Take a Virtual Hike

DESIGN DOCUMENT

Team 19
Client/Advisor: Mat Wymore
Team Members: Trevor Nemes (Team Leader), Tyler Hassfield, Opeyemi Abass, Aashutosh Mallik, Akhilesh Ratnakumar, Zian Li
Email: sdmay21-19@iastate.edu
Website: https://sdmay21-19.sd.ece.iastate.edu/

Executive Summary

Development Standards & Practices Used

- Development
  - Each team member will focus on researching a specific element of the project.
  - Team will utilize new information to generate a plan in regards to building and integrating a VR Environment modularly.
  - The development process will follow a Waterfall life cycle.
- Communication
  - Weekly Zoom meetings to cover progress, have open discussion, and create a plan for the following week.
  - Private chat for quick, informal communication.
  - Documentation of meetings
- Coding Style
  - Consistent naming conventions, descriptive commenting, and regular testing.
  - Have modular design and implement a design pattern.
- Digital design
  - Use of Perlin Noise Algorithm for generating visuals.
  - Use the Three.js graphics API
  - Implement procedural generation to have an ever expanding world
- 1362-1998 - IEEE Guide for Information Technology - System Definition - Concept of Operations (ConOps)
  - Provides a uniform scheme for preparing and presenting a concept of operations document
- IEEE 1448a-1996 - Standard for Information Technology - Software Life Cycle Processes
  - Establishes a common framework for software life cycle processes, with well-defined terminology, that can be referenced by the software industry
- IEEE 12207-1996 - ISO/IEC International Standard - Information Technology - Software Life Cycle Processes
  - Provides a common framework for developing and managing software. This standard provides industry a basis for software practices that would be usable for both national and international business

Summary of Requirements

- Explorable 3D world (first person perspective)
- Nature-themed world
- Full-scale environment, including life sized mountains and trees (1-to-1 scale)
- Use procedural generation when possible
- Basic collision detection
- Fly-through mode and walk-through mode
- No noticeable delay/choppiness while exploring the world (30 fps)
- Core code should not be platform specific
- Core code should be modular and extensible

Applicable Courses from Iowa State University Curriculum

- CprE: 185, 288, 308, 388
- ComS: 227, 228, 309, 311, 327, 329, 339, 352, 363

New Skills/Knowledge acquired that was not taught in courses

- Three.JS API
- Working with JavaScript language
- Working with 3D computer graphics
- Creating and working with procedural generation algorithms
- Creating and working with soundscapes

Table of Contents

List of Figures

1 Introduction

1.1 Acknowledgement

We would like to thank and acknowledge Professor Mathew Wymore for his time, effort, and support during the development of this project.

1.2 Problem and Project Statement

Problem Statement:

The COVID-19 pandemic is mentally and emotionally stressful, and has limited opportunities for activities such as vacationing and connecting with nature. This project aims to create an explorable, full-scale three-dimensional virtual nature environment for the purpose of relaxation and stress relief in the stressful time of COVID-19. Planned unique features include a 1:1 scale with reality (how big should that virtual mountain REALLY be?) and elements of procedural generation. Due to these features, the direct use of a low-level graphics API, Three.js, is expected, and various memory and CPU optimizations may need to be employed to achieve satisfactory performance. Programming languages may include javascript or others.

Project Statement:

To get this project done, we will be using the Three.js graphics API to design the world. We will design all aspects of the world on a 1-to-1 scale with real life, such as trees, mountains, clouds, etc. We will create an algorithm using procedural generation which will help generate all aspects of the world randomly so that everyone's world will be unique every time they play. We plan on implementing a fly-through mode and a walk-through mode so that the users can choose either one to explore the world. We expect our application to run at 30 fps with no visible choppiness or lag. Lastly, if we have ample time, we will implement a nature themed soundscape to the application as well.

1.3 Operational Environment.

The virtual reality system will rely on a web browser to be executed. The machine used to power the application must have a GPU and Processor powerful enough to support a virtual environment. This system will likely operate indoors and be sheltered from any harmful physical conditions.

1.4 Requirements

Functional Requirements:

- The User shall be able to load into a virtual Environment upon startup
- The User shall be able to move around and explore the world freely
- The User will have the ability to choose between a fly-through and walk-through mode
- The environment must be 1:1 scale with reality
- The Game must implement a soundscape for the environment

Nonfunctional Requirements:

- The rendering of the environment must be aesthetically pleasing
- The movement/usability while exploring environment must be simple and effective
- The application must run on a reasonably priced and attainable computer.
- The Game must contain elements of procedural generation

1.5 Intended Users and Uses

Our user base has the potential to be very large. Anyone who is trying to explore a virtual world for the purpose of relaxation as well as anyone who wants to explore an environment that is 1:1 scale is our user. Our platform is going to be used as a form of a stress relief tool that helps people escape the "'normal" world and presents them with a virtual world that relaxes their mind.

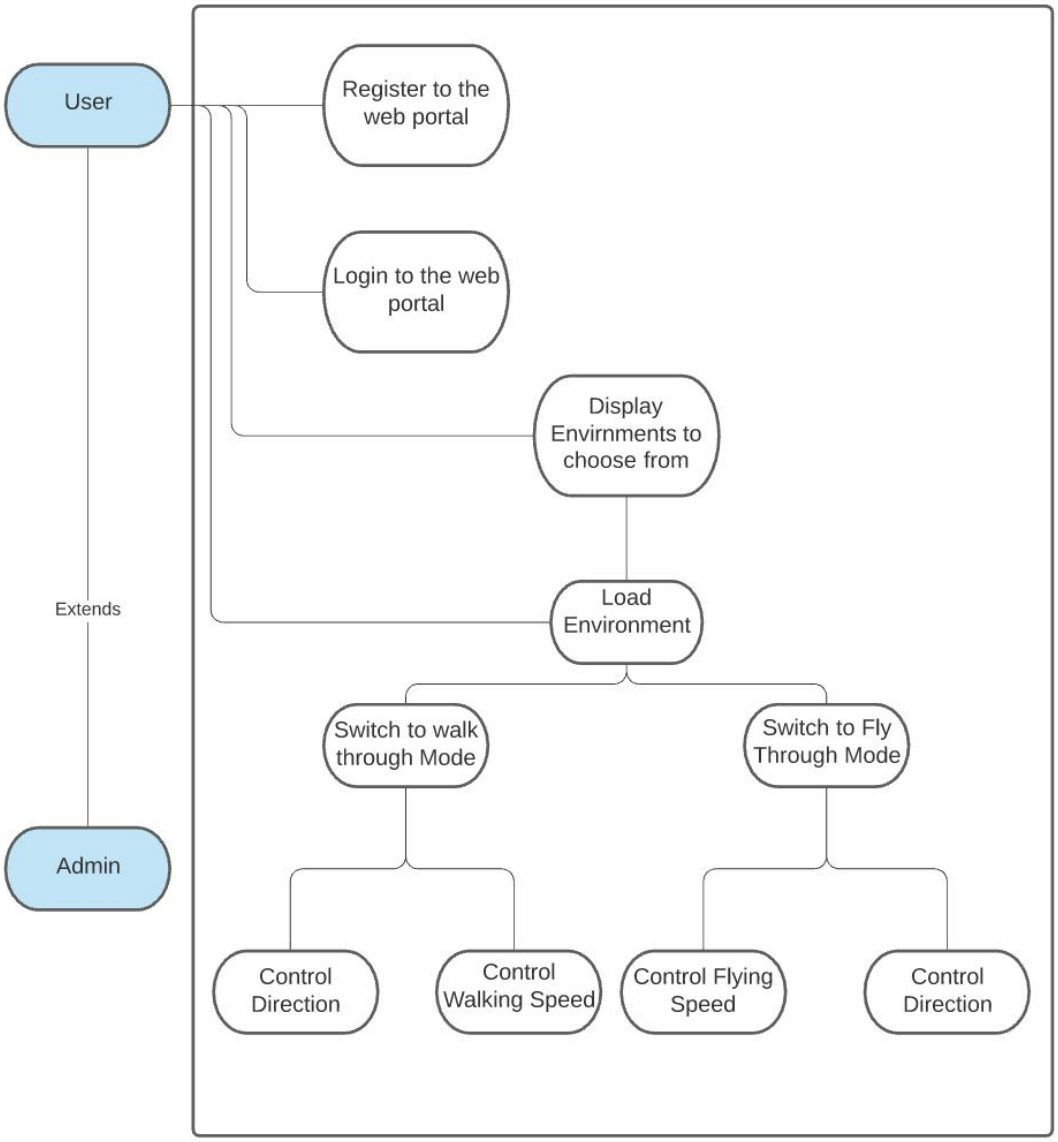


Figure 1. Use-Case Diagram

1.6 Assumptions and Limitations

When it comes to our projects assumptions, our final project will end up being a web application, so our game will be limited to running on a web browser rather than a phone application or something else like that. Our application will be a single player game, which means only one player can play on a browser at a time. Also this means we will not be implementing a multiplayer feature where people on other browsers can join your world. We will implement various buttons so our user has control of his movement in the game, this includes up, down, left, right, move forward, move backwards, the up and down buttons will only work when in fly mode.

When it comes to our project's limitations, the so-called hike in our game will be limited to one single location, so you can not move or fast travel to a different location while in the game. You will spawn in a forest of some kind and you will only be able to explore that forest, you can not change your location type, such as a desert or ice biome, while playing. Our game will be limited to a first person view, we will not be implementing anything else, such as a third person view. Also, the main thing is the fact that we only have two semesters to work on our project, which includes spending a lot of time learning stuff like our three.js 3D graphics API.

1.7 Expected End Product and Deliverables

Our deliverables consist of two products, The first one being the platform itself and the second being an instruction manual. We plan on releasing the first prototype at the end of first semester and the final product at the end of second semester.

We plan on delivering our product as a web platform. The platform will have two account types, "user" and "admin". The platform is going to have a 1:1 scale environment where the objects are created algorithmically using a procedural engine. The platform is going to be accessible to the user using a web portal. The user will have the ability to access the front end of the platform except. The admin will have access to the entire platform as well as the procedural engine which will allow the admin to tweak the parameters and type of the objects being created by it for the environment.

We also plan on delivering a manual to the client that will elaborate on the technologies used to create our platform. We also plan on including an instruction section in the manual as well as a walk through of the platform so that the client and the user is familiar with the product.

2 Project Plan

2.1 Task Decomposition

Tasks:

- Learning Three.js and WebGL (Research and Tutorials)
  - Includes learning about procedural generation algorithms
- Create and start coding the base environment of our application
- Create the procedural generation algorithm that will generate objects in the world
  - Have the algorithm create all aspects of the world at a 1:1 scale
- Work on user movement within the world
  - Implement a first person view
  - Create a walkthrough mode and flying mode
- Work on collision detection
- Create a nature themed soundscape for the world
- Test all of the applications features

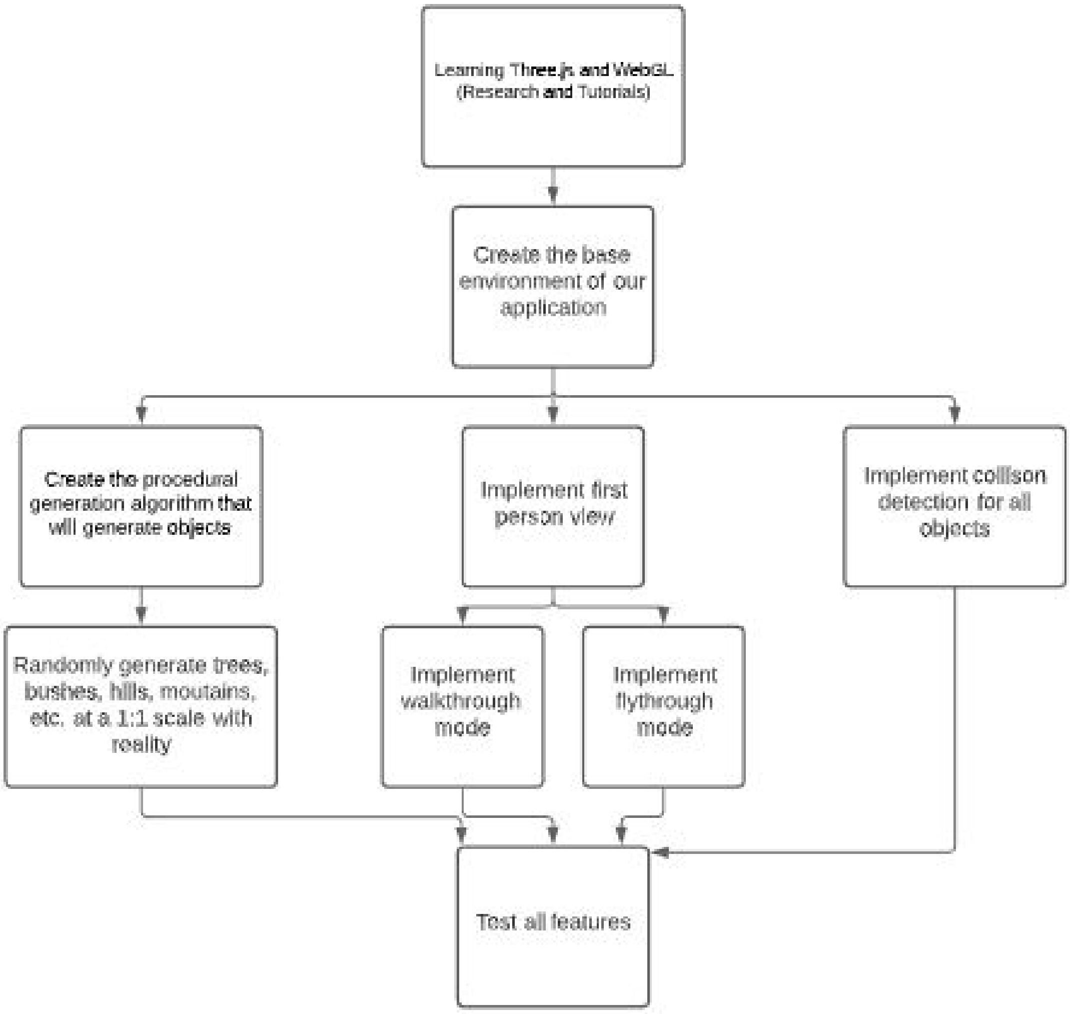

Learning Three.js and WebGL (Research and Tutorials)
Create the base environment of our application
Create the procedural generation algorithm that will generate objects
Implement first person view
Implement collison detection for all objects
Randomly generate trees, bushes, hills, moutains, etc. at a 1:1 scale with reality
Implement walkthrough mode
Implement flythrough mode
Test all features

Figure 2. Task Decomposition Diagram

2.2 Risks And Risk Management/Mitigation

This will be the team's first time working with Three.js and WebGL, and a few of us are familiar with Javascript. As such we have the risk of dealing with a steep learning curve and its hurdles to overcome. The estimated probability for this risk is 0.3.

Another risk we might face is the difficulty and complexity in writing procedural generation algorithms as we have a lack of experience in this area. The estimated probability for this risk is 0.6.

For our risk mitigation plan, we have decided to follow the 'watch and monitor' risk mitigation strategy which involves monitoring the project for risks and consequences and identifying any changes or shortcomings that can affect the impact of the risk. For example, if the teammates that are working on tasks such as developing a procedural generation algorithm run into any problems or are stuck at writing the algorithm, the rest of our teammates will try to step in and help them solve the problem.

Other potential risks might include:

- Bugs in code that are time consuming to solve, estimated probability for this risk is 0.4.
- Scheduling conflicts with teammates, estimated probability for this risk is 0.1.
- Making sure that 3D rendering performance runs at a consistent frame rate, estimated probability for this risk is 0.3.

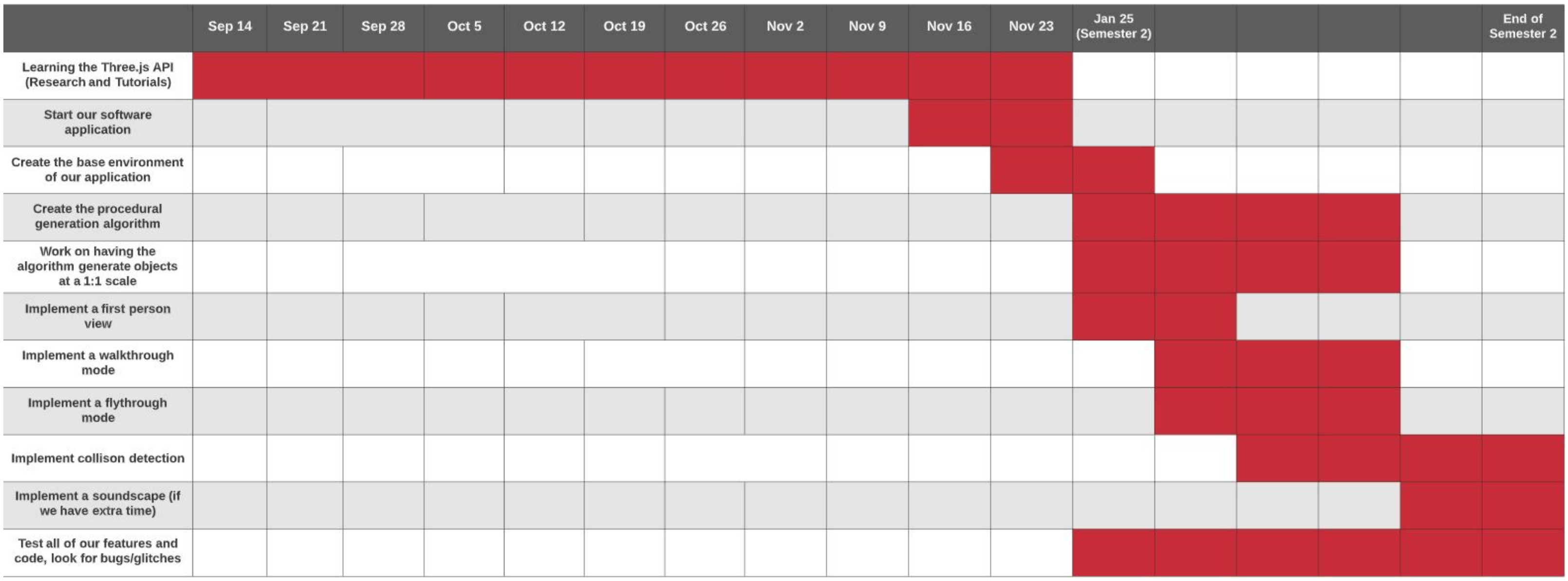
2.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

Our team has decided to set the milestones based on the project schedule. Our first milestone is to have the first chapter of the design document polished by the end of the first week of October 2020 which is week 7. Another milestone is to be done with the second chapter of the design document by week 10, other chapters of the design document subsequently following the project schedule. Our team has planned to be versed in Three.js as well as javascript by the second week of November by watching and doing tutorials. We plan on distributing roles to our team members by the end of the second week of November 2020. Our team plans on starting to learn about procedural generation beginning the first week of November. Another key milestone would be for our team to start implementing our learnings and start working on the prototype of the project beginning the second week of November 2020. Overall we plan on being done with the prototype by the end of December 2020 and the entire project by the end of May 2021.

For the project metrics, the application is going to be a 1:1 scale model of the Earth's environment. The platform will have procedural generation implemented that will generate objects for the environment dynamically using an algorithm which will be developed by our team. The platform will have the option for the user to be able to choose between walkthrough mode as well as flying mode. The application will also be equipped with collision detection that will detect any kind of contact as well as collision between the user and the procedurally generated objects.

For the evaluation criteria, We plan on evaluating the work done by using many types of tests for example; Unit testing, System testing, Stress testing, etc. We plan on measuring the progress of the project by using Trello. Since we haven't begun developing our project platform/application, we have not settled on the final metrics and the test cases yet.

2.4 Project Timeline/Schedule



	Sep 14	Sep 21	Sep 28	Oct 5	Oct 12	Oct 19	Oct 26	Nov 2	Nov 9	Nov 16	Nov 23	Jan 25 (Semester 2)					End of Semester 2
Learning the Three.js API (Research and Tutorials)	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥	🟥						
Start our software application										🟥	🟥						
Create the base environment of our application											🟥	🟥					
Create the procedural generation algorithm												🟥	🟥	🟥	🟥		
Work on having the algorithm generate objects at a 1:1 scale												🟥	🟥	🟥	🟥		
Implement a first person view												🟥	🟥				
Implement a walkthrough mode													🟥	🟥	🟥		
Implement a flythrough mode													🟥	🟥	🟥		
Implement collison detection														🟥	🟥	🟥	🟥
Implement a soundscape (if we have extra time)																🟥	🟥
Test all of our features and code, look for bugs/glitches												🟥	🟥	🟥	🟥	🟥	🟥

Figure 3. Project Timeline Gantt Chart

The following information represents all of the tasks with their corresponding due dates, which is also shown in the Gantt chart above.

- Learning the Three.js API (Research and Tutorials): November 23
- Create and start coding the base of the application: November 16

The rest of the tasks will be started within the first semester but will not be finished until the second semester, and they will all be worked on simultaneously.

- Create the procedural generation algorithm that will generate the world: End of second semester
  - Have the algorithm create all aspects of the world at a 1:1 scale: End of second semester
- Work on a first person view and user movement within the world: End of second semester
  - Create a walkthrough mode and flying mode: End of second semester
- Work on collision detection: End of second semester

We will only end up creating a soundscape for our application if we feel like we have ample time, our client said this is not a full on requirement and to only do it if we have time at the end of the second semester.

- Create a nature themed soundscape for the world: End of second semester

2.5 Project Tracking Procedures

For this part, we plan to use four different types of web-based platforms. They each play an important role in keeping our project on track. They are:

1. Trello board: We use this to distribute tasks and keep track of who is doing what part of the project on a weekly basis.

2. Discord: This is where we do most of our communications, we use this for anything. This includes questions, meeting planning etc.
3. GitLab: This is for storing and maintaining our code for the project. It also helps us keep the code structured and safe. With this, we can also experiment and try new ideas because of the branch feature of GitLab.
4. WebX: This is what we use to have our weekly meetings. On this platform, we have live discussion and assign tasks to group members. Also, we also have discussions with our project advisor and clarify any lingering concerns that discord communication cannot handle.

Together, all of these four platforms keep us organized, on track and synchronize with each other.

2.6 Personnel Effort Requirements

Task	Number of Hours Required	Description
Research APIs/Frameworks needed to design project	~5 hours per person	Every member of the team spent the first week researching the tools necessary to begin work on the project. These topics included a recommended IDE and graphics API.
Complete Three.js tutorials	~8 hrs per week First semester	Each team member spent a minimum of 5 hours a week completing Three.js tutorials. This was intended to introduce the group to the platforms we will be working with, as well as slowly develop an understanding of the steps needed to complete the project.
Create first Three.js instance and render an image on web server	~12 hrs per person 2 weeks	Every member will work on creating a working Three.js program that displays on a web server.
Design procedural generation algorithm	~15 hrs per person 3 people 4 weeks	3 team members work on deriving the procedural generation algorithm that will be used to create and implement textures/patterns

Figure 4. Personnel Effort Requirements

2.7 Other Resource Requirements

- Microsoft Visual Studio
- Three.js API
- Use of a web server
- No financial requirements

3 Design

3.1 Previous Work And Literature

Our project aims to create an explorable, full-scale three-dimensional virtual nature environment with unique features including a 1:1 scale with reality and elements of procedural generation. The use of a procedural generation algorithm to create unique and infinite worlds has been used in many different games. The big game that we looked at was Minecraft [1], which uses procedural generation to create endless unique chunks so that the world is infinite and never repeating. For our project, we plan on using Perlin noise, which is a procedural texture primitive, used to create unique landscapes like hills and mountains. Minecraft also uses this as a way to create various different landscapes and textures within the different biomes.

The main difference between Minecraft and our project is that we plan on creating all the various features, like trees and mountains, at a 1:1 scale with reality. Minecraft does not have this feature since trees are fairly small and even the biggest mountains in the game are only the size of a small hill. In our project we want our game to seem as real as can be, that is why we will also use real looking textures and graphics, unlike Minecraft. Minecraft uses cartoon-like graphics in their game which do not look real in any fashion, while for our project we want every aspect of our game to match up with reality and give the user a sense of nature.

One other game that we did some research on because it has a similar purpose to our game was Firewatch [2], which is a game all about escapism. It's about running away from real life, from it's hardships, from your mistakes. Almost every character in the game has a troubled past they are trying to flee from, mistakes they are trying to avoid facing, all of which are inescapable facts of life. That is a lot like what our game is meant to do, we want our game to be a place for people to escape their current COVID-19 life which is probably full of stress and hardships. The fact that Firewatch has somewhat of the same purpose as our game, we made sure to do some research on this game and see what we could learn from it in order to improve our game. Firewatch is based in a forest which our game will be too, their game is also all in first person which is partly why we chose our game to be all in first person. Also, in Firewatch you can interact with some things in the environment, which is why we are motivated to hopefully have some things in our games environment that the user can interact with. The only major difference between Firewatch and our game is that Firewatch has some spooky vibes to it sometimes which might freak the user out, we want to make sure our game only brings relaxing and happy vibes to the user while they are exploring our games environment. Firewatch sometimes uses silence or weird sounds to bring in these spooky vibes, so we will make sure we don't do anything that could possibly weird the user out while playing our game. Overall, Minecraft and Firewatch have helped us in planning out our game and its features a lot over the course of this semester.

3.2 Design Thinking

We first decided to design our platform using OpenGL and C++, we worked on the tutorials for about 4 weeks and learned different aspects of OpenGL. But later by doing more research and communicating with our client, we found out that using OpenGL to build a web platform wouldn't be an ideal solution. If we used OpenGL we would have to convert the built application into a web app which would take twice as much time. So after researching a lot we decided to switch to use

WebGL and Three.js. We chose Three.js because every web platform that runs on a browser is built on Javascript as browsers only understand Javascript. We also chose Three.js because it would be easily integrated with WebGL. We also found out that Three.js supports procedural generation which was a requirement for our project.

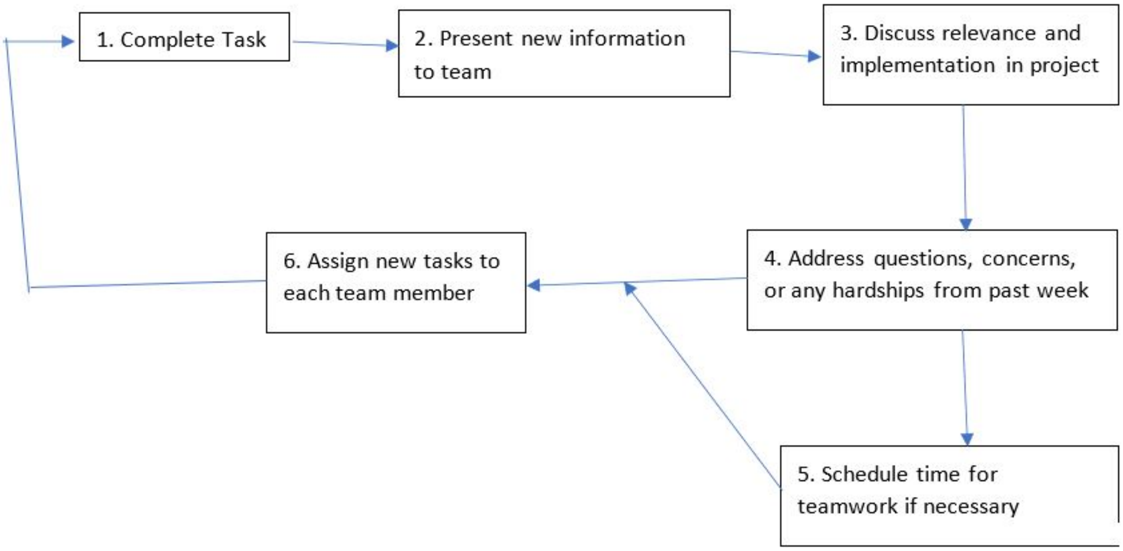

1. Complete Task
2. Present new information to team
3. Discuss relevance and implementation in project
4. Address questions, concerns, or any hardships from past week
5. Schedule time for teamwork if necessary
6. Assign new tasks to each team member

Figure 5. Design Thinking Diagram

3.3 Proposed Design

- We switched what we are going to use for our application. We originally planned to use OpenGL but after some research we decided to use three.js. We started by working through various three.js tutorials.
- The flow of how the client will interact with the application looks like the diagram below:

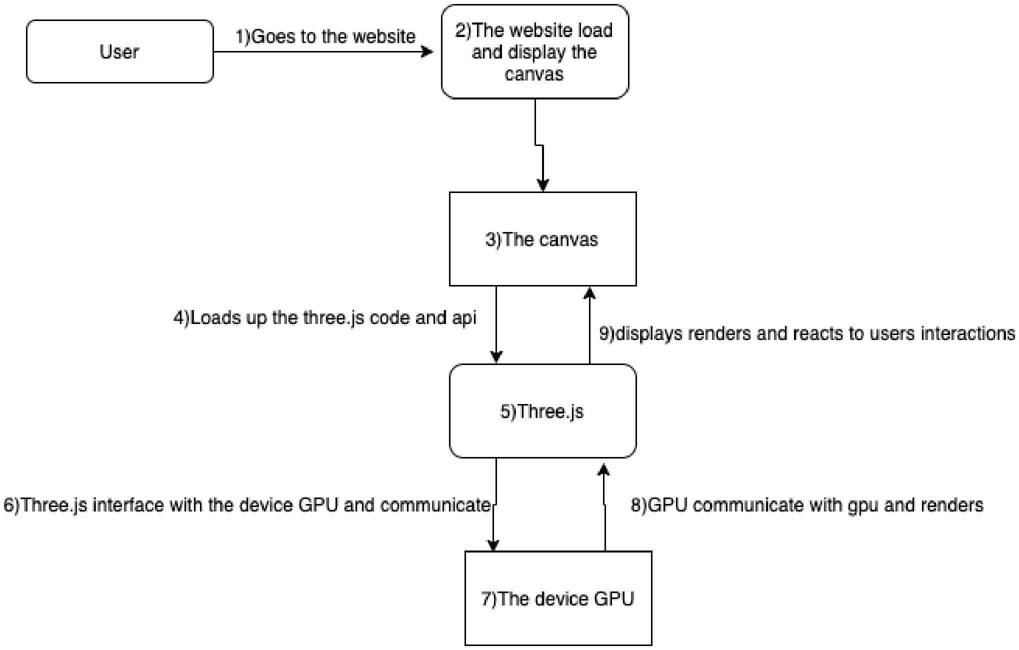

User
1)Goes to the website
2)The website load and display the canvas
3)The canvas
4)Loads up the three.js code and api
9)displays renders and reacts to users interactions
5)Three.js
6)Three.js interface with the device GPU and communicate
8)GPU communicate with gpu and renders
7)The device GPU

Figure 6. Application Flow Diagram

- We had people learn different parts of three.js, splitting up into groups so we can cover more material in a shorter period of time. We have people working on procedural generation of the game development and the creation of the environment of the game.
- Using three.js will satisfy the barebones 3D graphics application and proof-of-concept 3D graphics application requirements.

Test Applications:

Throughout this semester, we spent most of our time working on various Three.js test applications so we could learn everything there is to know about Three.js. We spent most of the semester on this because none of us have any prior experience with 3D graphic design or Three.js, so we needed to make sure we had a good understanding of Three.js before starting our projects applications. We had three people work on learning how to create procedural generation algorithms using Three.js and three people work on the creation of the environment. After splitting up into these focus groups, we one main image from the application some of us worked on to share that can give you somewhat of an idea of what our application will look like.

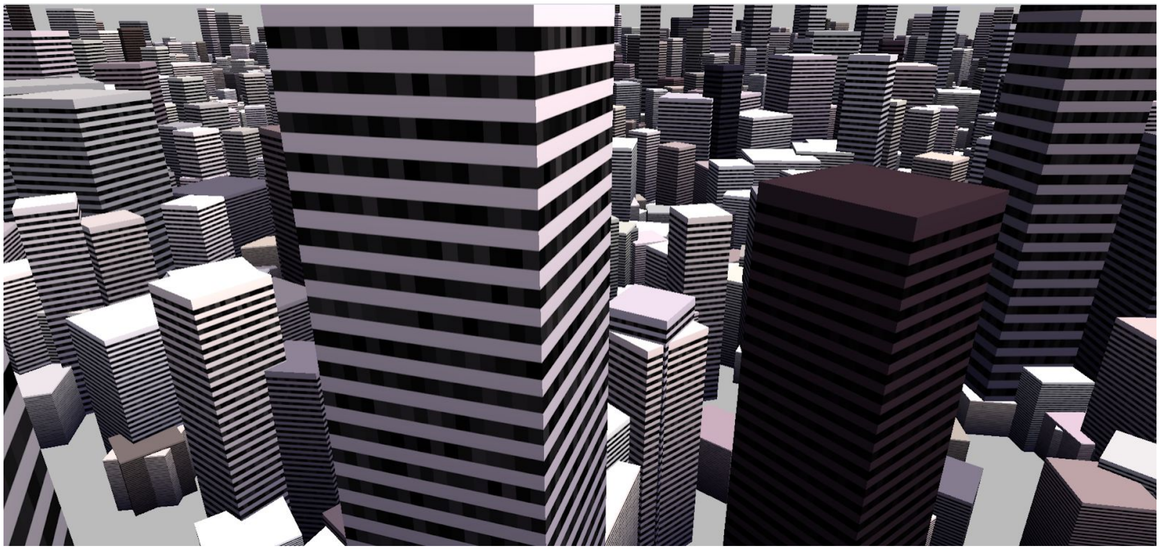


Figure 7. Three.js Test Image

This image shows an infinite environment with skyscrapers of different sizes and colors procedurally generated randomly throughout the environment. Our plan in our project is to have an algorithm like that which will procedurally generate things like trees, bushes, etc. instead of buildings.

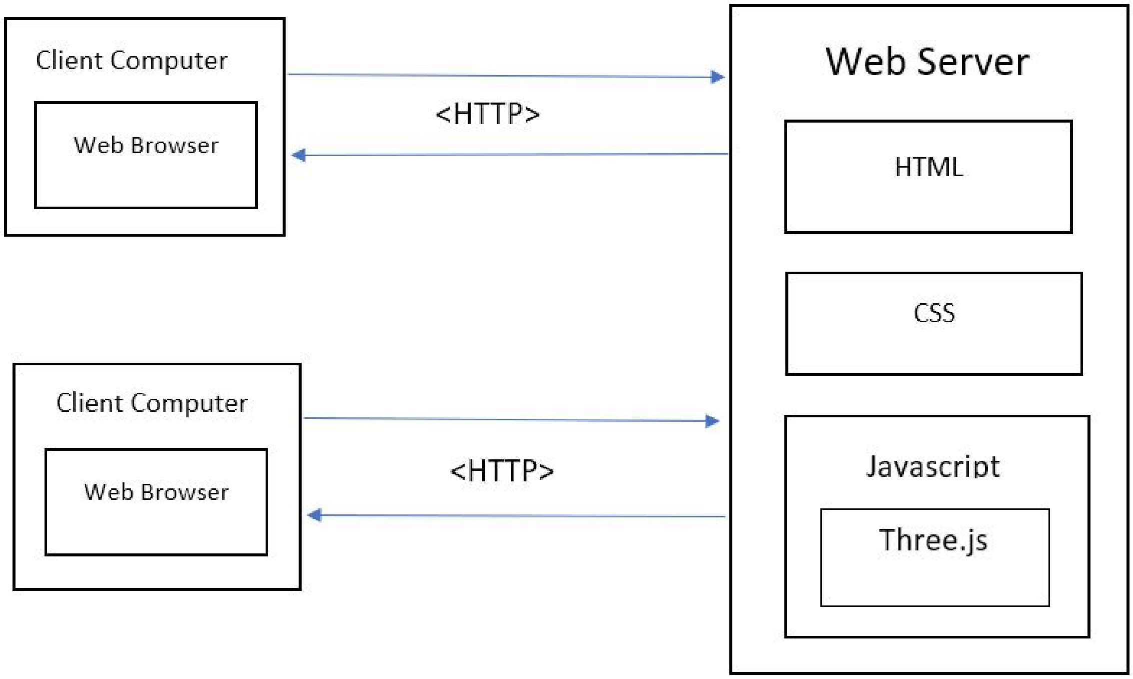


Figure 8. Architecture Diagram

3.4 Technology Considerations

Three.js is a cross-browser JavaScript library and application programming interface used to create and display animated 3D computer graphics in a web browser using WebGL. Three.js has the following strengths and weaknesses.

- Strengths
  - Designed to make working with basic game-quality 3D graphics easier.
  - Gives a very clean, low level access to all the rendering capabilities.
  - Loads instantly and integrates well into existing website code.
  - Easy to work with javascript libraries.
  - Convenience of accessing and working with it through the web browser.
- Weaknesses
  - Lacks in lot of documentation and informative references
  - Since the framework changes frequently, a lot of outdated solutions might be available online.
  - It's more of a renderer than a real engine. Because of this, it will easily reach its limitations in terms of visual processing power with more complex visuals and motions.

We believe that Three.js fits our needs of rendering a procedurally generated interactive 3D environment relatively well. Since our project will not involve too many complex visuals and motions, Three.js should be able to handle rendering a natural environment filled with trees, greenery and mountains. When it comes to outdated solutions found online, we will make sure to look for the latest tutorials that do not use old versions of the library.

3.5 Design Analysis

Our initial design plan worked very well with how we expected it to go. As we started working through various three.js tutorials, splitting up the work into different focus areas helped us learn more and cover more material as a group in a shorter period of time. Since we are on a tight schedule in getting this project done in two semesters, we needed a design plan that helped us learn the three.js API in a shorter time frame, and our design plan does just that. Also, the fact that we chose to use three.js for our project worked out perfectly because it had everything we needed to cover all of our project requirements, as well as the fact it is a cross-browser library so it works on Windows and Mac Operating Systems. We needed to use a cross-browser library because our group is a mix of Windows users and Mac users, so three.js made it easier for us to work together. When it comes to our thoughts on our design plan, we believe that it worked out perfectly in what we needed to accomplish. At first, our plan was to just do random three.js tutorials and learn whatever we could, but we soon realized that that plan would waste a lot of time. That is why we soon after modified our plan so that we all could split up into focus groups and focus on learning specific things with three.js. This design modification helped us learn three.js a lot more efficiently, and prevented us from learning stuff that would not be needed.

3.6 Development Process

Our team decided on using the waterfall process model for the course of the project. We spent the first few weeks gathering information like requirements, work environment, techniques we would need to learn, and what APIs we would need to use. From this, we decided that we would work on learning new material while generating a better understanding for the project with our client. Once we are knowledgeable enough to begin coding, we will have developed a thorough understanding of the scope of the project, specific requirements, dependencies, and a timeline. Due to the nature of the project and the given information we thought it would be best to follow a waterfall model.

3.7 Design Plan

The main use case of our project would be to allow the user to explore the wild in the virtual realm, and all of our models would be centered around and serve this use case. There would be three main models, environment generation, procedure generation of environmental objects and interface to interact with the environment. The environment would need to be generated first, then other objects like trees and chairs would be populated within it and finally, the user would be able to take certain action like sitting down on the randomly generated chair. At the current stage of our project, the actual dependency and coherency of models are still subject to change.

4 Testing

4.1 Unit Testing

As we get into our software project, we will have to do a ton of unit testing on every little thing we code in order to make sure our code works exactly how we expect it to, this will also help prevent bugs when our final project is shared with our client. When it comes to the main modules that we will be testing in isolation, the first thing we will be testing is our games first person movement controls. We will have to test the first person walking controls as well as the first person flying controls because we will be implementing both modes in our game. The next big thing we will have to test extensively is our procedural generation algorithm which is what will be used to fill our world with infinite objects such as hills, trees, bushes, mountains, etc. We need to test this to make sure the algorithm works based on how we expect it to, for example, we want to make sure the world has some sort of organization, we do not want bushes inside of trees or trees inside of other trees because that would look bad and very unrealistic. The next thing we will need to test is collision detection, we need to test this to make sure that while moving around the world, you should not be able to move through things like trees or mountains since they are solid objects. The last big thing we will need to test extensively is the fps of our world since we expect our game to run on 30 fps with no delay or choppiness. We will be testing a lot more things in isolation on top of what was just talked about once we get more into coding our project.

4.2 Interface Testing

When it comes to testing the interfaces of our application, this will include testing the web server interface and the application server interface. We will not be needing a database for our application and project, so we will not need to worry about testing a database interface. Most of the testing we will be doing on these interfaces will be edge case testing, in order to test certain things about each interface that might not normally be tested or come up while playing the game. This testing will help prevent a lot of annoying little bugs within the two interfaces, and thus create an overall better application. We will do interface testing on our application to ensure that end-users do not encounter any problem when playing our game. Also, to check if a solution is capable of handling network failures between an application server and website. Lastly, we want to ensure that errors are handled properly or return an error message for any query made by our application.

4.3 Acceptance Testing

We would have scheduled demonstrations to show the progress and that the functional requirements are being met. For the non-functional requirements, we can write a status report to demonstrate and inform the client that the non-functional requirements are being met.

4.4 Results

We have not started coding our actual application yet due to the fact that we spent this semester planning our project out and learning how to code in JavaScript and how to use the Three.js Graphics API. We plan on starting our actual project over winter break and into next semester, so once we start that we will also start all of our testing. Due to this fact, up to this point we do not have any test results since we have no code to actually test. We expect to see plenty of failures in the integration of our project application, as well as the stress and performance tests that we will run to make sure our application is running well. This is because we need to make sure our final application has no noticeable delay/choppiness while exploring the world, since this is a requirement straight from our client. We will mainly focus on testing the main parts of our code and the main things which we know will cause problems in our code, however, we also know there will be many other unexpected issues within our code that we will need to solve and eventually test for. The main problem that we will be running into throughout next semester while coding our application is getting our application to work and show on a live web server using Three.js because of the fact we still are not experts after only one semester of learning and practicing with the API. We hope to test our application as much and as thoroughly as possible in order to make sure our final product is of high quality for our client.

5 Implementation

By the end of our first semester, our whole group should be pretty comfortable with coding in JavaScript and using the Three.js API. We have our project created and pushed to our repository so that everyone in our group can clone the repository to their local system, this gives everyone the opportunity to start working on our project over winter break if they would like to, however, this is not required. Next semester, all our focus will be on coding the different aspects of our project so that we have it all done by the end of the spring semester. As talked about throughout this design document, our team has split into two groups of three, one group focused on the creation of a procedural generation algorithm and another focused on creating the games environment. We will stay in these focus groups going into next semester when we start coding these aspects of our application. So, the team members that focused on procedural generation throughout the first semester, will be focused on creating the procedural generation algorithm needed for our application next semester. On the other hand, the team members that focused on creating the games environment throughout the first semester, will be focused on creating our games environment needed for our application next semester. We will be doing it this way because creating the environment and the procedural generation algorithm are the two main parts of our application, so once we get those two things done, we will have the bulk of our final application done. After we get those two things finished, as a group we will work on first person walking and flying movement and collision detection. Once those things are done, the only things left are testing the different parts of our app to make sure they work as intended as well as to make sure there is no noticeable delay or choppiness while exploring the world. As touched on in part 2.4 of this document, the project timeline, we would like to have all of the main components of our application stated above done with at least 2-3 three weeks left in the semester so we can go through the final testing phases to make sure the application has no bugs and runs as intended. This plan and schedule is tentative, and can be changed if needed as we progress through next semester, but, our current plan for next semester should be effective in helping us get our projects application done as efficiently as possible.

6 Closing Material

6.1 Conclusion

Thus far, our team has conducted research regarding various graphics APIs. From this we decided which would best fit our project needs. Three.js was the most suited and provided the best compatibility. We then began learning and implementing Three.js on our own in order to prepare for the actual implementation in the project. We will begin development by generating terrain using procedural generation techniques. Our goal is to then create basic objects that mimic an outdoor environment (trees, bushes, water, grass, etc). Once we have a completed environment we plan to incrementally improve the quality of textures, placement, and generation. The most important factor in our process and as a team is accountability. Every team member must do their part in order to successfully complete the project.

6.2 References

[1] "Procedural Generation in Game Development," *davidepesce.com*, 16-Jun-2020. [Online]. Available: https://www.davidepesce.com/2020/02/24/procedural-generation-in-game-development/. [Accessed: 08-Nov-2020].

[2] A. Webster, "Firewatch review: a game that perfectly captures the beauty and terror of nature," *The Verge*, 08-Feb-2016. [Online]. Available: https://www.theverge.com/2016/2/8/10922560/firewatch-review-ps4-pc. [Accessed: 08-Nov-2020].