

Guayaki the Game



Members: Matthew Lehmann, Sam Ehrlich, Tri Bui, Luiz Ramirez,
Jordane Coombs

Wireframe

THE WEBSITE:

Guayaki the Game Homepage

Guayaki Home My Profile

Username Password Login

Welcome to Guayaki!
A new community-driven PC game designed to provide gamers with a fun, challenging, and competitive environment.
Brought to you by: Matthew Leitman, Sam Ehrlich, Paul Rancarz, Jordane Corbitt, and Ti Bo

USERNAME	BEST TIME
chilling	
Wasting	
more, the, guy	
DrFoolish	

Login

Username

Password


☒ Remember me

Sign in

f Sign in with Facebook

G Sign in with Google

User Profile



Placeholder text for user profile bio

Possibly a social media/posting option for bio

THE GAME:



Tools

- Project Tracker: Smartsheet (Gantt Chart)

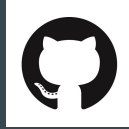


- Rating: 3 - Smartsheet was a simple way for us to set an agile/scrum schedule at the beginning of the semester for us to follow

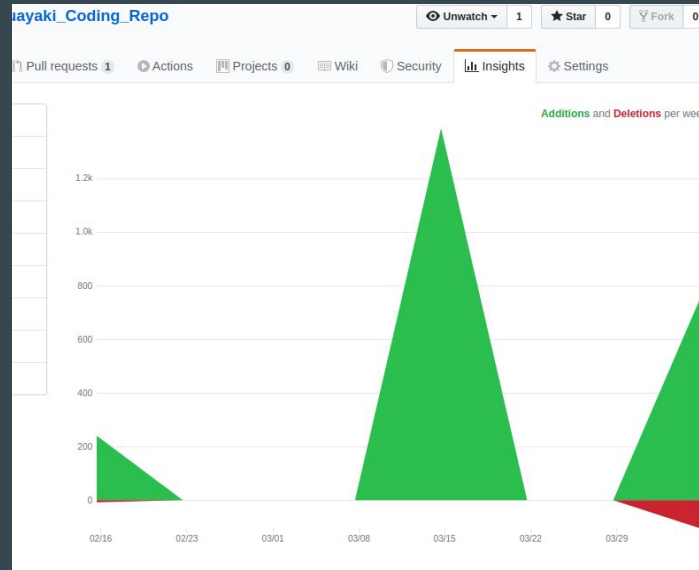
Task Name		Feb 9							Feb 16						
		S	M	T	W	T	F	S	S	M	T	W	T	F	
1	Game interface														
2	Obstacle Course - Course itself, Speed boosts, player, platforms/ramps													Sa	
3	GUI - Welcome Menu, Options Button, in Game Menu Options														
4	VFX, SFX: Lighting, object movement, map design, color schemes, etc														
5	Video Recording														
6	Online database for the high scores														
7	Social Website														
8	Account Registration within the website:														
9	Integration with back end and game														
10	Downloadable on steam or other market														

Tools

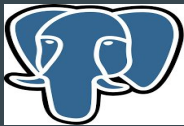

- VCS Repository: GitHub



- Rating: 5 - Github was obviously very useful in developing the project. It provided an effective means for keeping track of all of our ejs and HTML/CSS code for the website.



Tools

- Database: PostgreSQL 
- Rating: 5 - Very simple and useful database manager
- Another Custom Repository: Unity 
 - Rating: 4 - Unity actually has its own means for keeping track of code from multiple users. Sam, Paco, and Tri used the multiple contributor functionality of the Unity gaming engine to share the game code amongst themselves so that all 3 of them could work on it.

Tools



- Testing Tool: JSfiddle
- Rating: 3 - JSfiddle provided a decent means for testing the robustness of our ejs code. Simply to ensure that the base JS within the files could actually be read appropriately in any environment.
- Deployment Environment: LocalHost
 - Rating: 5 - LocalHost is a very nice tool to have for deploying applications because there's no web service that needs to be utilized to handle running your program. For us specifically that is important because we are trying to make a game that communicates with a website, while both the website and game share a common database even though they aren't hosted on the same service

Tools

- IDE: Sublime and Unity



- Rating: 5 - Sublime and Unity3D were incredibly helpful in this project. Sublime provides a very simple IDE that allows for ejs, js, and HTML/CSS development. Unity on the other hand provides a very user friendly environment for developing 3D games that look aesthetic and feel great to play.
- Framework: Node.js
 - Rating: 5 - The server side scripting language we learned in class, so it made it relatively easier to incorporate into the project usefully



Methodology

- Agile
 - Rating: 4 - Agile was a nice way to work on this project because it focuses on the individual task achievement over the comprehensive documentation of everything we were doing. Having the team split up into website and game focused subteams, documenting all of the Unity code and EJS and HTML/CSS scripts together was a bit of a hassle. Using this methodology however we knew we could focus more on the project itself rather than documenting everything extremely well.

Challenges

- Website
 - Managing the node modules necessary to employ the express node.js library. Specifically just having to learn what a bunch of the modules actually did to know whether or not they were necessary.
 - Also finding online how this node modules folder can be downloaded before any node.js project rather simply.
 - Research was really the main way we overcame the issue and it ultimately didn't impact our project plans very much.

Challenges

- Website
 - Login page
 - It was a bit difficult to manage how logins from different users would be handled.
 - One idea was to monitor select results from the pSQL database. So no matter what, when a user enters a login name in the login forms, the database queries for any entry that matches the username. If the query returns an empty pSQL object, the website returns an error and asks for the user to retry.
 - This ultimately didn't affect the project significantly but this was the method used to overcome the challenge.

Challenges

- Game
 - Managing the contributors function in Unity so that multiple members could work on the game.
 - Through enough research the team ultimately figured out how to share the Unity files amongst one another.
 - But overall it changed how the development tracking tools were used throughout the project (ultimately making those tools used less than initially anticipated at the beginning of the semester) and also made the agile methodology more suitable for the project since documenting all of the necessary code was harder to do with Unity's contributor function helping us share code already.

Challenges

- Game
 - Another challenge with the game was figuring out how to connect the unity game to the pSQL database that the website had access to. So that when a user logged in, in the game, their score was sent to the database, and the website could report this.
 - Ultimately this was the main reason we couldn't deploy our app online like we wanted to because giving unity access to an AWS or Heroku based pSQL database proved too challenging
 - The game was ultimately kept local to handle this issue and LocalHost provided a nice means of accomplishing this.

DEMO TIME

...