John Dunham

4/7/14

Genetic Algorithm Assignment

Overview:

A few months back I saw this video and was frankly amazed. It was awesome to see how an AI control system could generate what I consider to be impressively life like animations. To top it all off it seems as though they employed a genetic algorithm to optimize the control system. Naturally, when I heard that I needed to implement a genetic algorithm for this class I wanted to replicate this work in some way shape or form. As such the goal of my genetic algorithm is to attempt to produce a two-dimensional skeletal animation.

Once again I am sorry for the length, but there was a lot of data to synthesize and details that were ripe for discussion and investigation.

Environment:

Going in I knew I wanted to work in a JavaScript; inspired inspired by this variation of the genetic car, my desire to do a project in JavaScript this semester. As a result I had to roll out my own skeletal system and apply some kind of physics to it.

For the physics I utilized Box2DJS, a port of the flash version of the popular rigid body physics system, as I had some familiarity with the objective-c implementation and, at the time of selecting it, seemed to be well documented. In determining the physics system I was then able to define the skeletal system that I would be animating as it gave me concrete limits to what I could and couldn't do with my animations.

My skeletons consist of three classes of entities: head, bone, and joint. Each skeleton has one and only one head placed on the leading edge of the first bone defined for the skeleton. The head is connected to the "torso bone" by a Box2DJS construct known as a revolute joint. This joint is then constrained to not rotate to prevent the rotational motion of the head from producing strange results.

The bone entity class defines the square Box2DJS entities I refer to as bones. There is little of note in regards to this entity class, save that individual entities of type bone are disallowed to collide with one another (this prevents strange errors from occurring). Joints in my skeletal system take the form of Box2DJS revolute joints and have the dual purpose of representing joints and muscles in my skeletal system. A joint only comes into existence when a new bone is added to the skeleton, parenting the new bone to a parent bone that is ultimately the child of the "torso bone" via a joint enforced hierarchy system.

With that established I found it necessary to create a skeletal drawing system in which a user can click and drag bones, with the drawing system automatically defining joints, creating the head, and expanding the chromosome.

The Chromosome:

Maximum Rotation | *Minimum Rotation* | *Rotation Speed* | *Active Time* | *Rest Time* | *Stuck Time* | *Active Start*

I decided to take a slight cue from the SIGGRAPH video mentioned in the overview when designing my chromosome and attempted to represent skeletal musculature in the chromosome. As is mentioned in the Environment section musculature in my skeletal design was represented in the joint system, as a result I determined that the best course of action to represent musculature in my chromosome would be to focus on the joint system.

In defining the chromosome I had to first consider what I was able to modify in my physics engine in terms of joints. I found that there were about four things I could change that would have any impact on how a particular joint operated: *upperAngle*, *lowerAngle*, *motorTorque* and motor speed.

Upper angle and lower angle are both somewhat obvious, but they define the range of motion the joint has relative to its starting position. As I was attempting to generate some kind of skeletal animation and skeletons (typically) have a set range of motion, I instantly latched onto setting these values from the chromosome as the range of motion allowed by a joint could very easily make or break a run/walk cycle. After some tweaking and tests I defined the ranges for the two chromosome values as *3* and *-3*, as I found that range allowed for just the right amount of diversity in the motion. The *upperAngle* and *lowerAngle* fields were translated to *Maximum Rotation* and *Minimum Rotation* in my chromosome, respectively.

While still relatively easy to understand, *motorTorque* is a little less straight forward. In Box2DJS *motorTorque* defines the maximum force that may be applied to a joint. While this might be a useful setting to tweak, I found that certain torque settings prevented large bones from moving and smaller bones to go haywire. As I found testing for bones moving out of joint bounds (going haywire) to be somewhat costly and non-trivial, I set this to a default limit and adjusted the bones so they would typically behave normally.

The final modifiable joint setting I was able to find was setting the joint's motor speed. It was cheap and easy to simply modify the joint speed in the update loop to acquire a motion for the joint and it was relatively simple to fine tune for my purposes. Due to the ease of use and sheer utility of this parameter it was included in my Chromosome as *Rotation Speed* (at a range of 0 to 6 radians per second). It should be noted that this speed is a constant, however I do allow it to change directions when at the joint limit or when over the *Stuck Time* as I mention below.

Surprisingly those three chromosome components are capable of doing quite a bit of damage on their own, however, the motions produced at this phase were not conducive to a walk cycle. I knew I would need to apply some changes to how the skeleton applied speed to its motors if I wanted to see some more interesting result. *Active Time* and *Rest Time* were borne of this desire to see more diversity in how the joints applied their speed. In the chromosome I gave each of them a value 0 to 30 that represented time steps in Box2D world (in real world time this is ~0 to ~.5 seconds) for how long the joint would be active and at rest respectively. This simple change allowed for near continuous motion or rest for joint systems, but still allowed for oscillations to emerge which definitely improved the overall fitness of the skeletons when I ran tests.

As I was running tests on the chromosome I realized that the joints would occasionally lock up, getting caught on something or another, and my model of flipping the speed when the joint hit a limit just wouldn't fly. As a result I introduced a portion to my chromosome which I referred to as *Stuck Time*, which would add a counter to my motor that would increment whenever a joint was deemed "stuck" and reverse the motion in an effort to become unstuck. A joint would be "stuck" if the number of update cycles that passed while the joint was both active and motionless exceeded the encoded *Stuck Time* (0 to 30 cycles). While this has proven to have a low incidence I still included it in an effort to work around local maxima for potential skeletal configurations.

The final element of the chromosome was stupendously straightforward, it merely dictated whether or not a joint was moving at the start of the simulation. Naturally I called this *Active Start* and I mainly included it to prevent viable options from getting completely ignored because it was flailing when it dropped into the scene.

Finally, with the joint chromosome defined I could construct a chromosome on a per skeleton basis. The chromosome itself was represented with a JavaScript array, which allowed me to aggregate the differing data types of the chromosome with no real adverse consequences. The chromosome would then be loaded into each joint definition on a per individual basis and the whole skeleton reset with this information.

Population and Crossover:

Walking is hard. Minor variations in even one joint can completely topple a skeleton that had an incredibly stable motion. That being said, it's also really easy to get caught in a local maxima when walking because of this very fact. As a result I knew that the population control would need to have some affordances to encourage stability, but random variation was paramount to finding good solutions.

In the tests I performed the population had a single elite, carrying the most outstanding member of the entire program run until that generation. I opted for a single elite as it ensured that an at least somewhat valid joint configuration lived on and allowed for the rest of the population to try crazy strategies without harming the general progress of the skeleton reaching its goal. The main drawback I found to this approach is that I really never hit a convergent point on any of the skeletons I tested, so perhaps expanding the elites a bit may have helped to reach this point faster as more "good genes" would be selectable for crossover (although I do have some fear of early convergence if there are too many elites).

I selected crossover through a roulette system. Every member of the population was given a normalized fitness value based on the total fitness of the population as a whole. This normalized fitness then determined the rate of selection for a gene: the elite having the highest chance to reproduce and the worst rarely reproducing. The results of this selection were then passed through a two point crossover function. I predominantly selected two point over single point crossover because I felt that it would encourage unique strategies for the animations more. As I attempted the system with both I can say at least anecdotally that this was the observed case. For mutation I iterated over the chromosome, changing it on any fields the mutation rate exceeded the random value (pretty standard mutation).

My methodology seems to work relatively well and at least for my interpretation of the problem it seemed to preform relatively well in pursuit of elite (4/4 skeletons running with this crossover reached a goal state as outlined below). That being said in retrospect I feel as though my crossover may have favored chromosome variation a bit too much. The fragile nature of solutions for this approach and the likelihood of joints that only work well in tandem becoming separated by the crossover, causing children to frequently become "weak" as will be outlined in more depth below. If I were to modify the crossover and population control in the future I would likely increase the number of elites in an effort to see more/better convergence (although my elite selection prevented a solution from not being found).

Fitness:

My fitness function may very well be the simplest aspect of my genetic animator. I measured fitness on four vectors: *maximum distance travelled*, *stopping distance*, *head contact*, and *time to complete*.

The main stat of the fitness function was the *maximum distance travelled*. In making this the main stat I made the point of the genetic solver to reach the green goal line, as the closer the head was to the goal the higher the fitness for that chromosome variation. An unintended consequence of this is that it encouraged explosive strategies that were not quite running/walking animations, but pretty cool all the same. This stat would be calculated as the distance of the final maximum position of the head from the head's starting x value. This distance would then be multiplied by a factor that would make a head that was at the goal have a score of 100,000.

Of course while *maximum distance travelled* encouraged forward motion the fitness score had to account for heads that didn't end at their maximum location. As such I subtracted a score based on the deviation of the head from its maximum distance multiplied by a factor of .25. This *stopping distance* value penalized strategies where the head backtracked, but not so much that a skeleton which travelled a substantial distance was penalized overly.

To speed up the generation processing and hopefully encourage some walk cycles I introduced the notion that a skeleton's head was to be protected if the skeleton were to be considered a success. In adding a substantial penalty (-5,000) in the form of *head contact*, I began to see behaviors emerge in which the skeletons began protecting their head more. Although, the skeletons didn't always protect their heads in the way that I wanted them to. They were more likely to loop an arm around their head than to try and keep the head away from the ground.

To ensure that there was a benefit to a skeletal animation that moved faster I added a *time to complete* factor to the fitness. It was miniscule (-.01), but it was never intended to impact reaching a solution, but to improve solutions that were already found. On multiple occasions I found that this assumption held true.

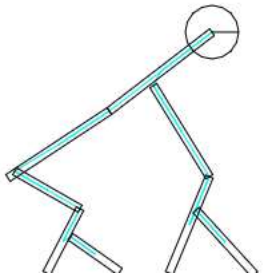
My fitness function is far from complete, if I had more time I would like to track head height and attempt to encourage skeletons that keep their head away from the ground. Additionally, I would like to add some fitness that discourages haywire joint motion, but that's for another time.

Examples:

To see the outstanding members from the described skeletons please open the index.html file in your browser, copy the contents of the Skele#_JSON.txt file and place them in the text box next to the import/export buttons. When the text is in the box click import and the play outstanding.

For all of the following examples the population size per generation was set to 20 and the mutation rate was .35.

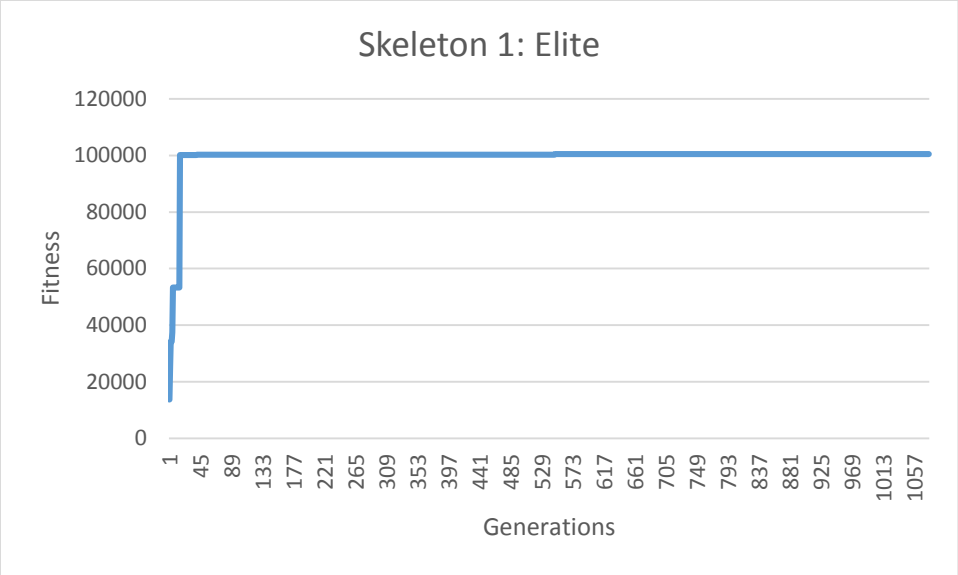
Skeleton 1: "Mud Fish"



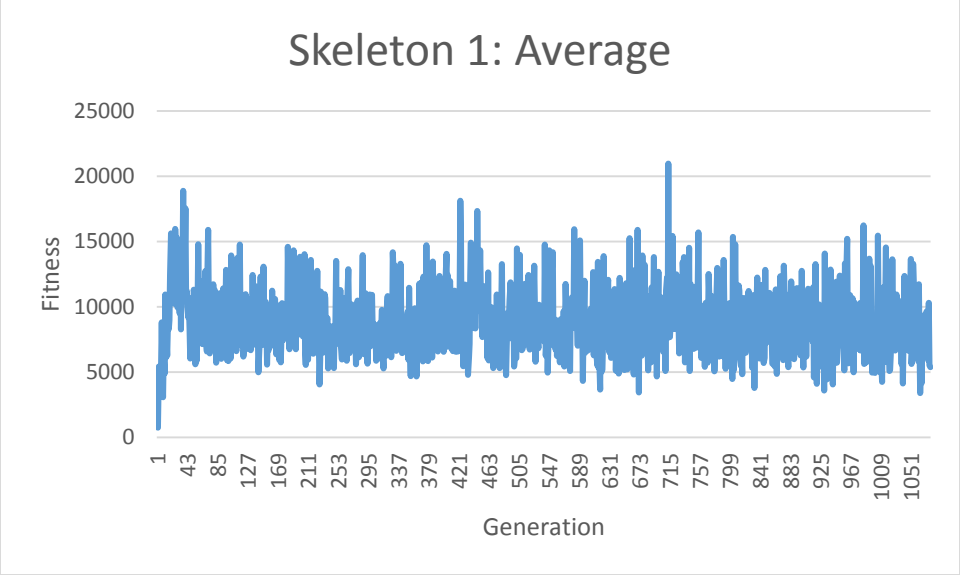
The "Mud Fish" is easily my favorite skeleton of the batch I've prepared for discussion. I didn't name it for its appearance, rather for the walk cycle its most outstanding individual produced. The first few generations aren't terribly interesting, there's a lot of flailing and happy accidents, which is totally within reason for the earlier generations as they are chiefly exploratory. However, around generation 15 things start to get interesting.

At this point the creature seems to be playing it safe, the motion is slower and focuses less on the explosiveness of its predecessors, perhaps in an effort to protect its head (although this is pure conjecture as I don't have a record of the genomes at this generation). As a result it actually makes it to the finish line, the first of any of my skeletons to do so. For this safe strategy heavily influenced the elite strategies in future generations, which is exactly how I had wanted the evolutionary process to work.

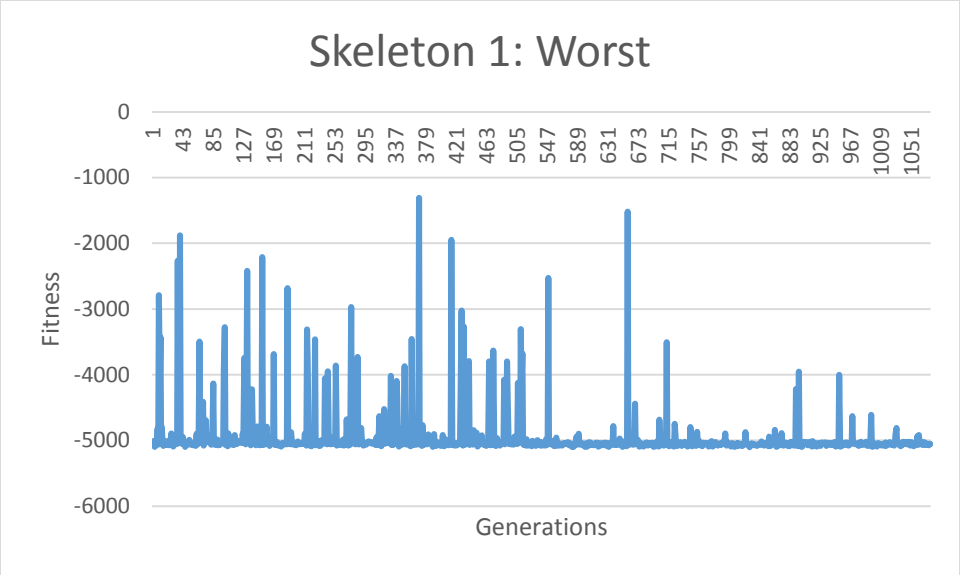
As time went on, however, the skeleton adopted a faster more aggressive strategy, curtosey of the *time to complete* fitness parameter. Once again this was exactly the outcome I had hoped for! That being said the strategy was fairly unstable for a walk cycle, taking a header just before it crosses the goal. After letting it run for a couple hundred generations (547) it finally found a stable variation of the previously successful approach which is responsible for my slightly ill-fitting moniker.



There is little to note here save that the somewhat high mutation rate and large generation size were capable of escaping one maxima to find another after several hundred generations. I would like it if it escaped it sooner, but I feel that to do that would take a substantial level of tweaking and for a general solution this is more than acceptable.

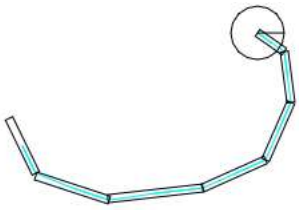


The average data is barely better than noise, but there is a somewhat interesting trend in that for the first 50 or so generations the average is increasing, which is the period of greatest upset for the elites. That being said this is the only transition at which this ocurrs so little may be said in regards to this as a rule. Herein average data will not be included in the report.



Finally, the worst data is literally worthless. As mentioned in fitness it is really easy for a skeletal animation to fail and when a skeleton fails instantly it's around -5,000 with my scoring system. As a result the worst for a generation is generally pretty static. Herein worst data will not be included in the report.

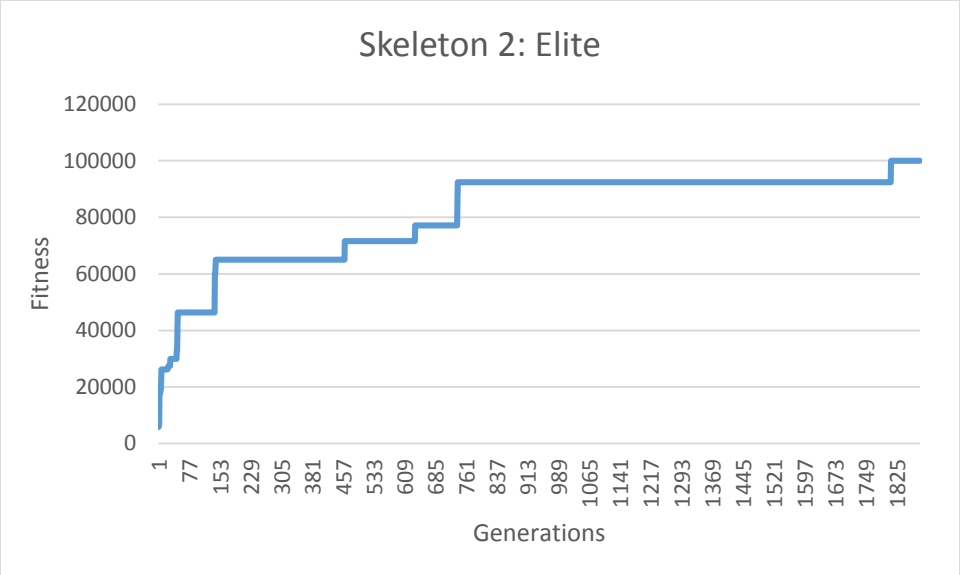
Skeleton 2: "Snake"



The "Snake" skeleton was an exercise to see how the Genetic Algorithm would handle a skeleton with a large amount of joints. From the outset it preferred strategies in which it curled around its own head, as the *head contact* fitness parameter destroyed any hopes of individuals that hit their heads from surviving.

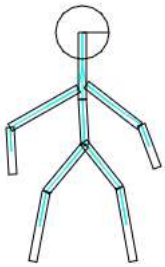
The drive to protect its own head really had a serious impact on the perferred strategy. The snake became more of a tumble weed somewhere near generation 20 as a direct result of that. In the case of the snake this was exactly how I wanted the fitness to manifest, as I wanted to see a creature somewhat similar to a rolling snake. It was actually pretty interesting to see how it evolved this head protection strategy into an explosive coil uncoil pattern.

Once again my fitness worked in the manner I had hoped, although it was apparent that some of the quirks of Box2D may have prematurely killed viable strategies, because the head went through bones and collided with the ground on a number of occassions.



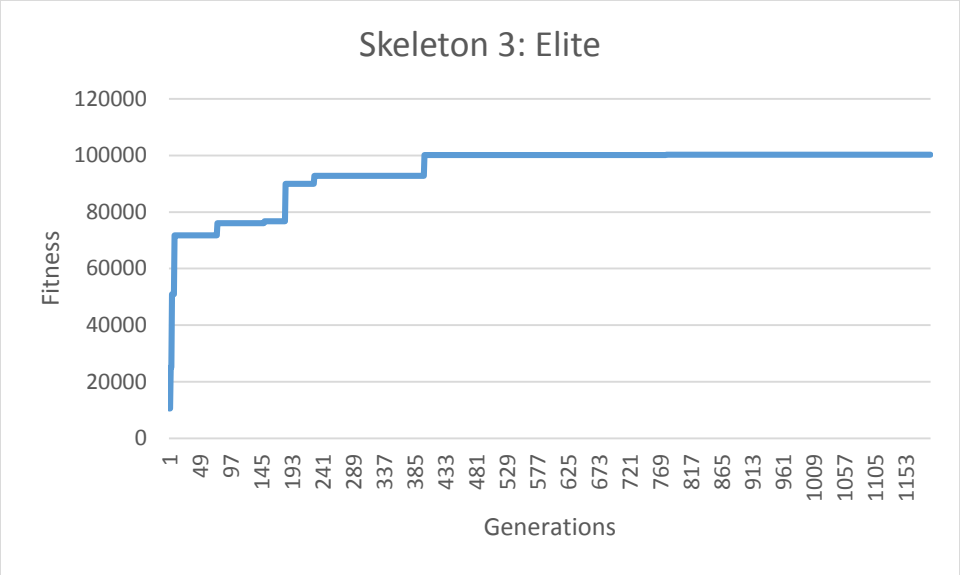
The pattern resembles Skeleton 1, but it's evolutionary events are less explosive. I suspect that the explosive periods are largely luck of the draw. It should be noted, that the average of this skeleton was generally increasing.

Skeleton 3: "Human"



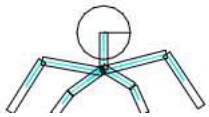
The "Human" is just that, a human rendered in stick figure form. I ran this chiefly because I assumed that this would be one of the first shapes someone may try if I were to make this into a toy of some kind. Initially, the human favored an aerial strategy akin to cartwheels. As the generations marched on the human began to use its abdomen more and more, as typically it could get in the way of the head from hitting the ground.

There was actually very little variation in the strategy after it dicovered the "cartwheels" as they were so effective and achieved all the strcit goals of the fitness function. The core value of this skeleton is that it revealed to me that my fitness has a distinctive weakness: it doesn't favor motion akin to walking. There is no incentive for the system to try and keep it's feet on the ground, as a result a cartwheeling chromosome can have the same value as one that runs like a human.



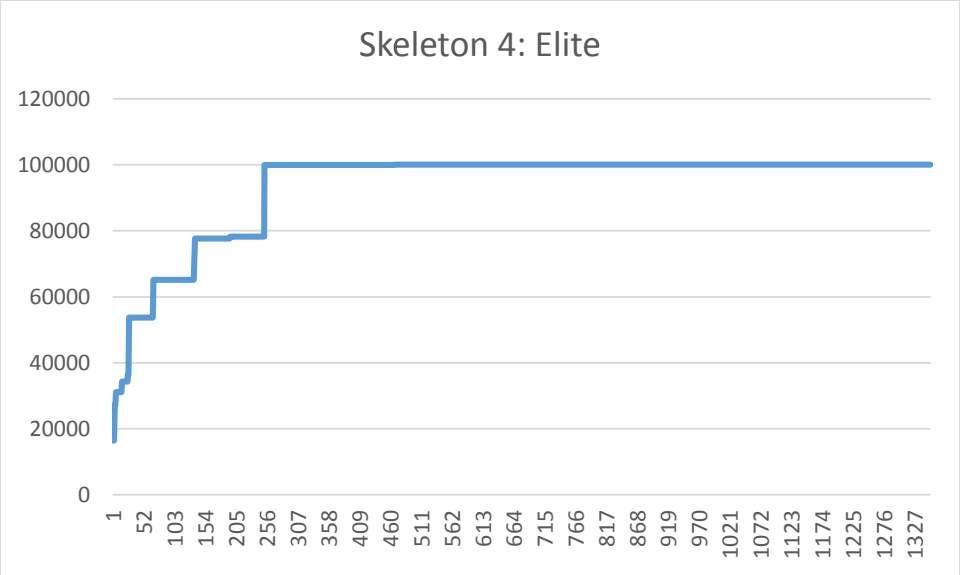
Once again the Skeleton elites increase in a manner similar to Skeleton 1, I suspect that this pattern is typical for my implementation.

Skeleton 4: "Spider"



The "Spider" is efectively a human without a lower torso. There is very little to say about the early motion of the spider as it follows a pattern very similar to that of the "Human". That being said the "Spider" does develop an interesting leaping pattern around generation 67, likely borne of the impressive horizontal speed the motion produced coupled with the fact that it reduced the chance of the "Spider's" head from contacting the ground.

It must be noted that the "Spider" is a buggy skeleton and it indicates a common struggle I had throughout the project. Ocassionally, the "Spider" will get caught in its own geometry and the bones will bug out. This actually causes certain chromosome configurations to not properly execute and as a result it periodically becomes non deterministic. This bad in the context of Genetic Algorithms, and I never found a good solution for it in the context of Box2DJS.



Once again it has a somewhat logarithmic trend line, although the bugs make the data somewhat compromised.

Conclusions/Difficulties:

I've said it before and I'll say it again, Box2DJS is quirky. I found myself fighting against some of its more virulent undocumented quirks and settings for a pretty substantial portion of this project. The most noteworthy of quirks being mentioned in the analysis of Skeleton 4 above. The core problem with this quirkiness is that it really limited the amount of work I could do improving the fitness function and chromosome, although that being said I like to think that I made some decent progress with the advancement of this implementation.

If I were to continue this project in the future there are several things I may change in an effort to improve the results created by this algorithm. First, I would add some fitness reward to keeping the head in a relatively stationary path. It is my belief that the addition of some kind of y-axis tracking of the head would have a distinct impact upon the quality of the walk cycles produced, as it would encourage more stable movement patterns.

Second, I would add some kind of penalty to crossed limbs that counteracts the acceptability of "haywire" strategies. I would have had it in this version, but I ran out of time to find out how to best detect this. Additionally, I would improve the chromosome a bit, most likely adding some form of behavior reactive to forces on the joint.

In all honesty there's about a million tweaks other I could have made, but most of them are tuning: adjusting mutation rates, generation sizes, bone mass, elite sizes, etc. Overall I feel that the current iteration performs within acceptable bounds for non-humanoid skeletal structures with low joint densities. Regardless of its ability to create realistic walk animations I certainly feel I found a space for genetic algorithms in the space of digital toys.