A User-friendly Platform for Visualizing Tree Growth

By TreeViz: Riley McWilliams, Alexander Bentley, Daniel Rustrum, Haitian Tan, and Qi Han

Clients: Dr. Kiona Ogle & Dr. Michael Fell | Mentor: Isaac Shaffer | Professor: Dr. Eck Doerry

Background

Climate change is causing damage to Earth and is largely caused by rising carbon dioxide levels in atmosphere. Trees play an extremely important role in Earth's climate behavior. They cover around 50% of Earth's land area and contain upwards of 90% of the global vegetation carbon. This means that a lot of CO2 that humans emit is being taken in by trees and used for their growth. Observing tree growth patterns can lead to a better understanding of how trees and climate interact with each other. These observations can also be used to predict the effects of climate change in the future.

To better understand how certain factors affect trees, Dr. Kiona Ogle and Dr. Michael Fell of Ogle Labs developed a simulation that shows a tree's growth over time. The simulation is called the Allometrically Constrained Growth and Carbon Allocation (ACGCA) model. It uses over 30 input parameters to run the simulation, which calculates the state of the tree over time. The output of the model contains useful information such as the tree's height and radius, and the amount of carbon in the leaves and trunk.

Motivation

The ACGCA model was purely used for research by Dr. Ogle, Dr. Fell, and Ogle lab associates. However, they wanted to expand the use of the model to anyone with an internet connection, for free, allowing everyday people to learn about tree growth from it.

The problem was that the ACGCA model is not user friendly, which limits who can utilize it. Running the model requires the user to have programming experience, and the outputs require knowledge in tree biology to understand. Additionally, the ACGCA model was not available online.

```
void growthloop(
    sparms *p, gparms *gp, double *Io, double *r0, int *t, double *Hc,
    double *LAIF, Forestparms *ForParms, double APARout[], double h[],
    double hh2[], double hC2[], double hB2[], double hBH2[], double r[],
    double rB2[], double rC2[], double rBH[], double sw2[], double vts2[],
    double vt2[], double vth2[], double sa2[], double la2[], double ra2[],
    double dr2[], double xa2[], double bl2[], double br2[], double bt2[],
    double bts2[], double bth2[], double boh2[], double bos2[], double bo2[],
    double bs2[], double cs2[], double clr2[], double fl2[], double fr2[],
    double ft2[], double fo2[], double rfl2[], double rfr2[], double rfs2[],
    double egrow2[], double ex2[], double rtrans2[], double light2[],
    double nut2[], double deltas2[], double LAI2[], int status2[],
    int errorind[], int growth_st[]
)
```

The ACGCA model's input. It required the user to have experience in programming to run it.

The ACGCA model's output. It was only available in raw data format, requiring the user to have biology knowledge to understand.

```
p: sparms, input parameters
gp: Vector: (timestep, years, tolerance, breast.height,parmax)
r0: The starting radius (m).
h: A time series of tree height from the simulation for each time step. The length is steps*
      years+1 due to the initialization (time 0) (m).
hh: Height at which trunc transitions from a paraaboloid to a cone. Also the height to the
      base of the crown (m).
r: A time series of tree radius (m) from the simulation for each time step. The length is
      steps*years+1 (time 0).
rB: Radius at the tree's base (m).
rBH: Radius at breast height (3.37 m).
sw: Sapwood width which has a maximum of SWmax (m).
vts: Volume of trunk sapwood (m^3).
vt: Volume of trunk (m^3).
vth: Volume of trunk heartwood (m^3).
sa: Sapwood area at base of trunk (m^2).
la: Total one-sided leaf area (m^2).
ra: Fine root area (m^2)
dr: incremental increase in radius
```

Solution

A Web Application

TreeViz has spent the past 2 semesters developing a user-friendly web application that takes the ACGCA model and provides a method of utilizing it that is more accessible to the everyday user. The user no longer must know programming to run it, only needing to enter or adjust the default values of the inputs in a graphical user interface as shown in the image below. In addition, a background in biology is no longer necessary to understand the output data. While we do still include the raw numerical data for research purposes, we have developed a visualization that renders the tree at each year. This allows the user to actually see the tree that was simulated from their inputs.

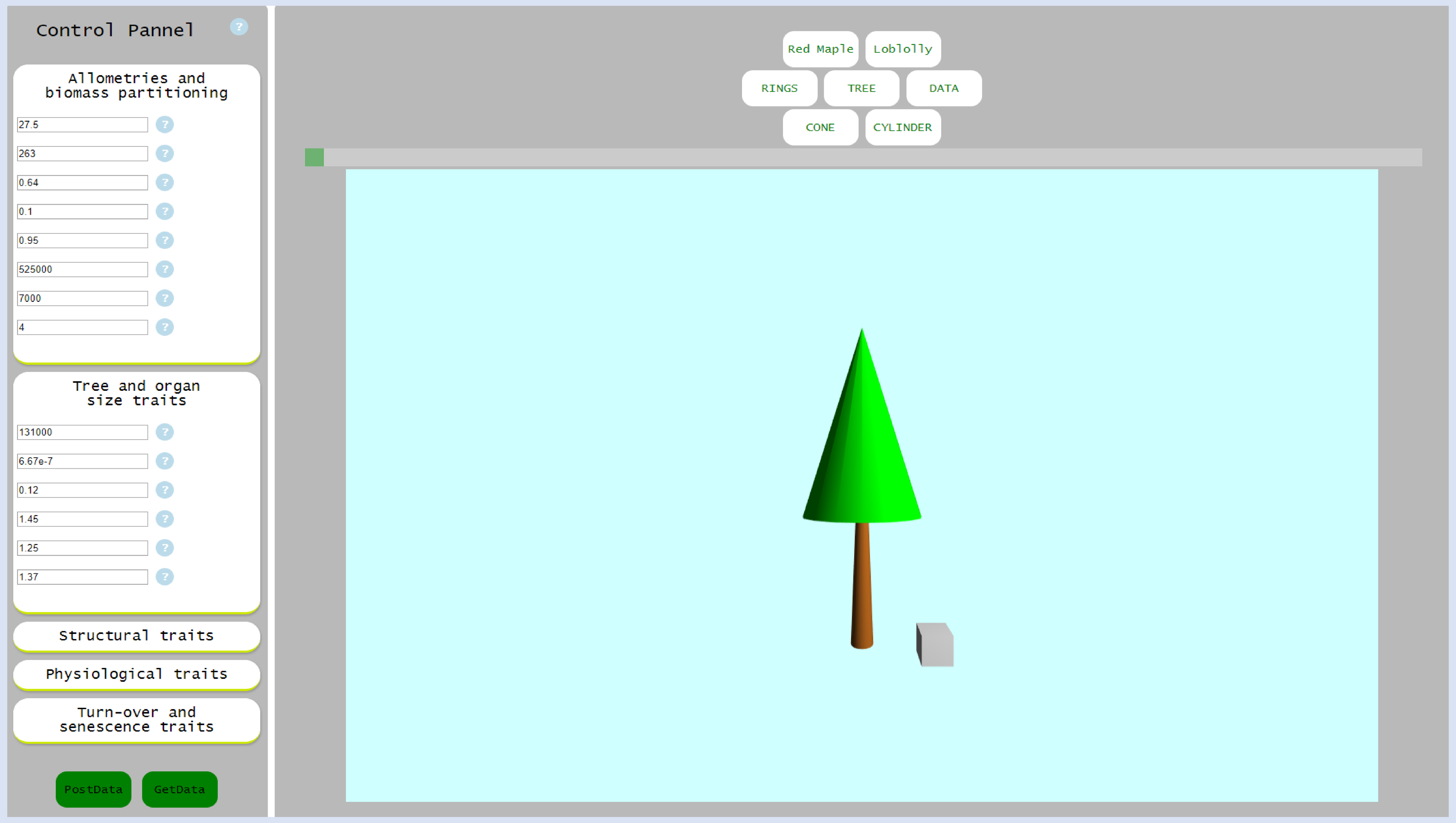
Key Features / Technologies

This product has many features that increase the usability of the ACGCA model for the everyday user.

- User-friendly website built in a JavaScript framework (Vue.JS)
- Organized input parameters with help bubbles
- Graphical tree built in a JavaScript graphics framework (Three.JS)
- A slider to watch the tree grow over time
- Available online (Amazon Web Services (AWS))

There are also features to help our clients with their own research.

- Numerical data output option (HTML)
- User login/survey to analyze demographics of use (Firebase)

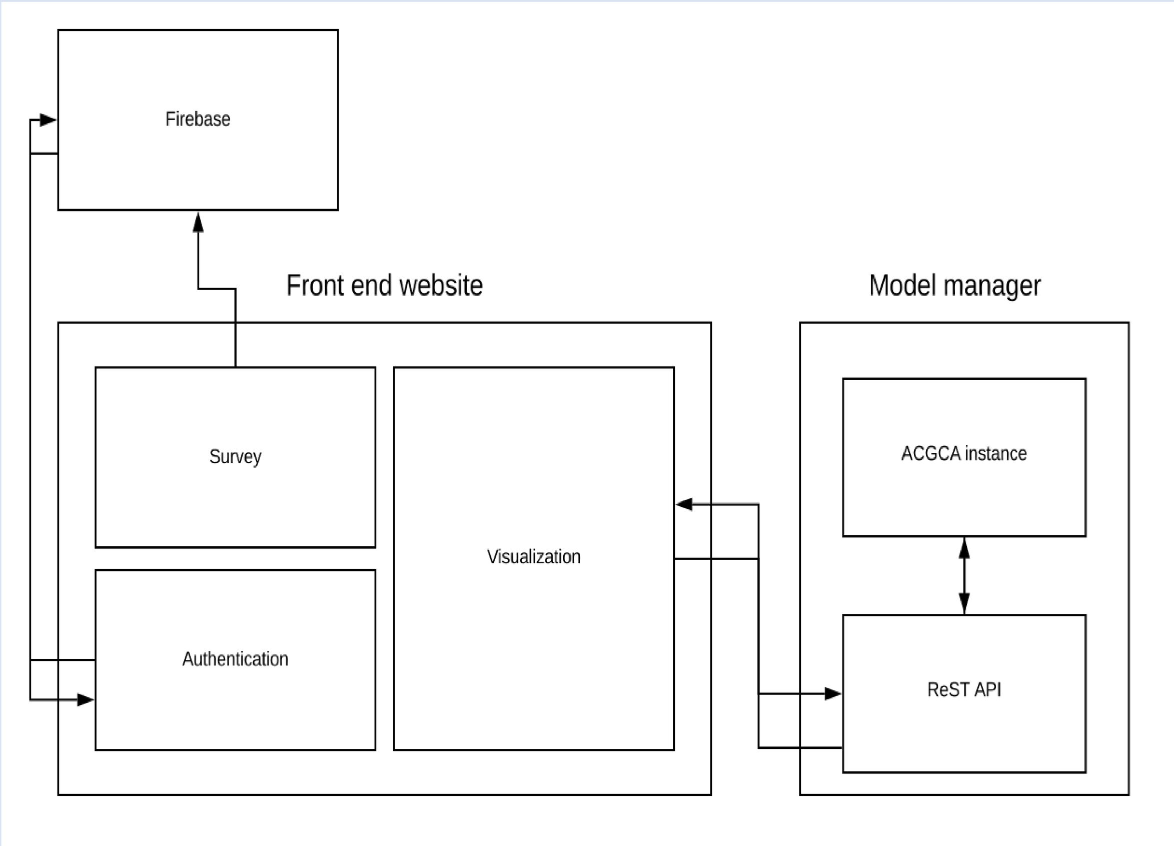


Architecture

Through weekly meetings with our clients, we gathered the requirements necessary to fully realize the product that they envisioned. The key requirements are:

- Access to the ACGCA model via internet
- User surveys for analytics / User login
- Visualize growth of tree over time
- Data transferring to and from the ACGCA model
- Cheap (low budget)

With these main requirements gathered, we came up with the basic architecture of the system as shown on the right. The central component is the front-end website, which includes log-in and survey functionality through a online database called Firebase, as well as the visualization. The model manager component contains the ACGCA model and a data transferring tool (ReST API) to handle communication between the model and the website.



Architecture Diagram

Challenges

One challenge we faced was a development issue with the tree visualization. At first we decided to use a game development engine called Unity for its built-in tree creator tool. However, we were unable to access the parameters necessary to generate trees procedurally and had to find a new tool. We found a 3D graphics JavaScript framework (Three.JS) that could fit perfectly with the rest of our JavaScript-based website.

Another issue we had was finding a way to store user data for login and survey analytics. We originally used MySQL as the database, but were having trouble integrating it into our system. After realizing that it would not work, we went with another option that we researched: Firebase. Firebase was significantly easier to implement, and allows our client to easily analyze survey data.

Testing

By conducting user testing, we will be able to get an idea of which parts of the website are user-friendly as they are, and which parts need improvement. This will allow us to focus our effort on making final upgrades on the most important features of the system before the semester ends. Additionally, we will be conducting unit testing and integration testing. These will allow us to double-check that each function works as intended and that the system works as a whole, respectively.

Value

The ACGCA model provides vital information for understanding tree growth. Now that it has been wrapped with more user-friendly input and output, and is being hosted online, the everyday user can access it and learn on their own. The graphical tree and intuitive slider makes it easier for users to experiment with how certain factors affect the growth of trees over the years.

With more people using their model, our clients hope that more people will become educated and interested in tree growth and the environment in general.

Future Work

Once we deliver the final product to our clients, they will be able to spread word of it so that more people will access the ACGCA model and learn about tree growth. They will also be able to conduct their own research in a much easier way, saving the time of entering all of the inputs via code and gathering the numerical output.