Physical Character Animation using Machine Learning

Project Plan

Team 4

Client/Advisor: Jim Lathrop

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

sdmay18-04@iastate.edu

https://sdmay18-04.sd.ece.iastate.edu/

Revised: 2/20/2018

Table of Contents

List of Figures

List of Tables

List of Symbols

List of Definitions

- Animation - character movements for a game or media
- Keyed animation - animation made by interpolating between manually placed keyframes
- Keyframe - represents a character’s pose at a given frame
- Physical animation - animation where the character’s movement is driven by physics and forces
- Physically based animation - animation where physics were studied or used to create keyframes

1 Introductory Material

1.1 Problem Statement

In modern video games there is an increasing need for high-fidelity physical and physics based animations for characters. These animations can be extremely time consuming and expensive to make using traditional methods such as hand-keying or motion capture, and require many professional animators. Also, most existing procedural animation tools only work on human skeletons.

This project creates physical animations for video game characters using genetic algorithms. Characters have realistic 3D physicality and learn coordinated muscle-based motion to satisfy a goal such as creating a walk cycle.

1.2 Operating Environment

This is a virtual simulation rendered on the Unity 3d game engine. The engine creates most of the graphical interface, with our code being written on separate script components written in C#.

1.3 Intended Users and Intended uses

Our intended users are game developers who need animations for their characters. Many games today use physical animations where the movement of the character is driven by physics. Other games that use keyed animations may use physics simulation to make their animations realistic. Animations and especially walk cycles are time consuming and may require a dedicated professional to make them. Our application will take characters and simulate them to learn physically plausible movements which game developers can then use in their game.

1.4 Assumptions and Limitations

Assumptions:

- The cost will be negligible considering we are not working for pay, using free tools, and the web server is provided.
- The simulation can be run faster than real time in order to get the data we need.

Limitations:

- There is a fixed amount of locomotion behaviors before the animal enters the behavior simulation, including walk, run, and sit.
- There are at least 3 animal species to test including dog, snake, and inchworm.
- Limited by the amount of training data.

1.5 Expected End Product and other Deliverables

End product will be a virtual simulation of character movements using a genetic algorithm/machine learning implementation. The product will be delivered at the end of our second semester in May. We will have the core functionality completed in February.

2 Proposed Approach and Statement of Work

2.1 Functional requirements

The simulation shall simulate at least 3 characters with varying skeletons.

The simulation shall result in learned animation movements.

The database shall hold at least 500 generations for analysis.

The website shall show a graph of the generations of animals to show improvement over time.

2.2 Constraints considerations

If we use code from an outside source we will provide necessary attribution to the original author.

The website used to observe the development of the models must be easy to use and understand for a user with no technical experience.

2.3 Technology considerations

Our technology stack is Unity and C# for the simulation. Unity will be our 3D environment for the simulation that handles rendering and physics with NVidia PhysX. This engine is very versatile and can handle a physics simulation well. The physics is stable at high speeds so we can run the simulation faster than real time.

Using Unity and the built in physics engine will be good for the reliable integration. However, it is difficult to modify the physics engine. We can write our own extensions but if we run into physics bugs some may be out of our control.

As a design alternative, we could have used OpenGL instead of Unity. One problem we faced when researching OpenGL is we would have to create a physics engine, and we would have to spend a lot of time rendering the animals to the screen. Unity takes care of that for us, saving us a lot of time.

2.4 Safety considerations

Since this simulation is purely virtual and does not exist in the physical world at all, the only safety considerations have to do with our data. We must consider that someone with malicious intent will try to corrupt our data. In order to prevent this we will keep our passwords private and our RESTful calls will only be able to be called by authenticated users.

2.5 Previous Work and Literature

Rob made a very basic machine learning project for HackISU in under 36 hours which serves as our proof of concept and some of the inspiration for this project. We have a general idea that physics simulations are good and reliable in Unity and how to solve

some of the anticipated problems, such as how to run the simulation faster. This proof of concept has many flaws because it is from a hackathon, such as a sloppy save and load mechanic, which needs to be redone with proper genome serialization. See reference 4.2.1 for the proof of concept.

Using a simulation approach, genetic algorithms are used today to solve many problems including project scheduling (see http://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=535089).

It is not uncommon to be able to find a walking program that uses machine learning to make a model walk. Google's Deepmind AI recently learned how to make a stick figure walk, run, jump, and climb without any prior guidance. (https://deepmind.com/blog/producing-flexible-behaviours-simulated-environments/)

2.6 Possible Risks and risk management

Scope issues, engine issues and unforeseen bugs are all concerns that may hinder our progress. Another risk is the possibility of having issues in the technology when communicating between the Unity program and the website.

2.7 Project Proposed Milestones and evaluation criteria

We will have a prototype of our project done by the end of December. The prototype should have at least one character learning to walk successfully. The data should be sent to the website to be analyzed and display on the page in a graph or table.

The end milestone is a finished simulation with learned animations for a variety of characters. The data should be sent to the website and have detailed analysis and charts.

We will evaluate the quality of our software by adhering to IEEE standards related to machine learning and genetic algorithms. None of our processes can be considered unethical according to IEEE.

2.8 Project tracking procedures

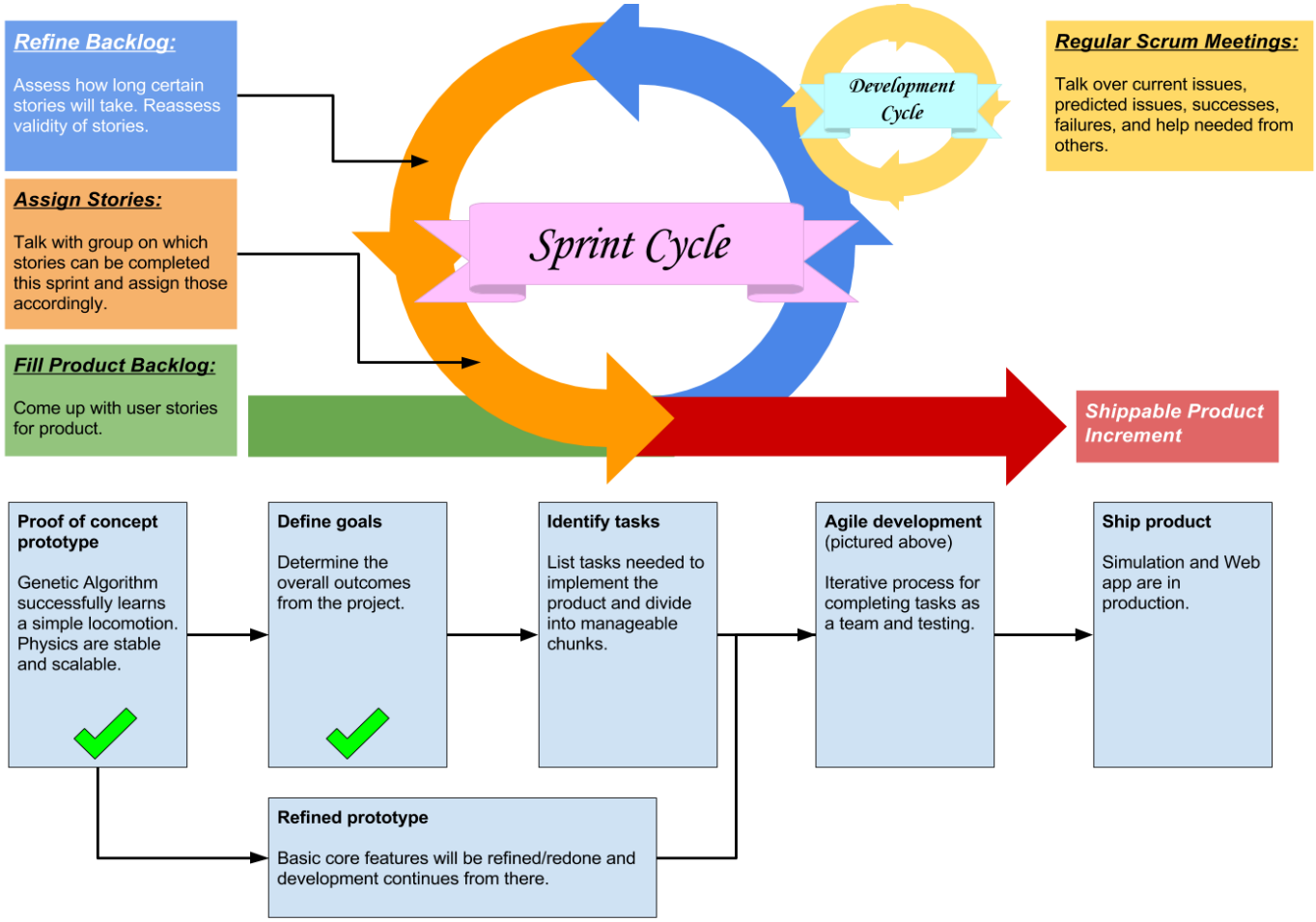
We have a Trello set up with Joe as our scrum master; keeping our tasks assigned and organized. We're using Gitlab as our version control to keep our work merge and updated. We have weekly meetings to check in on individual progress.

2.9 Objective of the task

We are creating a simulation with a genetic algorithm which generates physical animations for characters. It will send results to our web server for analysis.

2.10 Task approach

We will use GitLab as source control for our code. We will use Google Drive as a shared drive for project documentation and planning. We will use Trello for organizing our tasks.



2.11 Expected Results and Validation

We expect the outcome to be a functional simulation and web portal for analysis. The simulation will be very obvious if it works because the animals will learn to walk and do other activities rather than sitting still. The web portal should display graphs and data about the simulation so it can be analyzed.

3 Estimated Resources and Project Timeline

3.1 Personnel effort requirements

Member	Tasks	Explanation

Rob Quinn	Project/Simulation lead	Leading simulation programming, determining feasibility and new features
Joe Kuczek	Web Design/ Scrum Master	Make website, organize tasks
Kenneth Black	Simulation/ Machine Learning	Push for new development in machine learning algorithms and deep learning models
Andrew McKeighan	Simulation Development	Genetic algorithm programming
Luke Oetken	Simulation/ Machine Learning development Status Reports	Work on machine learning framework development, genetic algorithm development and animal models. Write weekly status reports.
Joey Sogard	Web Design	Create website backend and integration

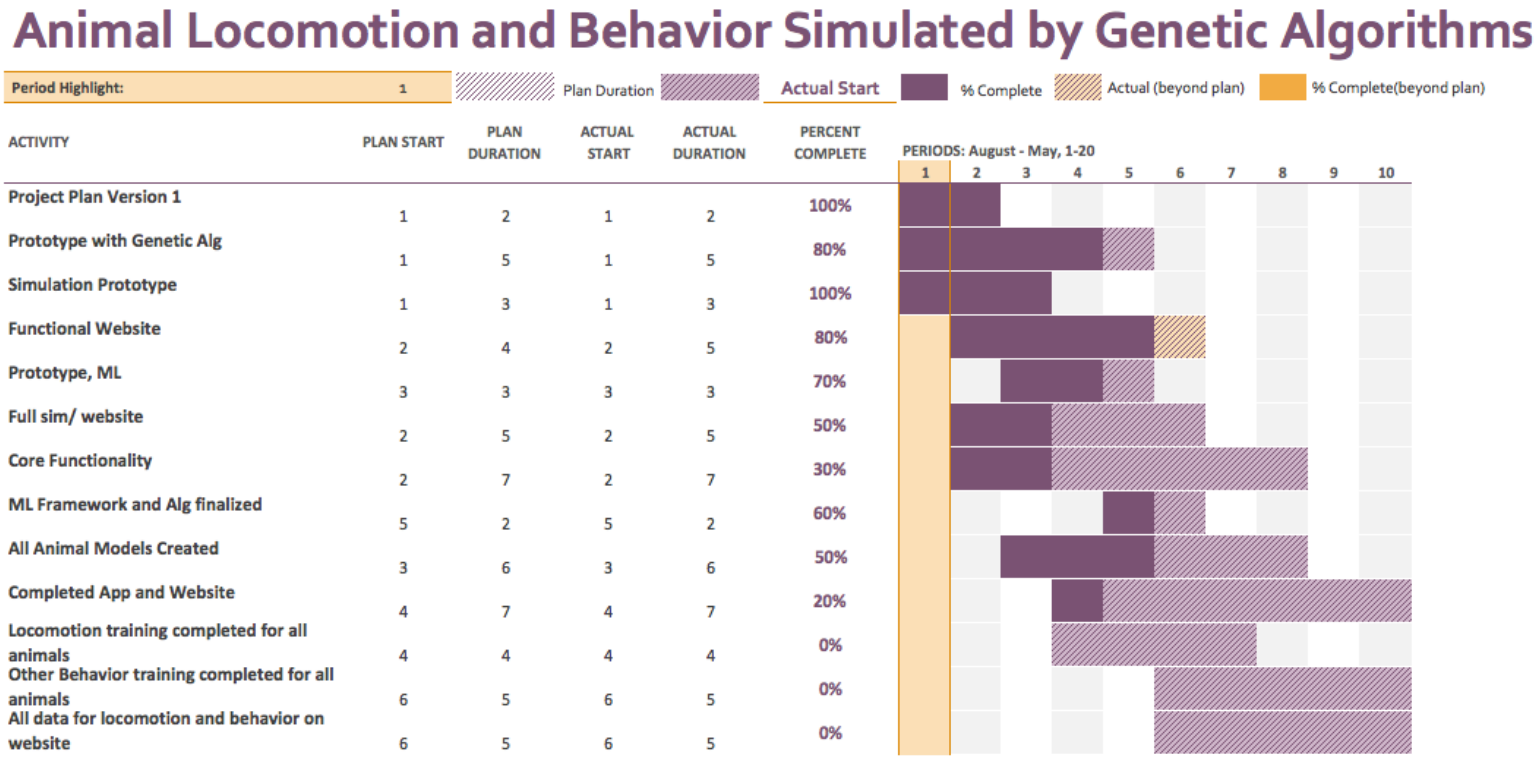
3.2 Financial requirements

The only financial requirement for this project is for the use of a dedicated server and database, but that will be provided by the university free of charge for us.

3.3 Project Timeline

- Project Plan Version 1 - September
- Prototype with some genetic algorithm and website running - December
    - Simulation prototype - October
    - Functional website - December
    - Prototype with different machine learning frameworks - December
    - Full simulation/website integration - January
- Core functionality done - February
    - Machine learning framework/algorithms finalized - February
    - All character models created - February
- Completed application and website - May
    - Locomotion training completed for all animals - March
    - All data for animation available on website - April

Gantt Chart:



4 Closure Materials

4.1 CONCLUSION

Our project will simulate characters with genetic algorithms for the purpose of creating physical animations. We will deliver a simulation app that can run different characters in the environment and a website to upload and analyze the data. Our team has experience with Unity, machine learning, and web, so we are qualified to make this project a success.

4.2 REFERENCES

4.2.1 Proof of concept dev-post
https://devpost.com/software/machine-learning-first-steps

4.2 APPENDICES

Unity
https://unity3d.com