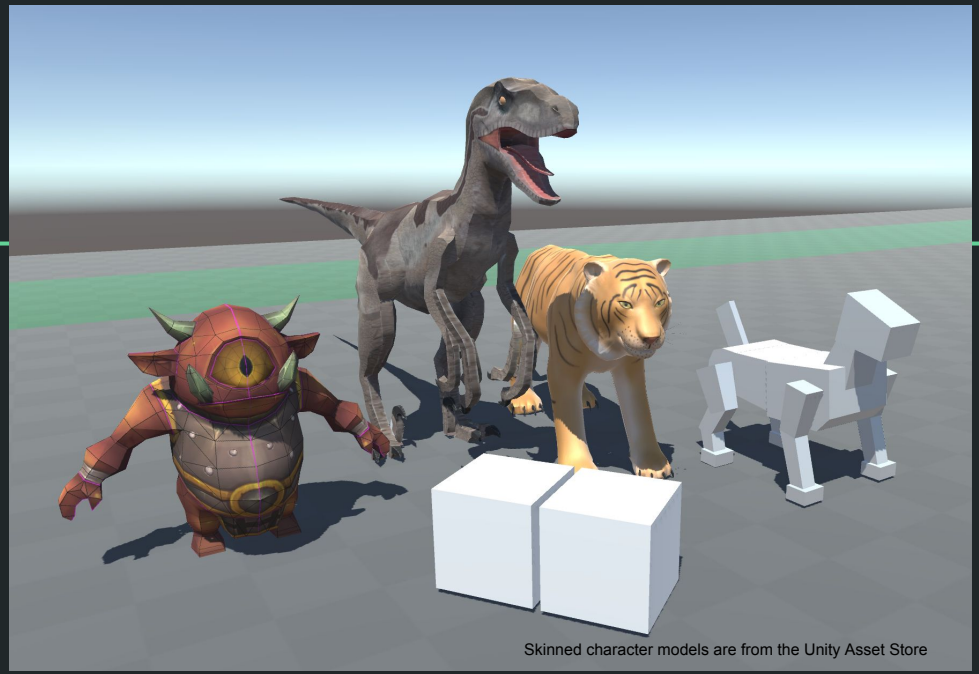
Physical Character Animation

Using Machine Learning

Rob Quinn
Joe Sogard
Joe Kuczek
Luke Oetken
Andrew McKeighan
Kenneth Black

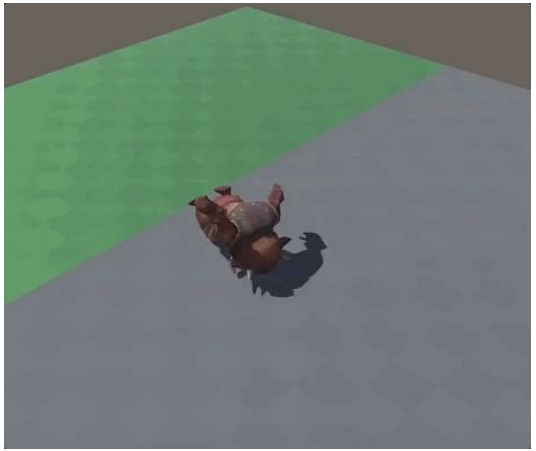
Client/Advisor:
Jim Lathrop

Team 04 May 2018



Introduction

Problem Statement



- Animating characters in video games and media is
  - Time consuming
  - Expensive
  - Hard to make physically accurate
- Our application uses machine learning to create physical animations for characters.

Character Physics Spectrum

100% Physics

Blended / IK / Active

Procedural

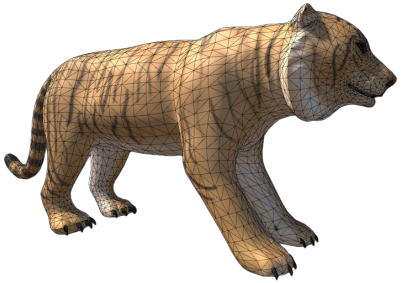
http://store.steampowered.com/tags/en/Physics/

Existing Animation Tools (Competition)

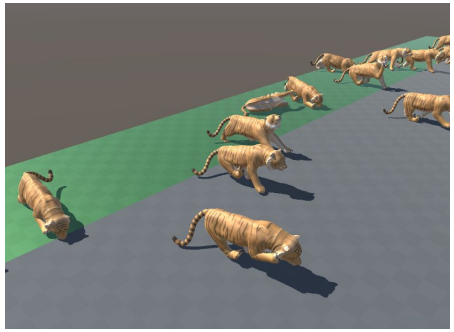
Tool/Process	Professional Animator (Hand-Keyed)	Adobe Mixamo	RootMotion PuppetMaster	**Physical Character Animation (us)**
Cost	$55,000-$80,000 per year	Free	$90 + existing keyed animations	**Licensed / in house (TBD by client)**
Character Limitations	None	Human (biped) only	None	**None**
Physically Based	With extra effort	Yes (motion capture)	No	**Yes**
Physics Driven	No	No	Yes	**Yes**
Automated Animation Discovery	No	No	N/A	**Yes**

https://www.gameindustrycareerguide.com/video-game-artist-salary/
https://www.mixamo.com/#/
https://assetstore.unity.com/packages/tools/physics/puppetmaster-48977

Concept Sketch



3d Game model



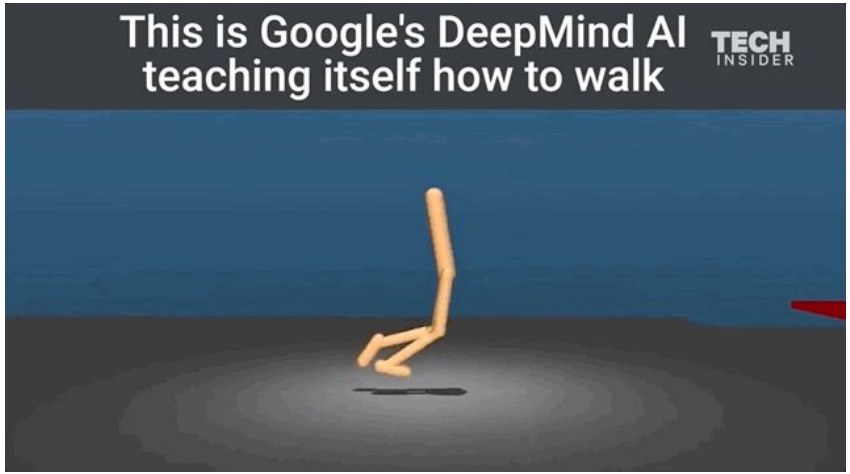
Physics simulation &
Machine learning

Learn physical animations
to use in game

Similar Use of Machine Learning

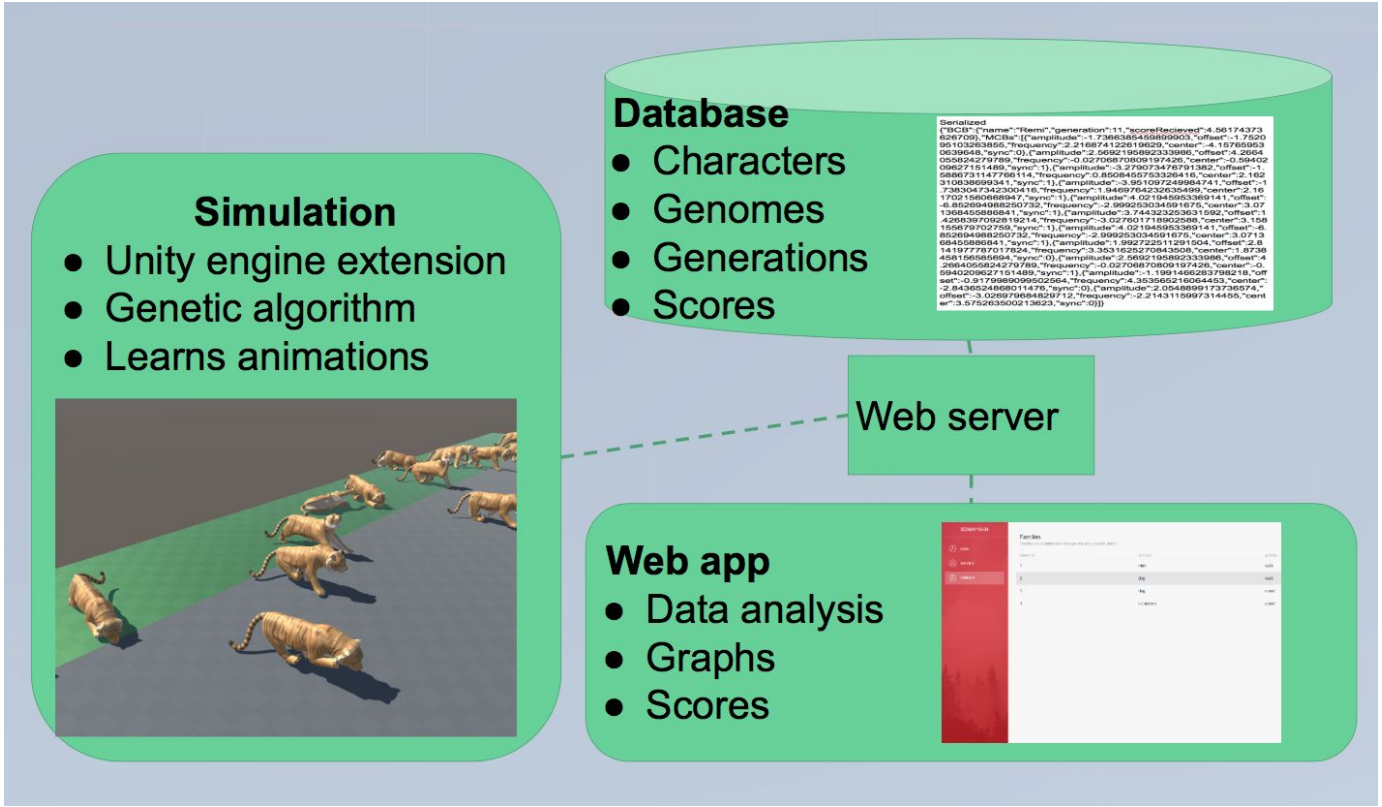
Our selling points

- Used to learn animations for characters
- Designed specifically for games
  - Optimized results for realtime games
  - Integrated in the most popular game engine, Unity



Tech Insider youtube.com

Modules


Simulation
Unity engine extension
Genetic algorithm
Learns animations
Database
Characters
Genomes
Generations
Scores
Web server
Web app
Data analysis
Graphs
Scores

Requirements

Key Non-Functional Requirements

Simulation

- Runs in Unity game engine
- Consistent with physics at various time scales
- Runs without need for supervision from 0 to 1000 generations
- Stable genomes can be reused as a new generation branch

Web application

- Have up to date graph data and genomes
- Be visually pleasing to easily analyze data

Key Functional Requirements

Simulation

- Simulate at least 3 species of characters
- Use genetic algorithms to learn animated movements
- Accurately simulate physics at up to 10x faster than real-time

Web application

- Hold at least 500 genomes (generations) in the database
- Show a graph of fitness scores over time

Considerations - Platform

- Virtual operating environment
- Tool for Unity (game engine)
  - Popular game engine for independent and AAA game developers
  - Cross platform (Windows and Mac development)
  - Includes NVIDIA PhysX
    - Maximum compatibility with Unity
    - Already in nearly all Unity games
    - Our results work instantly in a game
    - No conversions or calibration after learning is complete
- Also considered: OpenGL, Bullet Physics, Monogame, Unreal
  - Too much work unrelated to ML or goals (rendering, importing real game models)
  - Unity has best audience and compatibility

Considerations - Machine Learning Algorithm

- Genetic Algorithm
  - Relatively easy to implement from scratch
  - Easily extensible for our needs
  - Not reliant on external libraries or data models
  - Practical to integrate directly into Unity engine during game development
  - Proven capabilities for locomotion
- Also considered: Unity Machine Learning Agents
  - Also runs in Unity engine
  - Machine learning framework using TensorFlow, Nvidia Cuda, cuDNN
  - Trade off: Could have provided much faster learning
  - Downside: Released after we started working, still in beta today and frequently changes

Team Resources

Cost Estimation - Software and Tools

We planned ahead and chose software that free, open source, or available to us through Iowa State.

- Unity - free
- Visual Studio Code - free
- Coding frameworks/libraries/languages - free
- Web server from Computer Science department - free

Cost Estimation - Research and Development

Man-Hours Budget

Total 422 man-hours by final status report

Planned well and within budget

	Fall 2017 Budget	Fall 2017 Cost	Spring 2018 Budget	Spring 2018 Cost
Research and Testing	100	113	70	95
Simulation Development	60	44	110	92
Web Development	40	26	70	52
Total man-hours	200	183	250	239

Risks and Mitigation

- Scope
  - Must make sure to limit project scope to what is feasible in two semesters.
  - Refined goals part way through to focus on games
- Unity Engine Issues
  - New releases and updates for Unity could significantly disturb our simulation environment
  - Spectre & Meltdown forced us to update (still works)
  - All updates thoroughly tested before updating the project
  - Unity has bugs including physics bugs we had to avoid
- Simulation/Web Communication
  - Inputs don’t get sanitized but are from one source and one format so we know what to expect

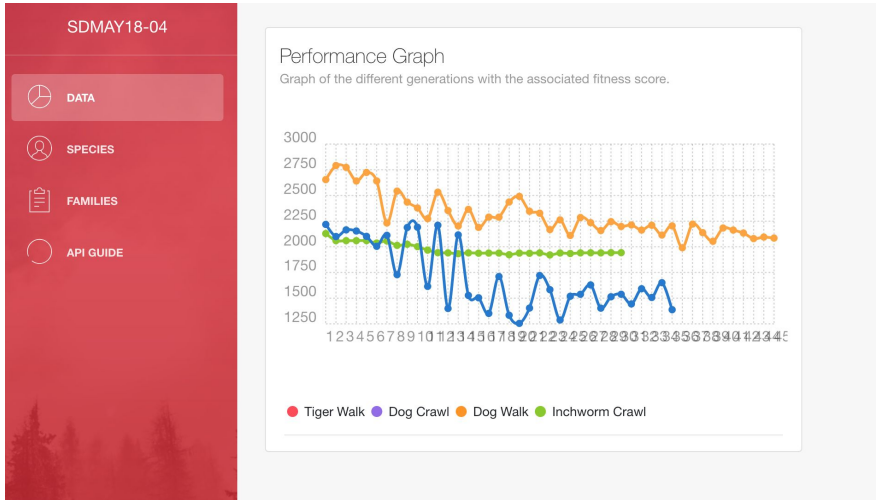
Application Details

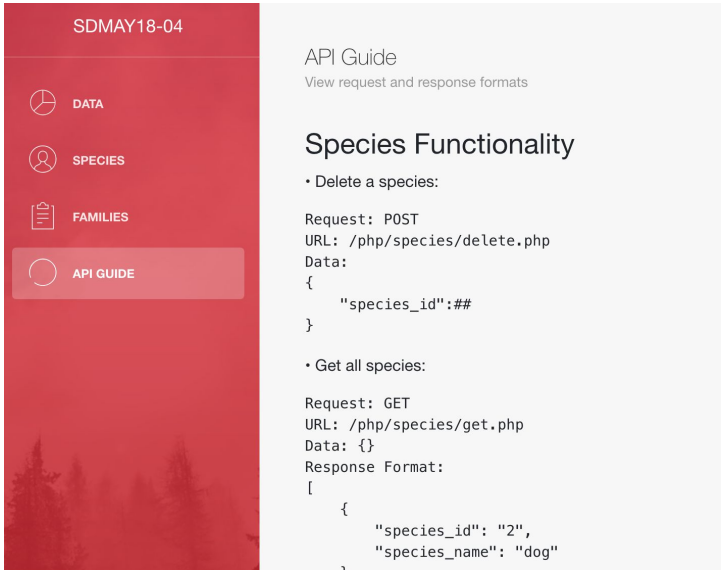
Technology Platform

- Simulation
  - Unity (Engine, Rendering)
  - C# (Code and Genetic Algorithm)
  - NVIDIA PhysX (Physics engine)
- Web Analysis
  - PHP backend
  - Apache web-server
  - MariaDB (MySQL database)
- Web Front End
  - jQuery
  - Vue.js
  - AJAX with RESTful requests
  - BootStrap with HTML/CSS

Web Overview

- Display graphs of fitness data for each character
- Compare different characters and view simulation trials
- Collect data from simulation and display neatly





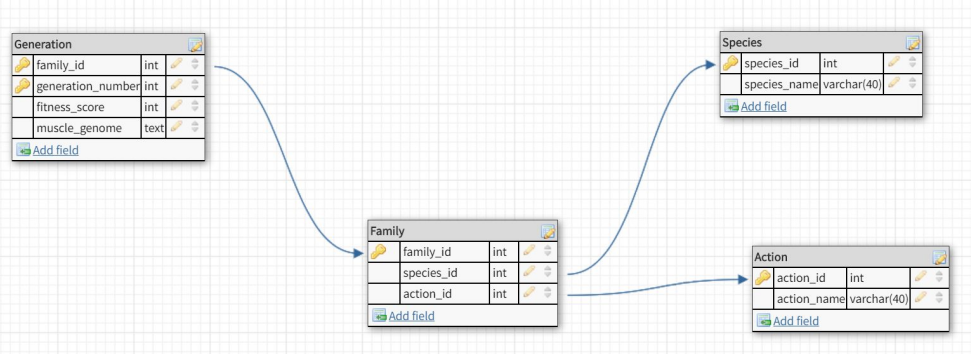
Web Details

Database Schema:

- Generation
- Family
- Action
- Species

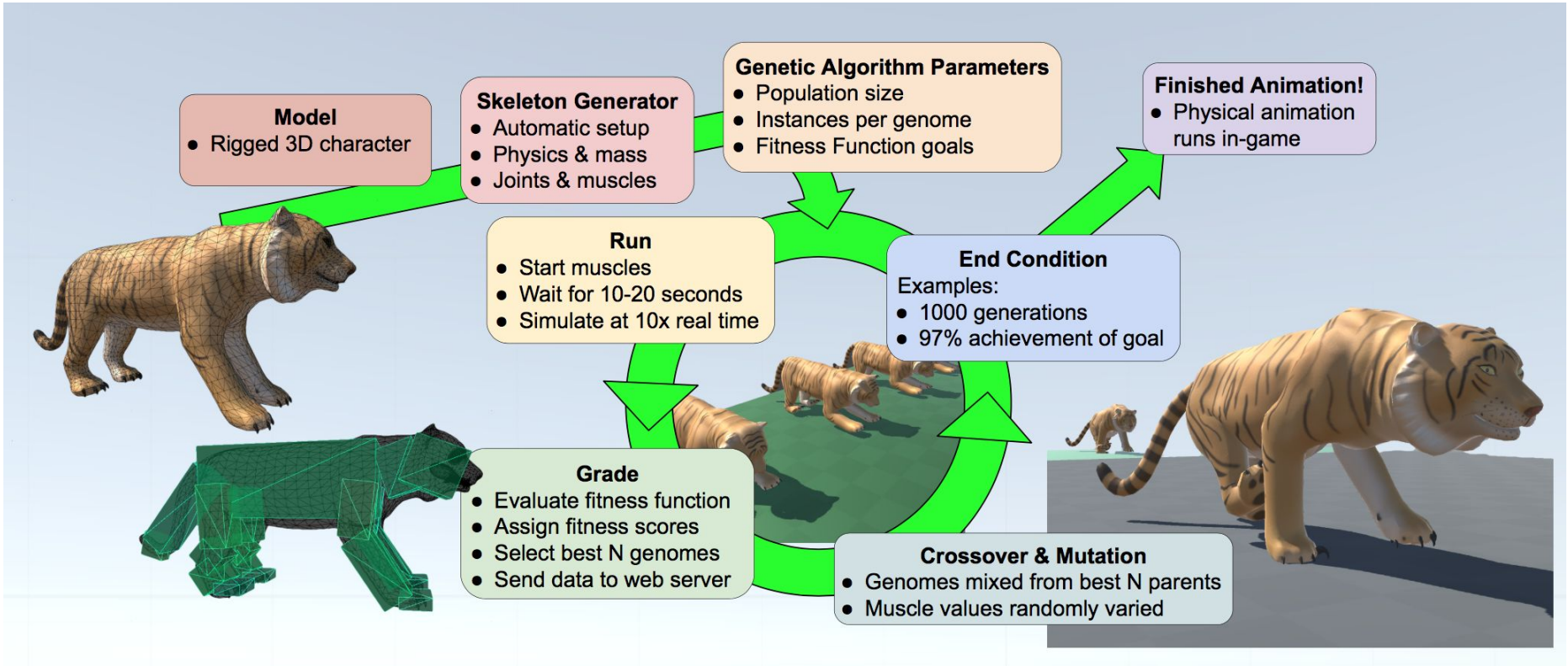
Web API:

- Accessible via C# application
- CRUD
- Error logging
- Developed in PHP



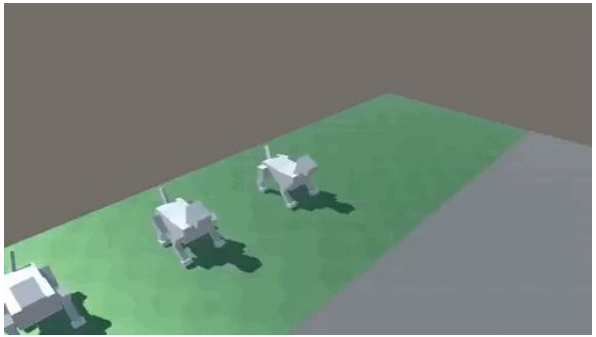
Algorithm Details

Functional Decomposition & Machine Learning

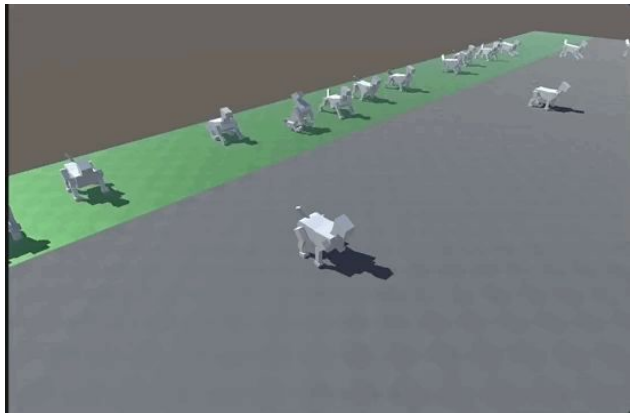


Simulation

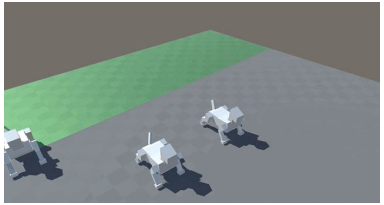
Remi @ Generation 0



Remi @ Generation 30



Compare to 160 gens before enhancements (December 2017)



Simulation - Genome

- Character
- Generation
- Fitness Score
- Muscles
  - Sinewave variables (4)
    - Amplitude
    - Frequency
    - Offset
    - Center

Serialized

{"BCB":{"name":"Remi","generation":11,"scoreRecieved":4.561743736267709},"MCBs":[{"amplitude":-1.7366385459899903,"offset":-1.752095103263855,"frequency":2.216874122619629,"center":-4.157659530639648,"sync":0},{"amplitude":2.5692195892333986,"offset":4.2664055824279789,"frequency":-0.027068708091974 26,"center":-0.5940209627151489,"sync":1},{"amplitude":-3.279073476791382,"offset":-1.5886731147766114,"frequency":0.8508455753326416,"center":2.162310838699341,"sync":1},{"amplitude":-3.951097249984741,"offset":-1.738304734230041 6,"frequency":1.9469764232635499,"center":2.1617021560668947,"sync":1},{"amplitude":4.021945953369141,"offset":-6.852694988250732,"frequency":-2.999253034591675,"center":3.071368455886841,"sync":1},{"amplitude":3.744323253631592,"offset":1.4268397092819214,"frequency":-3.027601718902588,"center":3.158155679702759,"sync":1},{"amplitude":4.021945953369141,"offset":-6.852694988250732,"frequency":-2.999253034591675,"center":3.071368455886841,"sync":1},{"amplitude":1.992722511291504,"offset":2.8141977787017824,"frequency":3.3531625270843508,"center":1.8738458156585694,"sync":0},{"amplitude":2.5692195892333986,"offset":4.2664055824279789,"frequency":-0.02706870809197426,"center":-0.5940209627151489,"sync":1},{"amplitude":-1.1991466283798218,"offset":-0.9179989099502564,"frequency":4.353565216064453,"center":-2.8436524868011476,"sync":0},{"amplitude":2.0548899173736574,"offset":-3.026979684829712,"frequency":-2.2143115997314455,"center":3.575263500213623,"sync":0}]}

Sample Genome

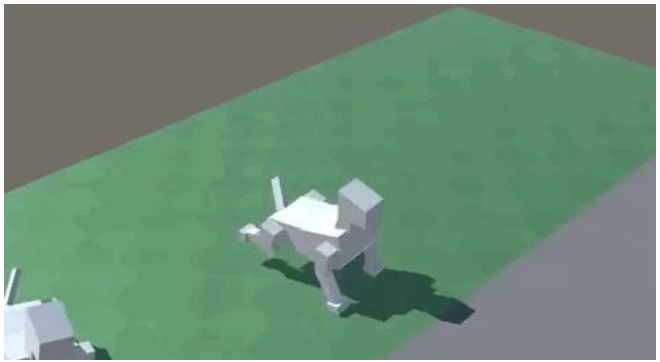
Undesirable Behavior

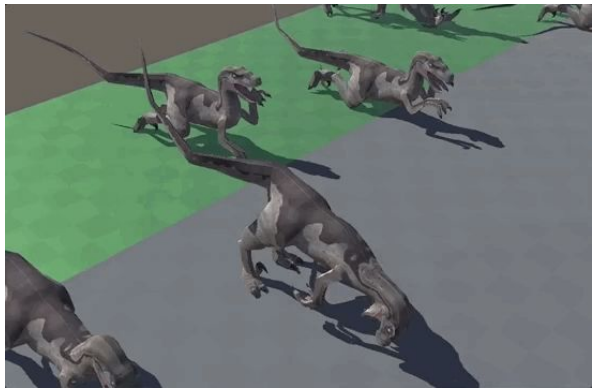
Behaviors to filter out

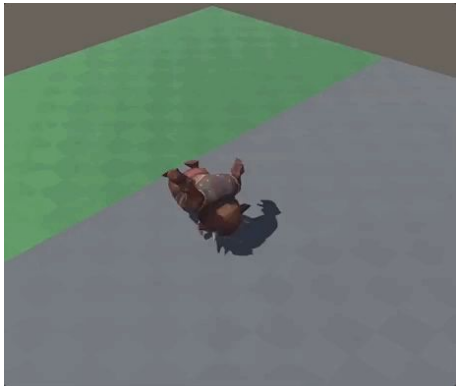
- Severely asymmetrical movements
- Falling down
- Stop, drop, and roll
- Glitching or exploiting physics engine
- Cartwheels / somersaults

Solved by

- Symmetric muscle groups
- Checking character’s head/posture
- Checking height off ground
- Duplicating and averaging with the same genome at varying positions

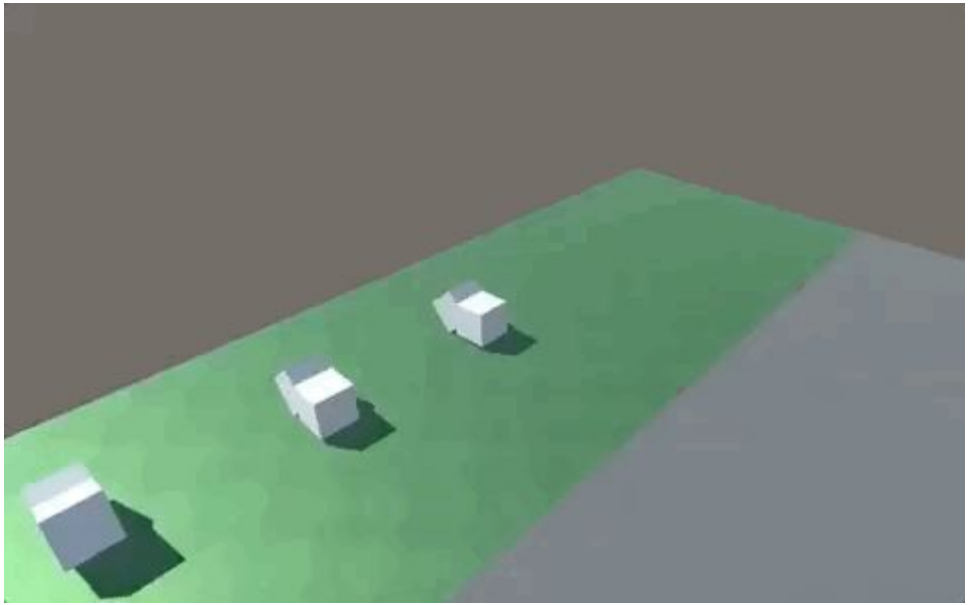






Test Plan

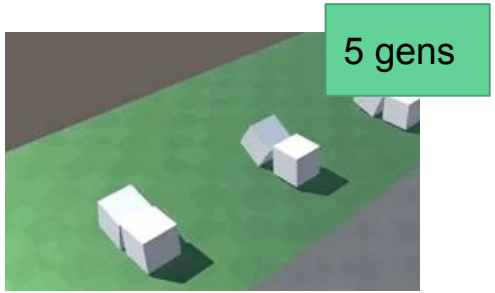
- Simulation
  - Run a scenario
  - Output to website and .csv
  - Plateau/Asymptote
- Rapid testing
  - Inchworm
  - One muscle
  - Finds a solution in ~5 seconds
- Website analysis
  - Unit tests
  - Manual tests

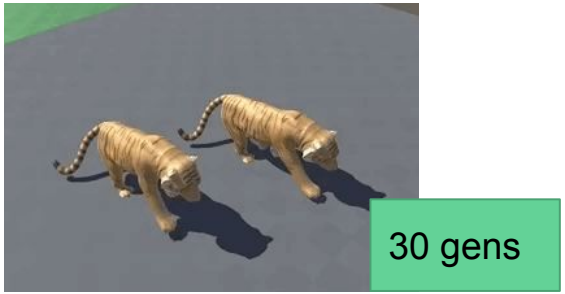


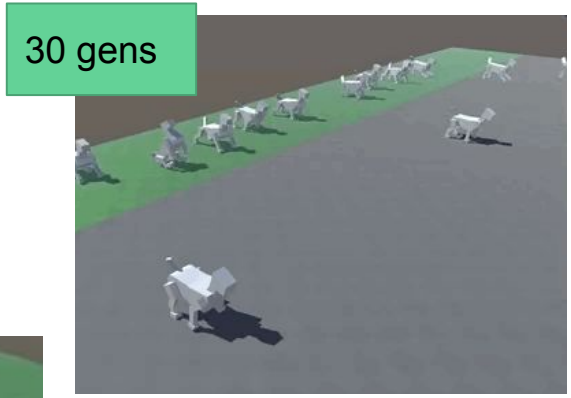
Results

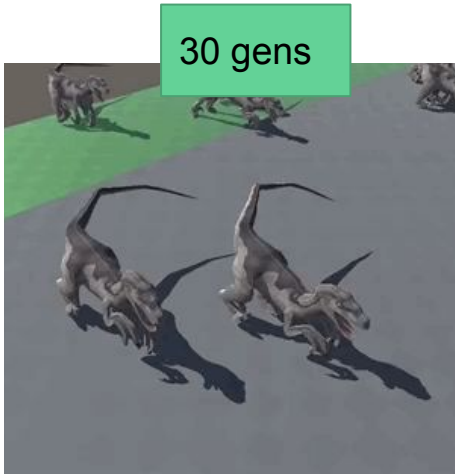
Characters

- Inchworm
- Remi (dog)
- Tiger
- Monster
  - Odd proportions, odd solution
- Chance (the Raptor)
  - Long limbs biped stress test







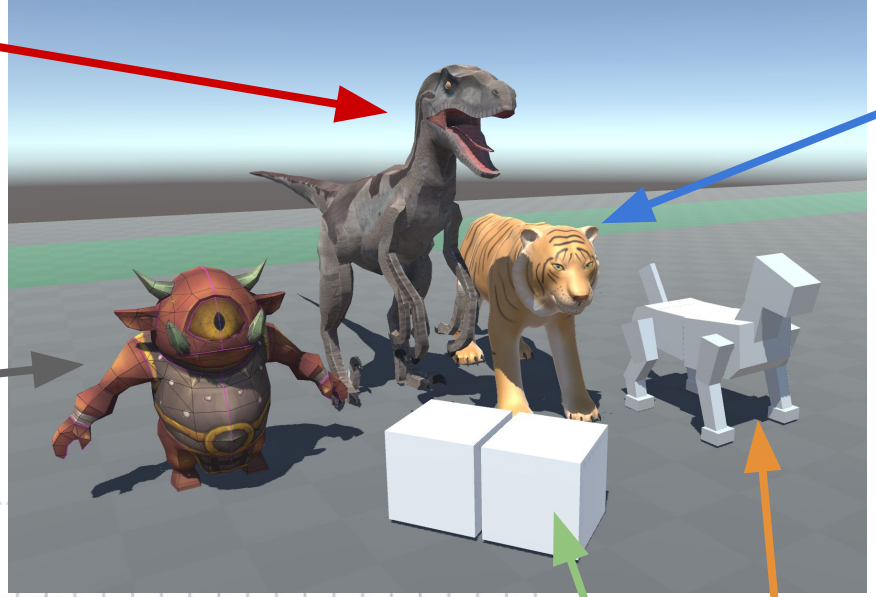


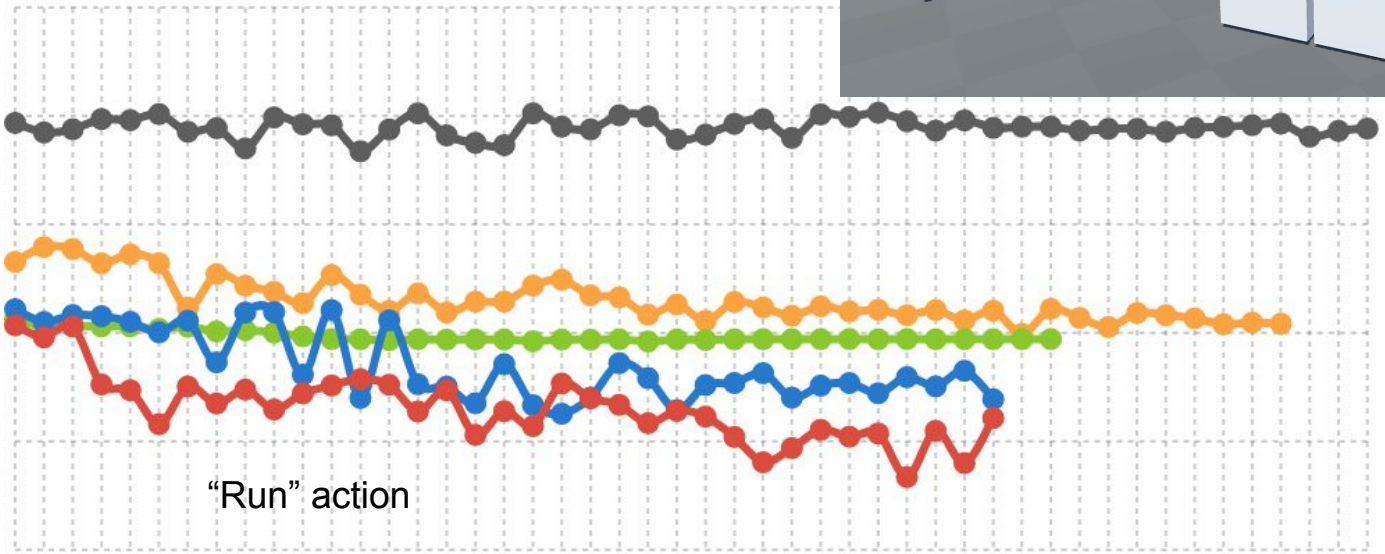
Data Example

Fitness over generations graph

Downward trend to asymptote

Lower is better

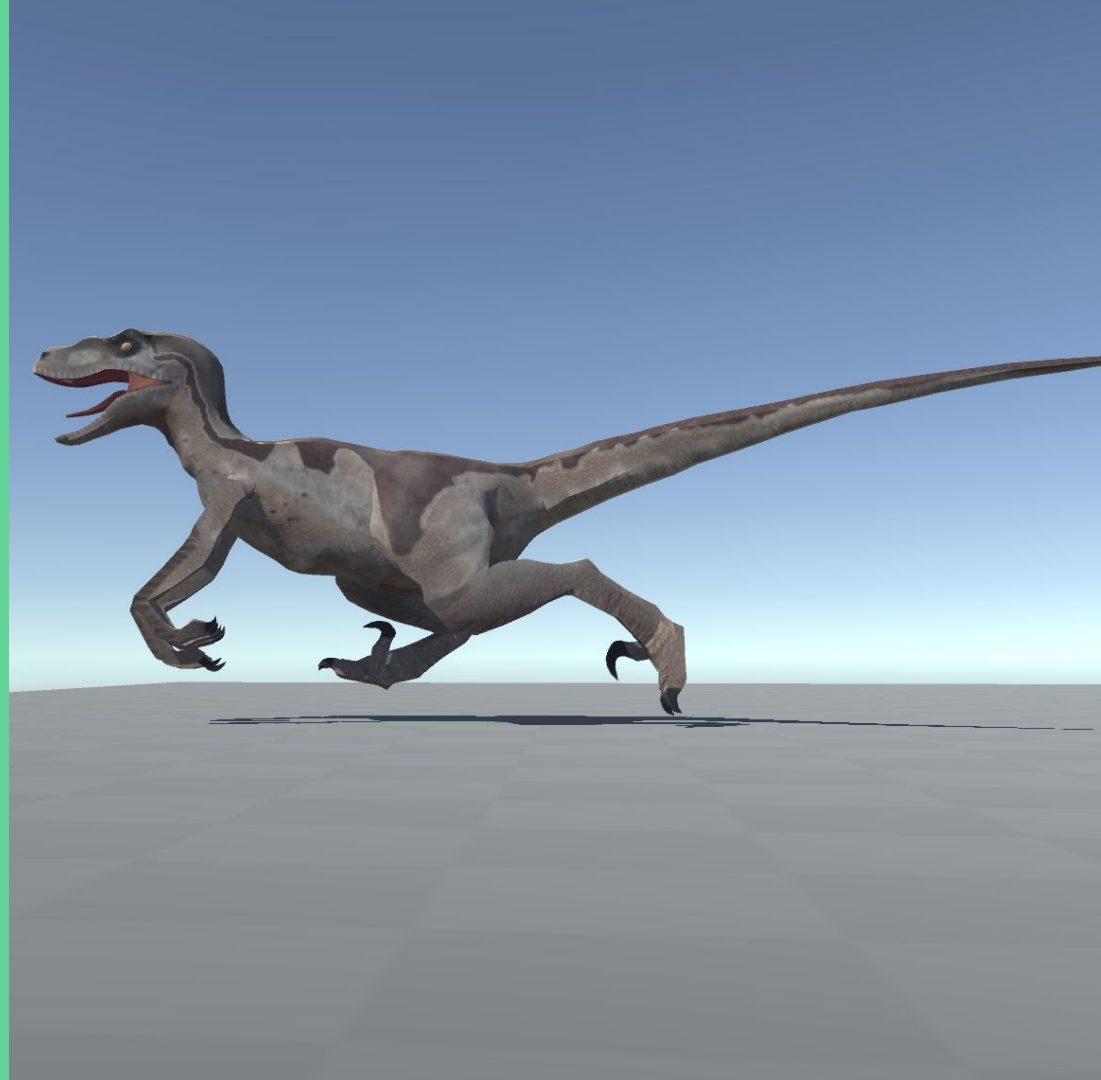




Responsibilities

Rob Quinn - Project lead, Sim lead programmer, client communications
Joe Sogard - Web lead, Back End programmer, Back End QA
Joe Kuczek - Full stack web, SCRUM master
Luke Oetken - Simulation programmer, Machine Learning, Status reporter
Andrew McKeighan - Simulation programmer, Quality Assurance
Kenneth Black - Simulation programmer, Machine Learning

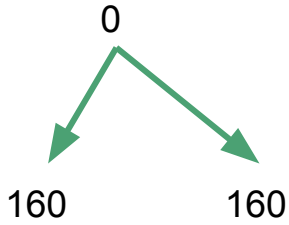
Questions?



Simulation Prototype - Divergent Behavior

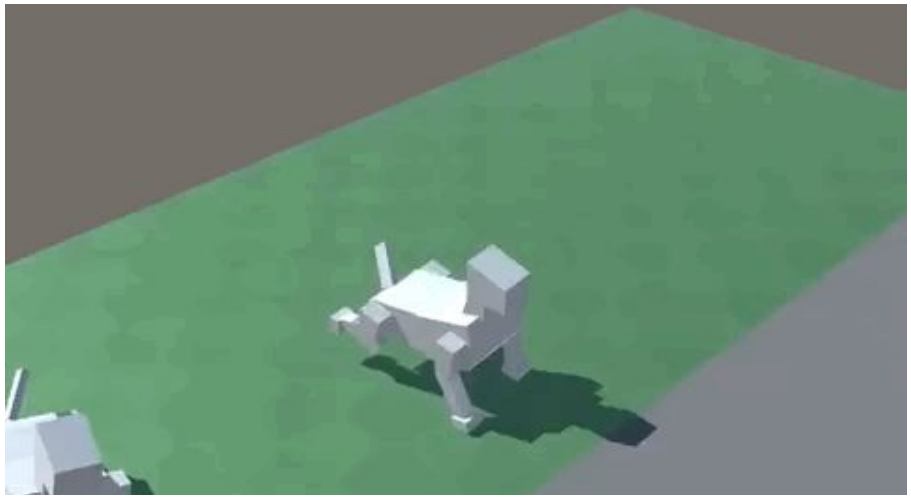
Remi @ Generation 160 Branch 1

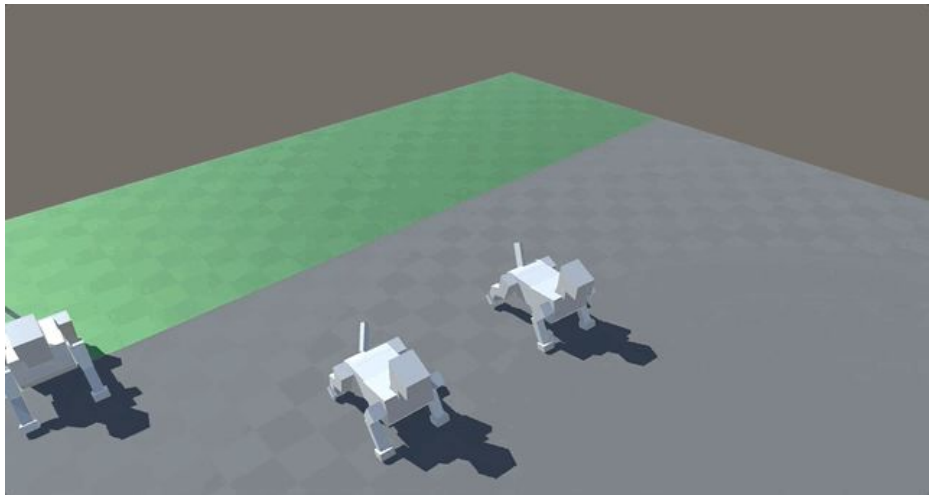
Back right leg up? Local maxima



Remi @ Generation 160 Branch 2

“Tippy taps” Local maxima





Simulation - Muscle Physics

Muscles - Robust and realistic

- Sinewave & target angle
  - Amplitude
  - Frequency
  - Center
  - Offset
- Tension-based torque custom equation
- Strength scaled by
  - Bone length
  - Body weight
  - Strength relative to body weight and gravity

Genetic Algorithm Mutation

Given information

- Max delta angle = 360
- Simulation time = 20 seconds
- Physics calculations = 100x per second
- Granularity = 0.00001
  (hundred-thousandth)

Mutation process

- Add random values to each parameter
- Scaled by power
- Small and large changes

Search Space

- Muscle Parameters
  - Amplitude [-360, 360]
  - Offset [-20, 20]
  - Frequency [0.05, 100]
  - Center [-360, 360]
  - 720 * 40 * 100 * 720 * 100,000^4
    = 2.1 * 10^29 per muscle
- Inchworm (1 muscle)
  (2.1 * 10^29)^1
  = **2.1 * 10^29**
- Remi (11 muscles)
  (2.1 * 10^29)^11
  = **3.5 * 10^322**

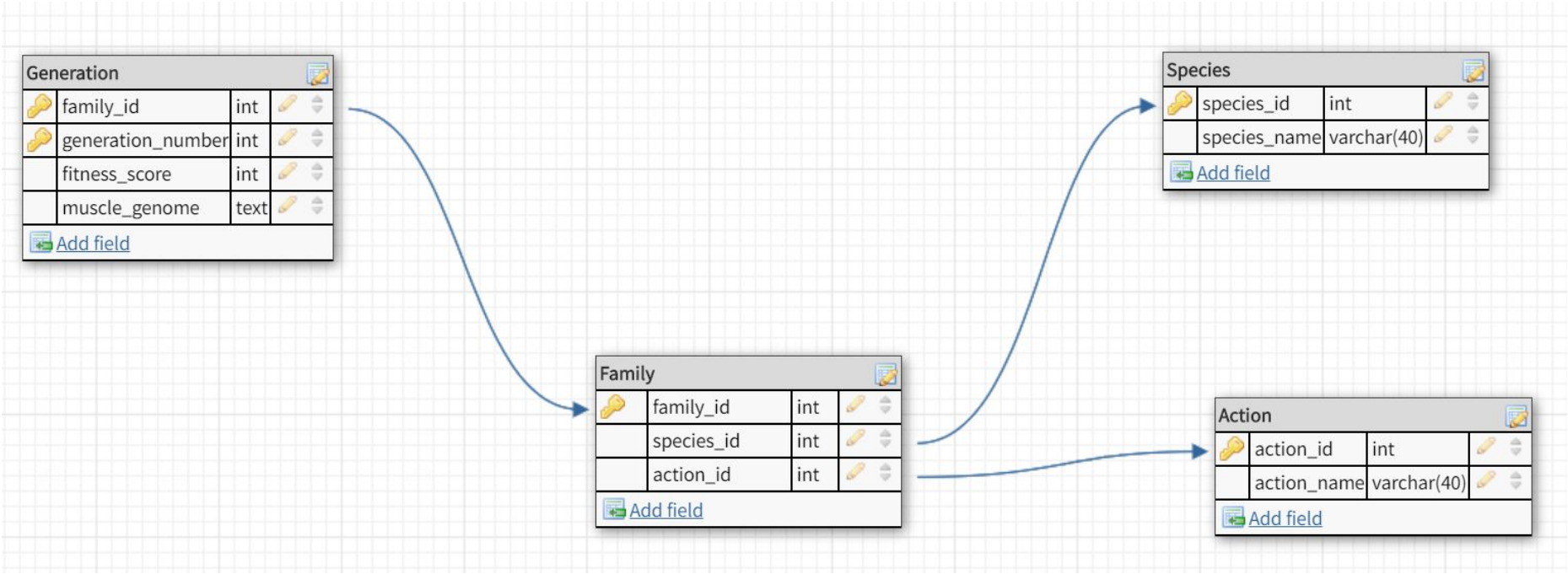
Genetic Algorithm Selection

- Fitness function
  - Modular
  - Difference of
    - position and target position
    - delta position and target delta position
  - Weighted values
- Best N
- Breeding (Crossover)

Team Workflow - Scheduling

A Section of our team's weekly schedule. Team goals, risks/issues, and individual tasks.

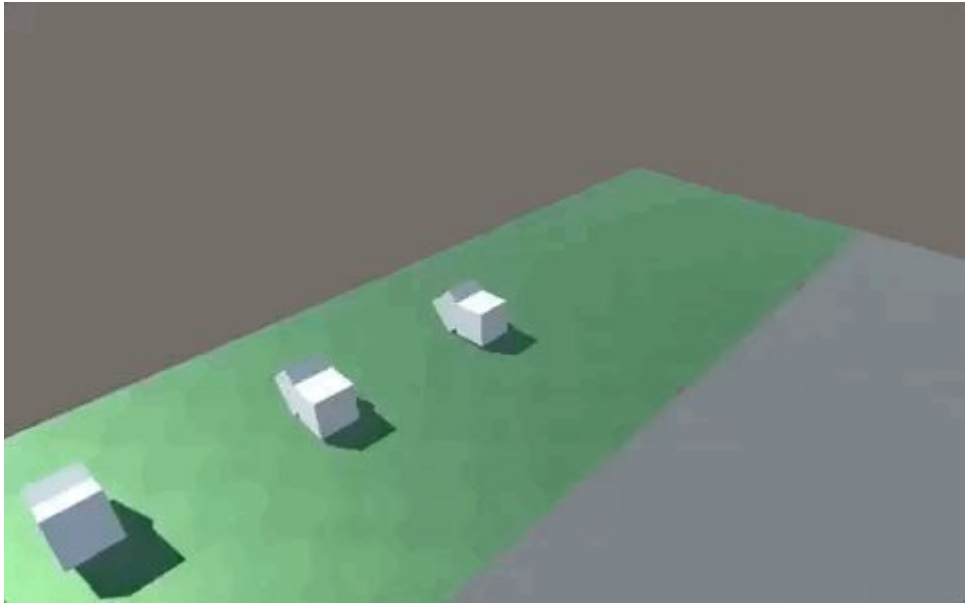
Tasks this week	Oustanding Issues	Luke - Crossover/Breeder	Ken - Nature/Mutator	Andrew - Fitness Functions	Rob - Modeling and Fill	Joe - Back End	Joe - Front End
Refine project purpose Luke - Get information about what kind of ML Unity agents is and why we arent using it, add to docs,Indicate which weeks have a bi-weekly report in column B	Upgrade to Unity 2017.3.0 and make sure it works	best k - instead of one best animals per generation, make it save for example the 3 top performers (variable). Will need this for breeding so there is more than one smart parent	Add granularity to mutation. So if it is 0.0001 granularity, random values for muscles will be an integer * 0.0001	make fitness functions goal based where 0 meets the goal and a negative value is how far away (basically reverse the existing fitness function)	Test 2017.3.0 Pass through documents with updates and todo notes Find references on physical animation	Research how to interface C# and server	Have a variety of graphs dis
Adding features to Genetic Algorithm		Research GA crossover methods https://en.wikipedia.org/wiki/Crossover_(genetic_algorithm) Probably uniform where the 'bits' are the values of the 4 muscle parameters, and a random mixing ratio	Add limits to each of the 4 muscle values. This is in our presentation slides but muscle center is [-360, 360] etc	Fitness function objects with target position, weighted	Prepare game model of character for retargetting Procedural skeleton setup	Plan the format of data so Joe can graph it	Work on using our data to d
Focus on mutations and crossover to make algorithm solutions come faster Submit status report		Have uniform crossover done for random combinations of parents	Research what other mutation functions we can use besides +/- random values https://en.wikipedia.org/wiki/Mutation_(genetic_algorithm)	Add parameters to limit undesirable behavior - cartwheels, falling over. Think of other undesirable behaviors that might occur	Model for chance the raptor Rigged mesh for chance	Continue implementing with our data	Continue implementing with data
Focus on integration of web and simulation, as well as each simulation component working at once		averaging - for every animal/genome, make n instances of it (identical but spread out), then during evaluation average the scores (this will smooth out physics inconsistencies that occur from floating point differences on the plane)	Implement some more interesting mutations	Add termination options. - Number of generations - When the goal is met by some margin like within 1%	Physics and tesing chance	Have one page done that shows a character's data and runs	Have one page done that sh a character's data and runs


Generation
family_id
int
generation_number
int
fitness_score
int
muscle_genome
text
Add field
Species
species_id
int
species_name
varchar(40)
Add field
Family
family_id
int
species_id
int
action_id
int
Add field
Action
action_id
int
action_name
varchar(40)
Add field

Simulation - Rapid Testing

Inchworm

- Simple animal
- Validate algorithms
- Learns to walk in
  - 5 seconds real time
  - 20 seconds sim time (4x)



Real-time gif (not time lapse) Gen 0-1

Team Workflow - Agile Method