Take a Virtual Hike

FINAL REPORT

Team 19
Client/Advisor: Mathew 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.
- 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 terrain
  - 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

Engineering Constraints

- Use of THREE.js limits the programming language to JavaScript
- Required to be a web application
- Achieve the Engineering requirement of running on a reasonably priced and attainable computer
- Time constraint: 2 semesters to get project done

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

- Working with Three.JS API
- Working with JavaScript language
- Working with 3D computer graphics
- Creating and working with procedural generation algorithms
- Using perlin noise terrain generation
- 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 aimed 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 included 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, was used, and various memory and CPU optimizations were needed to be employed to achieve satisfactory performance. Programming languages included javascript and html.

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 application we created relies 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 3-D generated 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
- Must include collision detection
- 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 3D virtual world for the purpose of relaxation as well as anyone who wants to explore an environment that is 1:1 scale is a potential user. Our platform is to be used as a form of stress relief tool that helps people escape the "'normal" world and presents them with a virtual world that relaxes their mind.

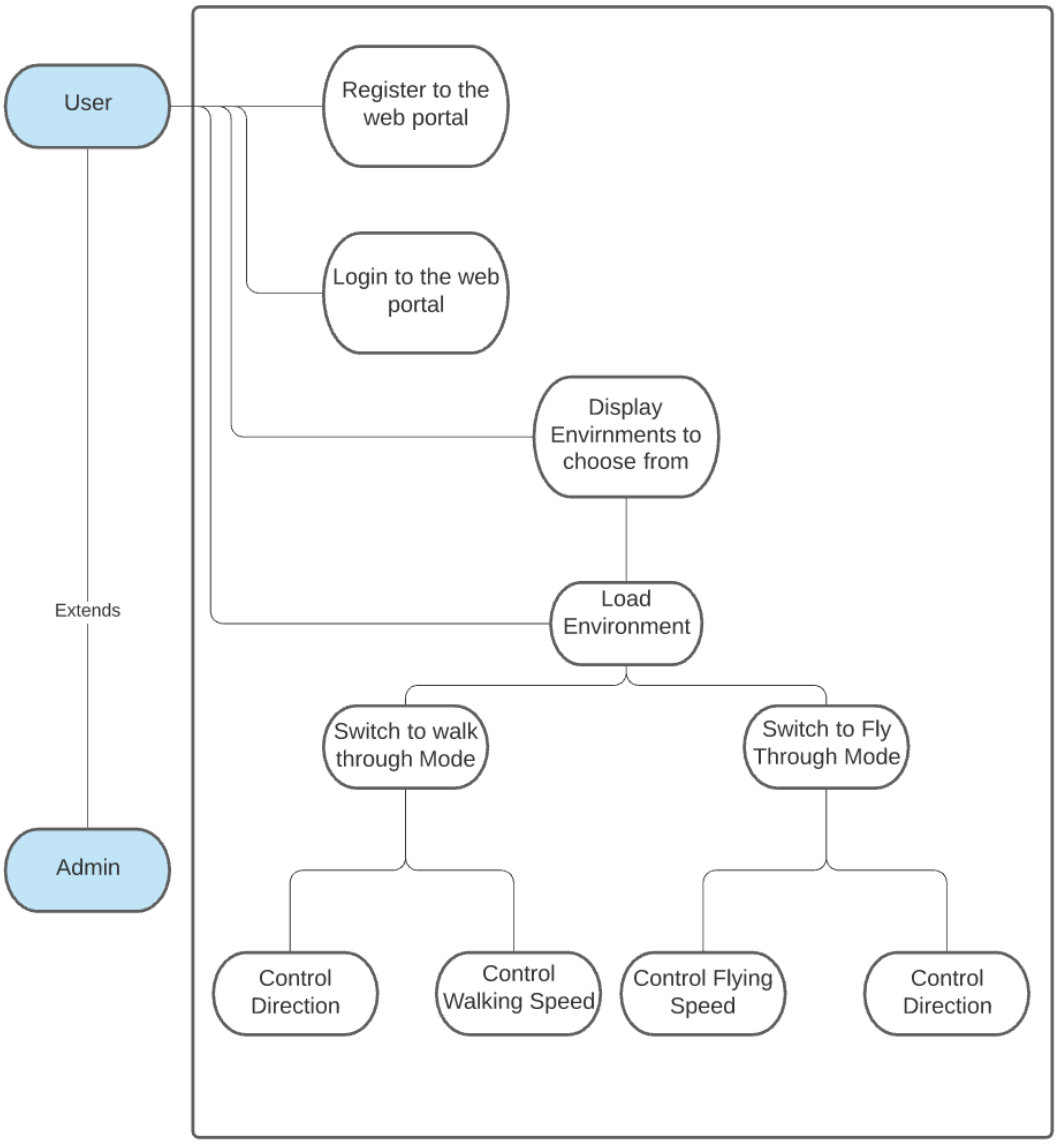

Figure 1. Use-Case Diagram

1.6 Assumptions and Limitations

When it comes to our project's assumptions, we assumed our final project would end up being a web application, so our game is limited to running on a web browser rather than a phone application or something else like that. Our application is a single player game, which means only one player can play on a browser at a time. Also this means we did not implement a multiplayer feature where people on other browsers can join your world. We implemented 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 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 is limited to a first person view, we did not implement anything else, such as a third person view. Also, the main thing is the fact that we only had two semesters to work on our project, which included spending a lot of time learning various things like our three.js 3D graphics API and JavaScript.

1.7 Expected End Product and Deliverables

We are delivering our product as a web application. The final application has a 1:1 scale environment where the objects are created algorithmically using a procedural generation algorithm. The application is accessible to any user using a web portal, such as Google Chrome or Safari. The user has the ability to access the front end of the platform only.

We are also delivering a manual to the client that will elaborate on the technologies used to create our platform. We are also including an instruction section in this manual as well as a walk through of the platform so that the client and the user is familiar with the product. This manual is given at the end of this final report as well, within Appendix I.

2 Project Plan

2.1 Task Decomposition

Finished Tasks:

- Learned Three.js and WebGL (Research and Tutorials)
    - Included learning about procedural generation algorithms
- Created the environment of our application
- Created the procedural generation algorithm that generates objects in the world
    - Creates all aspects of the world at a 1:1 scale
- Created user movement within the world
    - First person view
    - Including a walkthrough mode and flying mode
- Created a collision detection system
- Created a nature themed soundscape
- Tested all of the applications features

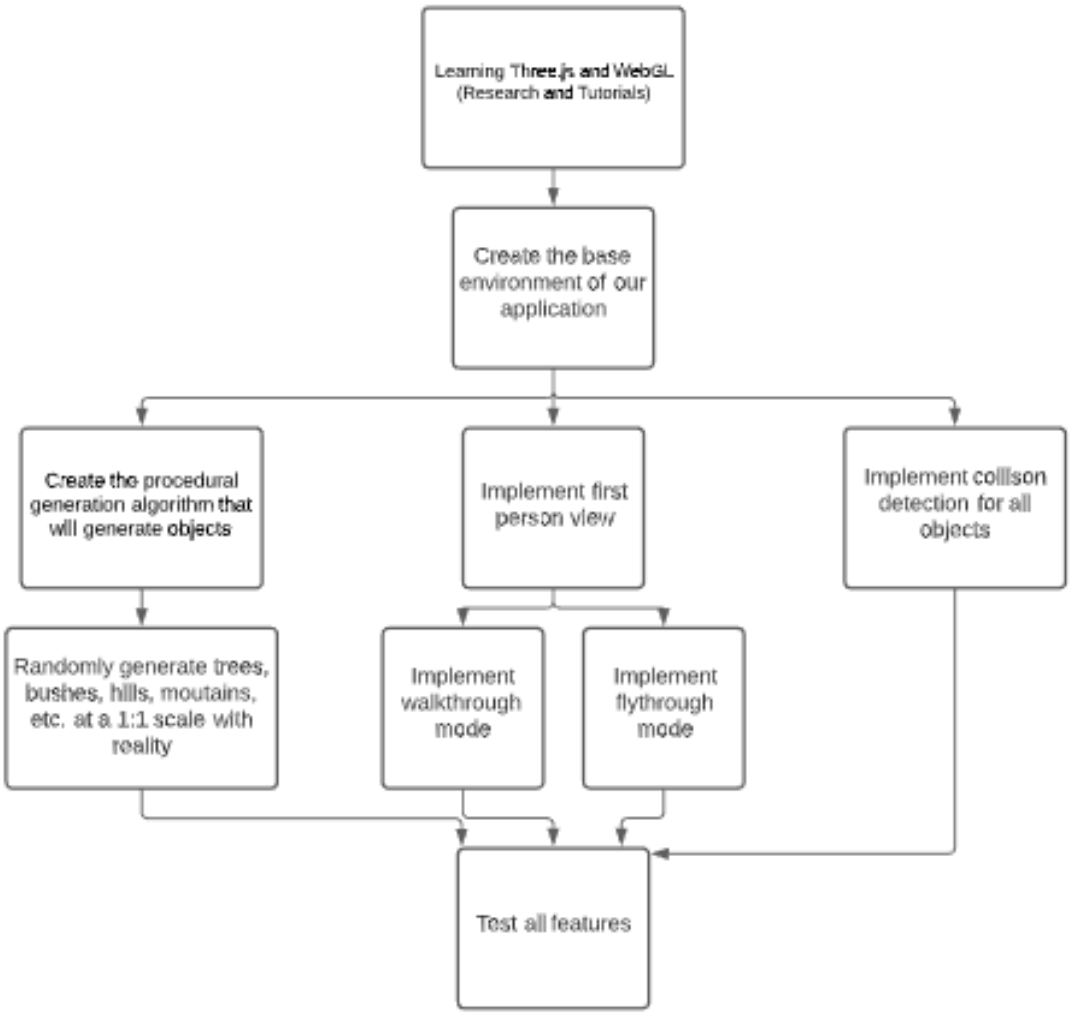

Figure 2. Task Decomposition Diagram

2.2 Risks And Risk Management/Mitigation

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

Another risk we faced is the difficulty and complexity in writing procedural generation algorithms as we also had a lack of experience in that area. The estimated probability for this risk is 0.6.

For our risk mitigation plan, we 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 step in and help them solve their problem. This plan worked out very well

throughout the duration of our project because we all helped each other on our individual tasks numerous times to help solve problems quicker.

Other risks include:

- Bugs in code that were time consuming to solve, estimated probability for this risk is 0.4.
- Scheduling conflicts with teammates, estimated probability for this risk is 0.1.

2.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

The first milestone we got done this semester was to get the unique terrain generated using perlin noise. The next milestone was to create the procedural generation algorithm which randomly generated objects like trees, bushes, rocks, grass, water, etc. After that we got the first-person walk through and fly through modes finished, both of which included collision detection. The last milestone we got done was to create a nature themed soundscape for our environment. After getting all of these main milestones done, we spent the rest of the semester perfecting the environment so it looked realistic and ran as smoothly as possible.

For the project metrics, the application is a 1:1 scale model of the Earth's environment. The application has procedural generation implemented which generates objects for the environment dynamically using an algorithm we created. The application has the option for the user to be able to choose between walkthrough mode as well as flying mode while in the environment. The application is also equipped with collision detection that detects any kind of contact as well as collision between the user and the procedurally generated objects.

For the evaluation criteria, we evaluated the work done by using many types of tests for example; Unit testing, System testing, Stress testing, etc. We measured the progress of the project by using Trello.

2.4 Project Timeline/Schedule

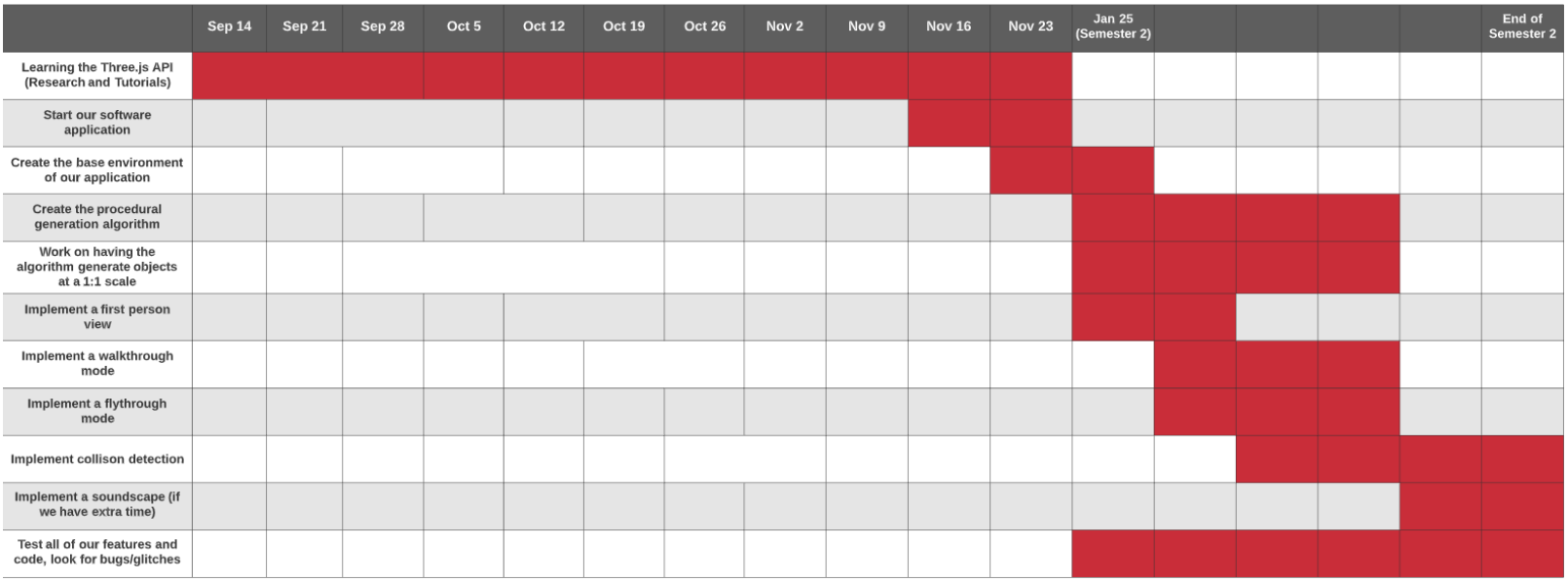

Figure 3. Project Timeline Gantt Chart

2.5 Project Tracking Procedures

For this part, we used four different types of web-based platforms. They each played an important role in keeping our project on track throughout this second semester. They are:

1. Trello board: We used 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 did most of our communication, we used this for anything. This includes questions, meeting planning etc.
3. GitLab: This was used for storing and maintaining our code for the project. It also helped us keep the code structured and safe. With this, we could also experiment and try new ideas because of the branch feature of GitLab.
4. Webex: This is what we used to have our weekly meetings. On this platform, we had live discussions and assigned tasks to group members each week. Also, we had discussions with our project advisor/client and clarified any lingering concerns that discord communication could not handle.

Together, all of these platforms kept us organized, on track and synchronized with each other throughout the duration of the project this past semester.

2.6 Individual Efforts

Name	**Cumulative hours**	**Main task**
Trevor Nemes	75	Procedural generation algorithm
Tyler Hassfield	80	First-person movement
Opeyemi Abass	50	Procedural generation algorithm
Aashu Mallik	72	Water generation algorithm
Akhilesh Ratnakumar	50	Terrain/sky generation
Zian Li	42	Terrain/sky generation

Figure 4. Individual Efforts

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 aimed 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 planned 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 wanted to make sure our game only brings relaxing and happy vibes to the user while they are exploring our games virtual 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 helped us a lot in planning and creating our game and its features 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 3 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.

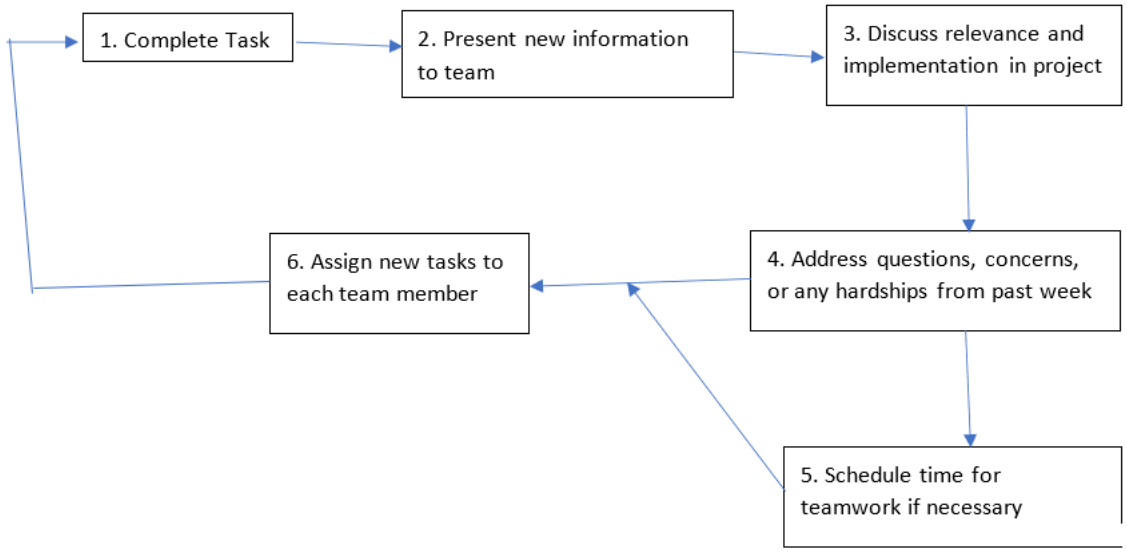

Figure 5. Design Thinking Diagram

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.
- The flow of how the client will interact with the application looks like the diagram below:

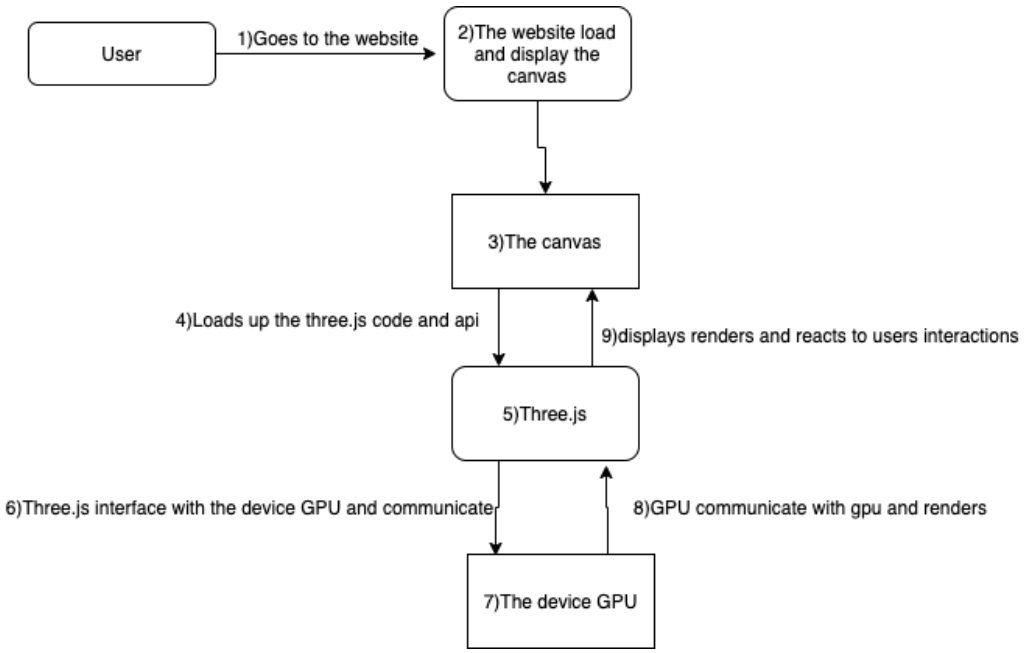

Figure 6. Application Flow Diagram

- 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 algorithms, some working on generating 3D terrain, and others working on first-person movement.

- Using three.js satisfied the barebones 3D graphics application and proof-of-concept 3D graphics application requirements.

Test Applications:

Throughout the first 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 had 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 application. We had two people work on learning how to create procedural generation algorithms using Three.js, two working on first-person movement, and two people work on the creation of the environment. After splitting up into focus groups, we had 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.

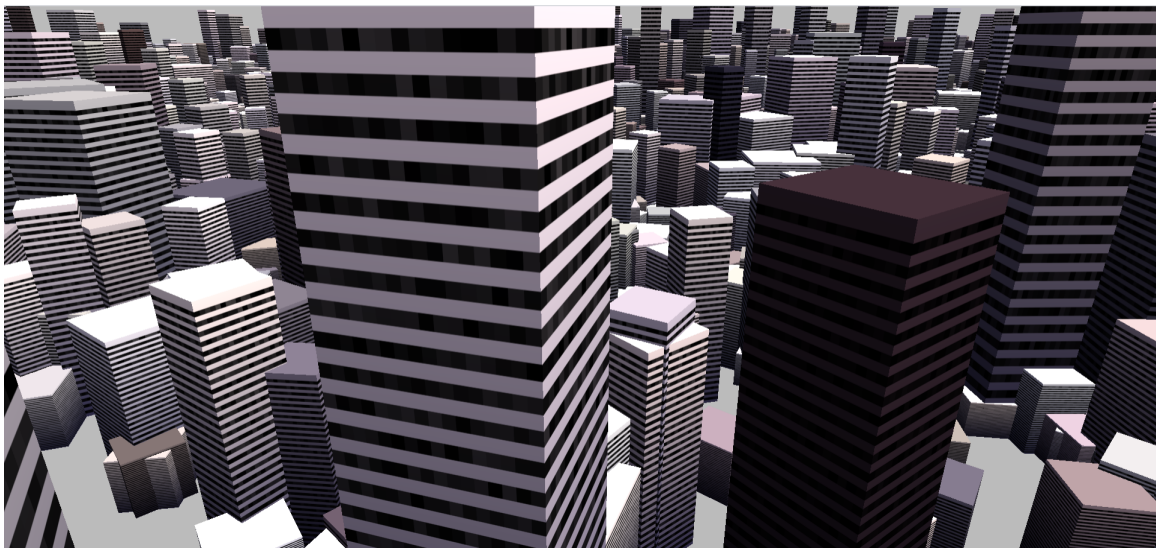

Figure 7. Three.js Test Image

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

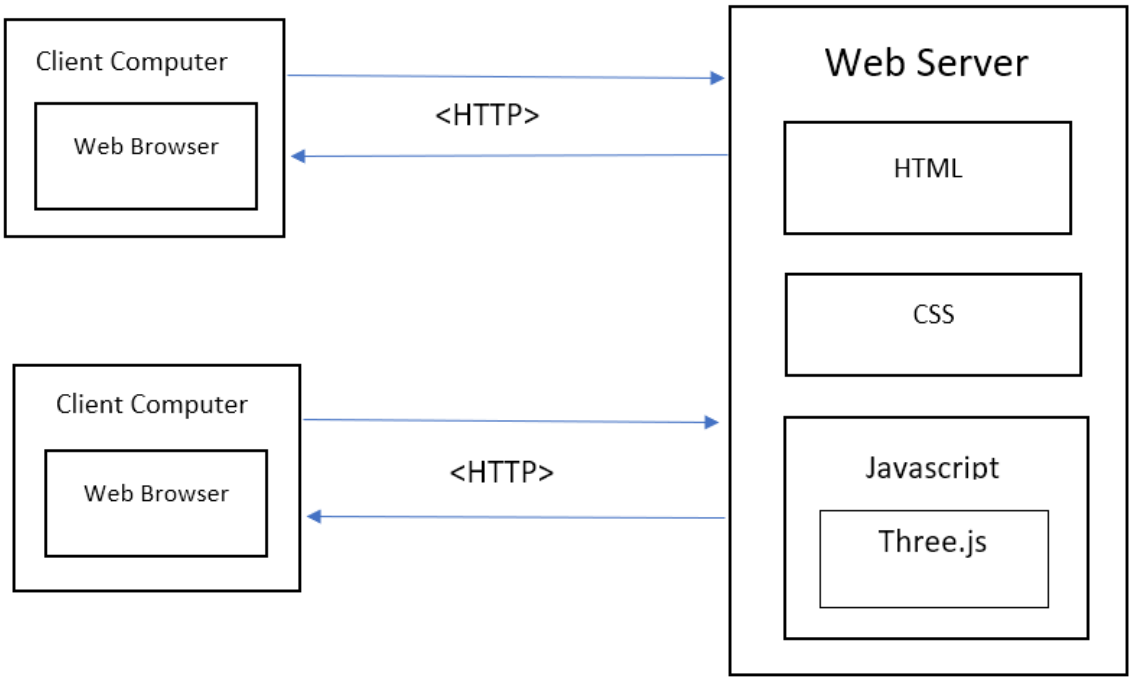

Figure 8. Architecture Diagram

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 believed that Three.js fit 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 came to outdated solutions found online, we made sure to look for the latest tutorials that did 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. When it comes to the second semester of work, we also split up the work in the same way as the first semester. This worked out great because as a team we were able to get all of the major parts of the project done a lot more efficiently, and then move on to perfecting the project and adding bonus features.

3.6 Development Process

Our team decided to use 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. After gaining all of the knowledge needed, we split up the work and started coding our individual parts of the project.

3.7 Design Plan

The main use case of our project is 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 are three main models, terrain generation, procedural generation of objects and an interface to interact with the environment. The environment needed to be generated first, then other objects like trees and bushes would be populated within it.

3.8 Security Concerns and Countermeasures

Since this project is a web-browser based application, there is little to no network security concern. To prevent user data leakage, we would not require a user registration or login, nor do the application store the user activity anywhere in the server. However, due to the size of possible memory allocation from using this application, it is possible to over allocate the memory space and reduce or impact the performance of other running software. To counter this memory allocation problem, the team is still working on optimizing code structure and algorithms.

4 Testing

4.1 Interface Testing

When it comes to testing the interfaces of our application, this included testing the web server interface and the application server interface. We will not be needing a database for our project, so we will not need to worry about testing a database interface. Most of the testing we did on these interfaces was 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 helped prevent a lot of annoying little bugs within the two interfaces, and thus created an overall better final project. We did interface testing on our application to ensure that end-users do not encounter any problems when playing our game. Also, to check if a solution is capable of handling network failures between an application server and website. Lastly, we wanted to ensure that errors are handled properly or return an error message for any query made by our application when running and being used.

4.2 Acceptance Testing

We had scheduled demonstrations with our whole group and faculty/client to show the progress and that the functional requirements are being met. For the non-functional requirements, we wrote a status report every two weeks to demonstrate and inform the client that the non-functional requirements were being met.

4.3 Results

When we first started testing our application, we saw plenty of failures in the integration of our project application, as well as the stress and performance tests that we ran to make sure our application was running well. This was because we needed to make sure our final application has no noticeable delay/choppiness while exploring the world, since this was a requirement straight from our client. We mainly focused on testing the main parts of our code and the main things which we knew would cause the majority of the problems in our code, however, we also knew there would be many other unexpected issues within our code that we would need to solve and eventually test for. The main problem that we ran into throughout the semester while coding our application was getting our application to work and show on a live web server using Three.js because of the fact none of us were experts after only one semester of learning and practicing with the API. We tested our application as much and as thoroughly as possible in the short time we had in order to make sure our final product is of high quality for our client.

One top of all that, the main way we tested throughout this second semester was just by loading up our application and doing various things like walking up and down mountains, running into trees and bushes, jumping, flying, etc. in order to see what errors/failures came while performing these actions. We mainly wanted to make sure we covered all possibilities for the user to make sure that absolutely nothing could go wrong to ruin the users experience and immersion. While performing these tests on the jumping feature, we eventually got it working to perfection to where no matter what we did there were never any weird bugs. Another big thing we tested was the flying feature, which ended up being easier to implement that we expected. After testing this feature, just like jumping, it ran smoothly and perfectly with no known bugs or failures. The one thing we worked on and tested the most throughout the semester was the walking feature. This feature was the hardest to implement due to the fact the terrain is not even, with there being mountains and hills throughout the environment. After a lot of testing, we finally came to the conclusion that the only way to maximize this features performance was to lower the frame rate of our application. This made it so the user could walk while the frame rate gave the code time to accurately update the user's

y-value so that they were always on top of the terrain no matter what. This feature was by far and large the one we spent the most time on, even after all the work and testing we put in, the feature is far from perfect due to the fact it is not as smooth and clean as the flying feature. However, in the end, after all the work and testing we carried out, our final project ended up being a pretty good success, even though it was not perfect.

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.

By the end of the second semester, we had our finished product to share with our client. Our implementation for our project stayed somewhat consistent with our previous semesters implementation in that we split up the work to get various parts done simultaneously. As talked about throughout this document, our client had a lot of set features required for us to implement into our application. Based on each of these features, we assigned one feature to each group member so that we could all work on one main feature simultaneously. The features we split up between members included the terrain generation, tree generation, rock and bushes generation, water generation, sky generation, and first person controls. Our initial plan at the beginning of the semester was to have all of our major parts done around the halfway point of the semester so then we could start putting everything together into one project. In order to do this we used the git branch feature so that we could each create our code within our own branch, and then when we were all done we could start merging our code together into the master branch. When it came to the member working on the terrain generation, they needed to create a function that created a unique terrain that included flatlands, hills, mountains, and textures on each to make it look realistic. For the tree generation feature, we needed to create a function that procedurally generated multiple types of trees with

textures so they looked realistic. For the rock and bushes generation, this member just needed to create a function that procedurally generated bushes and rocks into the environment as well as with textures on them. Next, for the water generation, this member needed to create a function that placed water throughout the environment, only on the lower parts of the terrain. Then, for the sky generation, this member needed to create a realistic looking sky for the environment which included clouds, blue sky, a sun and light source coming from the sun's direction as a way to make shadows in the environment. Lastly, the first person controls were required to have walking and flying mode with collision detection implemented so at least we could walk up and down mountains without falling through the terrain. By the halfway point in the semester, as previously planned, all members got their individual parts done completely. After that, our plan was to start merging all of our code together to eventually have one working project, with the master branch on GitLab. Over the coming weeks we started by merging the first person controls with the terrain generation code. This ended up being the hardest code to merge together which is why we planned to start with it and give us plenty of time to get it done. After that, we spent approximately another three weeks on merging the tree generation, bush/rock generation, sky generation, and water generation with the terrain generation and first person controls code. After finally merging all of our parts together successfully, we then spent the rest of the semester perfecting our application so it ran and looked as good as possible, as well as polishing out the first person walking movement. As I talked about previously in the testing section, we have a really hard time perfecting the first person walking movement in our project because walking throughout a terrain that was not completely flat made it harder to implement without walking through mountains instead of up them. As I said, we spent by far the most time working on the first person walking movement, however, we did manage to finally get it finished to our liking in the final week of the semester. After finally finishing that, our project was completely finished and successful in following the requirements given to us by our client. By following our project plan and implementing it throughout this semester with some minor alterations, we were able to finish this project completely in just one semester.

6 Appendices

6.1 Appendix I - Operation Manual

Access the application by opening a web browser and visiting http://sdmay21-19.sd.ece.iastate.edu/. Once at the webpage, click on the “Take a Virtual Hike” tab to load up the application. Then, select the start button to enter the virtual environment. The application allows the user to move freely around the environment with the following controls:

- Mouse - Look around
- W - move forward
- A - move left
- S - move backwards
- D - move right
- Space bar - Jump

6.2 Appendix II - Alternative Designs

One of our initial goals was to implement a procedural generation algorithm that creates an infinite world. Unfortunately, we were unable to achieve that due to the combination of two factors. One is the team’s lack of experience writing such complex algorithms, and the second is the time constraint of working to implement the many features within a semester. We had another design idea to implement a pathway on

the terrain the user could choose to walk on. However, we decided to abandon that idea as implementing such a small detail turned out to be a lot more tedious, and the user could still walk anywhere on the terrain, so there would be no point in it.

6.3 Appendix III - What We Learned

When we started this project, none of us knew anything about Procedural generation or creating a 3d virtual environment or working with Three.js and Webgl. But as the semesters progressed we started gaining expertise in every aspect of the project either by watching tutorials or by diving into the documentation. Through trial and error, We learned to generate random objects using the Perlin Noise Algorithm, we learned to create and render scalable 3d virtual environments in a web browser using Three.js and Webgl, we learned to implement collision detection engine into our application, we learned to implement soundscape to make the virtual environment seem realistic, we learned to implement first person view controls for the user to navigate the environment, we learned to use Git for version control and most importantly we have learned to work in an agile environment as a team.

7 Closing Material

7.1 Conclusion

We learned a lot working on this project, we were able to achieve most of our goals and have a fully functional version of the game which users can use and experience.
We were able to stay on track, meet all our deadlines and practice effective communication.
The structure we laid out last semester was effective in keeping us on track and productive.
We were able to finish 90% of what we set out to achieve for the project. All but one of the core goals for the project was achieved and the evidence and proof can be found on our team website.

We have a working model of the game, in which users can walk and explore the environment. The project can run on all platforms as we planned at the beginning of the project. We were also able to achieve full immersion for the user, by making the user experience the environment in first person view. We also made the game user friendly by making the user experiences intuitive and also displaying useful instructions and commands in the environment.

Our development process was functional and useful, we achieved this by using GitLab for everything that has to do with the game code, discord for communication and reaching out for help, trello board for assigning tasks to teammates and webX for our weekly meetings and making sure everyone is on track.

We also learned a lot and grew as an engineer in our respective fields. We gained a significant amount of experience in working in a team, how to contribute to a project, being responsible, holding each other responsible, how to communicate effectively, how to ask questions, proposed solutions to problems and how to help teammates in the most effective way.
We also learned how to deal with not meeting goals, we were not able to achieve one of our core goals, which was implementing procedural generation algorithms for the game environment.
Even without this, we were able to make the project work without any damage to user experience when playing the game.

This was a great learning experience that we are all glad we did and because of, we are better prepared for the real world.

7.2 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].