

7 Results

We developed two test applications, each combining a different set of elements from the previous chapters.

Application title	<i>Creepy Crawlies</i>	<i>CC3D</i>
Physics engine	Custom particle-based 2D engine	PhysX
Bones	Bilateral constraints between particles	Three-dimensional rigid bodies
Muscles	Linear springs	Motors
Animation	Op-code sequences	Expression trees
Playback parameters (see Chapter 5)	$FIXED_DT = \frac{1}{500}$, $OPCODE_DT = \frac{1}{128}$	$FIXED_DT = \frac{1}{30}$

7.1 Creepy Crawlies

This application opens with a “creature editor,” where the user can build a virtual creature using particles, bone connectors, and springs (see Figure 7.1). When finished, the user can click the “Generate animation” button to begin the G.A. process that will generate an animation for that creature. The evolution runs indefinitely, but the user may choose to interrupt it at any time and see the best individual that has been discovered so far.

The algorithm takes about a dozen generations to produce an acceptable animation, although these early specimens are very jittery and prone to stumbling. The process usually stabilizes after the 30th generation with realistic, plausible animations for the character. These numbers are affected by the creature’s structure, though; it took 20 generations to animate a simple M-shaped creature, while a more complex quadruped-like being took 60 generations. Nevertheless, an important point to remember is that genetic algorithms are stochastic and unpredictable processes, and some creature designs required 100 or more generations before the evolution stabilized.

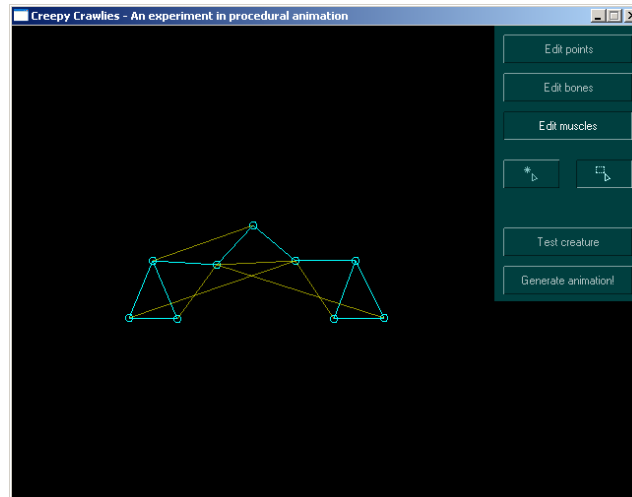


Figure 7.1: The first test application, *Creepy Crawlies*, in “creature editor” mode

Using a profiler [Abedi 2007] to measure the application’s performance, we discovered that the bulk of computational effort is spent running the physics simulation during the evaluation step (on average, 95% of the time was spent in the physics simulation code). Therefore, it was to be expected that complex creatures with many bones and joints would take a longer time to evaluate (and, therefore, a longer time to advance one generation).

We expected “nothing” op-codes to be extremely important during the evolution of an animation program, but this prediction turned out to be wrong. To test it, we designed our application so that, every time a random op-code had to be created (in the initialization of the first generation, in the constructive mutator, and in the replacing mutator), Algorithm 5 would be invoked. The parameter $f_{Nothing}$ could tweak the generation of invalid op-codes; $f_{Nothing} = 0$ would result in no invalid op-codes at all, while $f_{Nothing} = 0.5$ would result in one third of all op-codes to be invalid. To our surprise, the animations generated by the application were very similar in both cases.

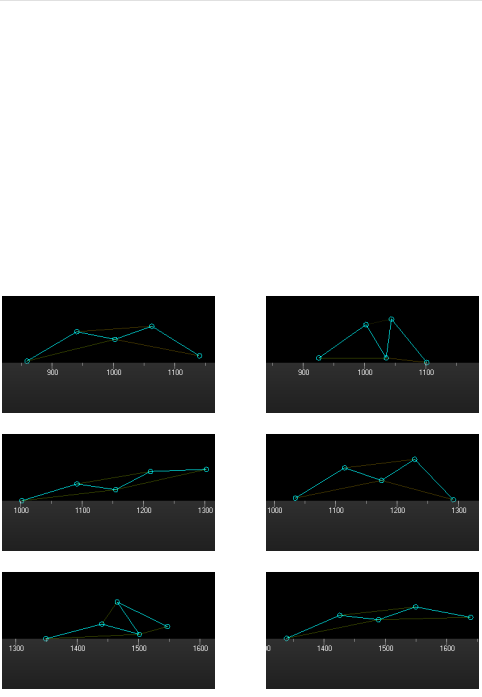
Algorithm 5 Here, N_S is the number of springs, N_C is the number of claws, and **rand** is a function that returns a random number between zero and 1.

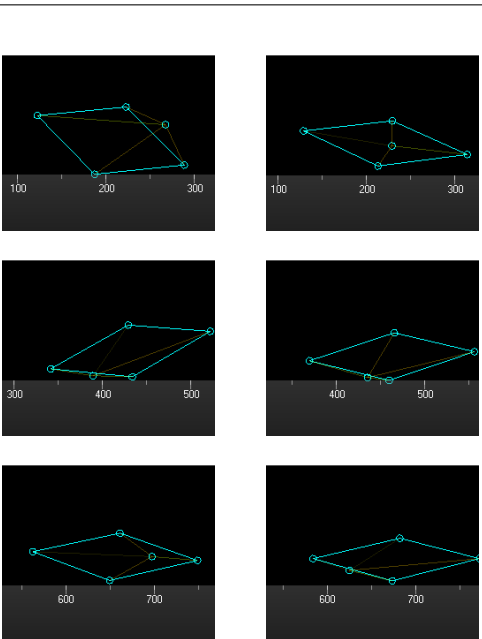
```

function MakeRandomOpCode()
  return  $(1 + f_{Nothing}) \times (3 \times N_S + 2 \times N_C) \times \text{rand}()$ 
end function

```

The next page shows two examples of creatures and their generated animations.

 <p>Still frames of animation from the creature's movement</p>	<p>The animation has 33 op-codes:</p> <ul style="list-style-type: none"> Relax muscle 0 Nothing Relax muscle 2 Stretch muscle 2 Nothing Stretch muscle 2 Nothing Nothing Contract muscle 0 Nothing Nothing Stretch muscle 1 Stretch muscle 0 Nothing Nothing Stretch muscle 1 Nothing Stretch muscle 0 Stretch muscle 1 Nothing Nothing Relax muscle 0 Stretch muscle 0 Nothing Nothing Stretch muscle 2 Nothing Contract muscle 1 Relax muscle 0 Contract muscle 0 Contract muscle 1 Contract muscle 2 Nothing
---	--

 <p>Still frames of animation from the creature's movement</p>	<p>The animation has 23 op-codes:</p> <ul style="list-style-type: none"> Stretch muscle 1 Nothing Nothing Stretch muscle 0 Contract muscle 2 Contract muscle 3 Stretch muscle 0 Nothing Nothing Relax muscle 1 Contract muscle 1 Nothing Relax muscle 3 Relax muscle 3 Nothing Nothing Stretch muscle 3 Relax muscle 0 Contract muscle 2 Nothing Relax muscle 2 Contract muscle 3 Nothing
---	--

7.2

CC3D

Unlike *Creepy Crawlies*, this application reads creature descriptions stored in text files. (We did not develop a 3D creature editor because it is an enormously complex task well beyond the scope of our limited time and resources.) The creature description includes only bones and joints; motors are automatically attached to all joints when the text file is loaded. The user initially sees the creature as a “rag doll” and may press a key to begin the evolution process, which runs for a fixed length of 100 generations.

As expected, the evaluation of individuals was significantly slower than in the previous test (physics simulation of 3D rigid bodies being more computationally expensive). However, even after 100 generations — which took over twenty hours to complete —, the application failed to find an acceptable animation for even the simplest of creature designs. It rarely moved from its initial spot, and instead thrashed around aimlessly.

We suspect the difficulties were caused by the complexity of the search space. Although this application was a failure, we decided to mention it in the text because the underlying concepts may be of value for future research works.