Jakobsen 2001]. This technique sacrifices accuracy[1] in favor of performance, which may or may not be acceptable depending on the application.

[1]Several physical variables become inaccurate or meaningless; these include velocity, conservation of energy, and friction. The position and orientation of the bone itself can only be inferred from the two particles's coordinates.


3.1(a): A bone represented by a mesh.


3.1(b): A bone represented by a parameterized geometric shape.

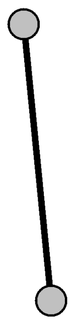
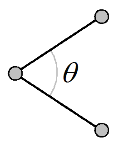

3.1(c): A bone represented by two particles and a bilateral constraint.

Figure 3.1: Possible representations for the character's bones.

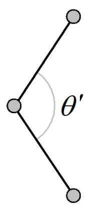
3.1.2 Joints

Joints keep bones connected to each other and constrain their movement.

When applied to rigid bodies, joints involve many complex mathematical concepts and require a significant effort to implement correctly, both of which are beyond the scope of our research. Therefore, we chose to employ a physics engine that provided joint implementations out-of-the-box [Nvidia 2008].



3.2(a): The joint in a valid state.



3.2(b): After a simulation step, the joint might be in an invalid state.

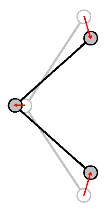

3.2(c): The particles are moved so that their angle becomes valid again.

Figure 3.2: Enforcing joint constraints in a particle-based simulation.

In the more simple case of particles, joints can be thought of as an extension of bilateral constraints between three particles, with one particle being the pivot and two particles being the arms. In a manner similar to the "bone length" in [Jakobsen 2001], the joint constraint repositions the three particles to ensure that the angle formed is always contained in a defined interval (see Figure 3.2).

3.2 Active Components

Active components are those that can be controlled by the brain.

3.2.1 Springs

A *spring* applies linear force when contracted or stretched past its natural length (see Figure 3.3). To achieve locomotion, the brain can send commands to change the spring's natural length (see Figure 3.4).

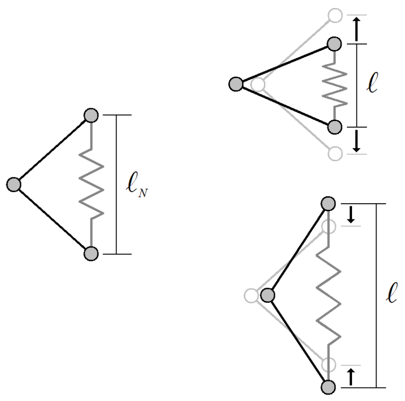


Figure 3.3: A spring exerts a force whenever it is stretched or compressed. As per Hooke's law, the force is proportional to the spring's deformation ($\ell_N - \ell$) and its force constant (k).

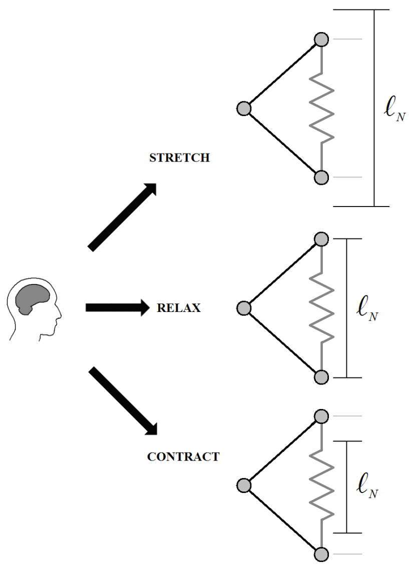


Figure 3.4: The brain sends commands to the spring, telling it to change its natural length. This pulls the bones together (by decreasing ℓ_N) or pushes them apart (by increasing ℓ_N).

Springs are a rather crude approximation of how muscles really work. However, they are useful as a simulation of abduction/adduction movements (such as pulling the knees together), which would otherwise require very complex modeling.

3.2.2
Angular springs

An *angular spring* connects to a joint (the junction of two bones) and applies a torque when it is contracted or stretched past its natural angle (see Figure 3.5). The brain can send commands to change the angular spring's natural angle.

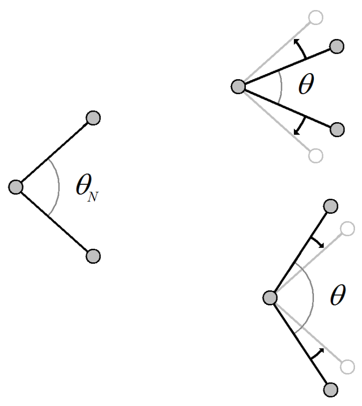


Figure 3.5: An angular spring exerts a torque whenever it is stretched or compressed. As per the angular version of Hooke's law, the torque is proportional to the angular spring's angular deformation ($\theta_N - \theta$) and its force constant (k).

Linear and angular springs have a theoretically infinite number of possible states — any real number can be set as their natural length or angle. Our test application simplified it down to only three states: The *relaxed* state will set the natural length to the value it had at the beginning of the simulation, the *contracted* state will set the natural length to half that value, and the *stretched* state will set it to twice that value. It was a somewhat arbitrary choice; depending on the application, this set-up might take a lot of tweaking to achieve good results.

3.2.3
Motor

A *motor*, like an angular spring, attaches to a joint. Unlike a spring, however, the motor is controlled based on its output torque. The brain can send commands to change the torque applied by the motor onto its two connected bones.

3.2.4 Claws

A useful (if somewhat unrealistic) component to have in a character is a *claw*, that is, an extremity that is capable of gripping walls and floors to anchor the body's movement. When the brain sends a *grip* command to the claw, it will "stick" to whatever surface it is currently touching (and thus become unable to be moved from that spot). When the brain sends the claw a *release* command, it will cease being "sticky." A claw that is in a "sticky" state but is not touching any surface is unaffected.

3.3 Bringing It Together

Figure 3.6 displays some examples of 2D and 3D creatures that were created with the components previously mentioned.

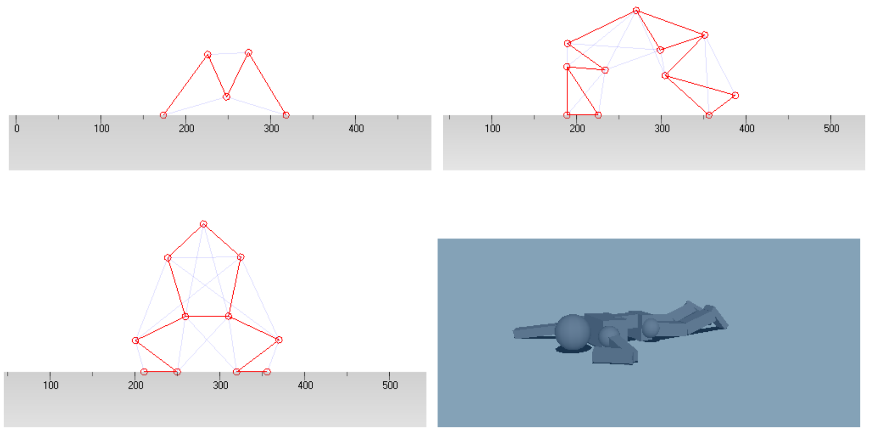


Figure 3.6: Examples of virtual bodies designed with our test program.

When the creatures are inserted into a physics simulation environment, they behave like traditional "rag dolls" — i.e., they fall and crumple on the floor. The next chapter will deal with the animation of these inert bodies so that they actually appear to be alive.