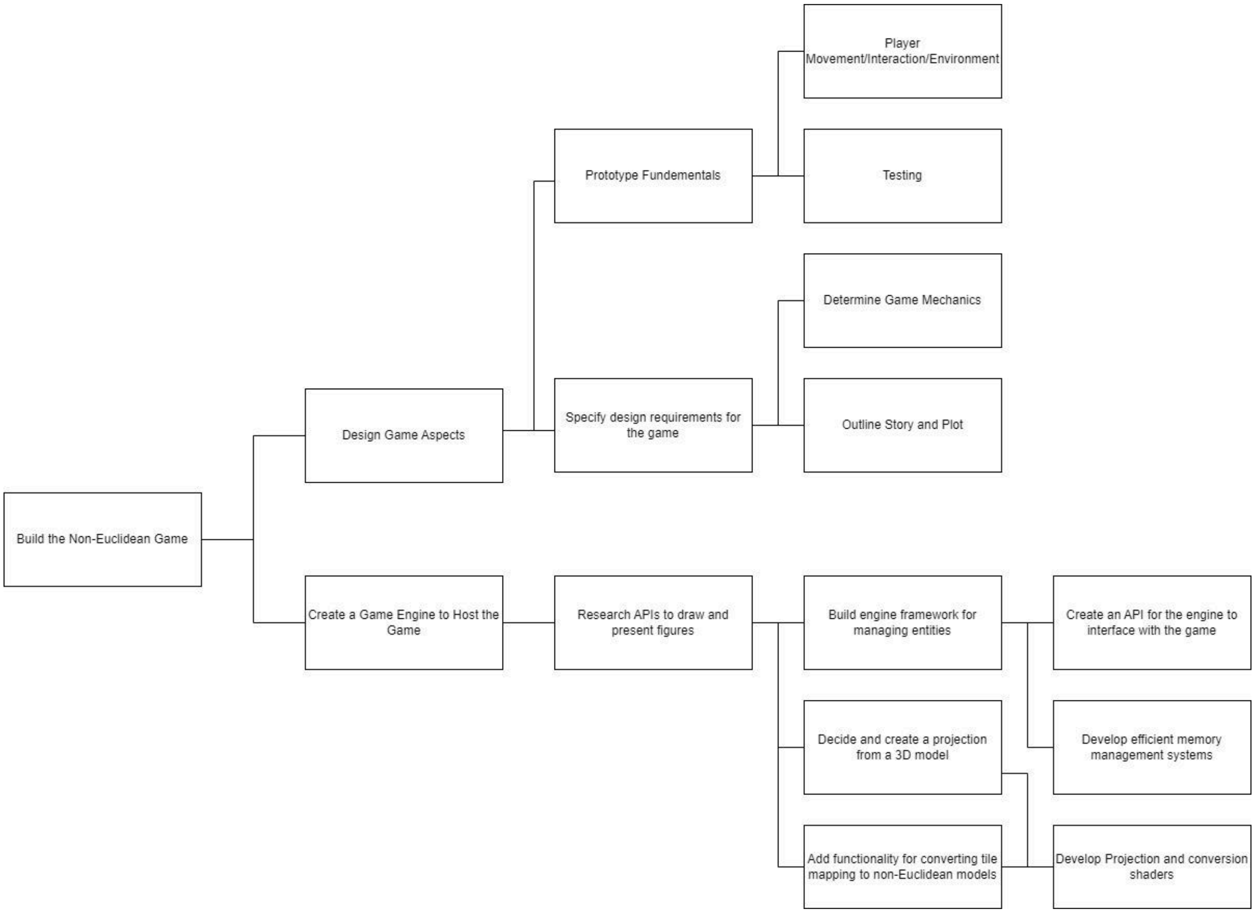
3 Project Plan

3.1 Project Management/Tracking Procedures

We are adopting a hybrid of both waterfall and agile approaches. The overarching method is a waterfall approach for the high-level structure of the timeline and task composition but the specific task completion will use an agile approach. The main reason this structure was chosen is because of the time constraint as this was for a senior design class. We only have 2 semesters of time to have some working final deliverables. Because of this constraint, we need a more rigid schedule to ensure we keep up with timelines to meet this deadline. This rigid schedule tends to fall under a waterfall approach. Another reason we chose this is the dependency of tasks. Specific tasks depend on the previous task being completed, so we cannot take a more agile approach and iteratively improve if we don't have anything to improve upon. Once we have a semi-working video game prototype, we can take a more iterative approach, tasks, and problems. This means that over the course of the year, we will shift from a heavy focus on waterfall to a more focus on the agile approach. Finally, we chose an agile approach to completing tasks because it allows us to make the product we want. As a student-proposed project, our client is our student, which gives us regular client feedback and allows us to change the project's direction based on what we, as a group, find more interesting to do. We are more flexible with the requirements and can make decisions as a group.

We are currently using GitHub and GitHub issues to list issues we have while currently on a task and are using it as a tool to assign smaller tasks to members. Our significant tasks and schedules are maintained in our shared drive folder in Google Sheets. We keep the significant tasks in the sheets because these tasks and deadlines are hopefully not going to change often. We keep track of smaller tasks on GitHub because it is easier to update deadlines and issues we are having when taking more iterative approaches to completing them. We use our weekly meetings to help keep track of deadlines approaching and if we are on track to meet deadlines.

3.2 Task Decomposition


Build the Non-Euclidean Game
Design Game Aspects
Create a Game Engine to Host the Game
Prototype Fundementals
Specify design requirements for the game
Player Movement/Interaction/Environment
Testing
Determine Game Mechanics
Outline Story and Plot
Research APIs to draw and present figures
Build engine framework for managing entities
Decide and create a projection from a 3D model
Add functionality for converting tile mapping to non-Euclidean models
Create an API for the engine to interface with the game
Develop efficient memory management systems
Develop Projection and conversion shaders

The overall decomposition in tasks involve decomposing the high-level tasks between the Rendering Engine and Game Design Teams. Each of the teams will be addressing the appropriate tasks to complete the engine and design requirements respectively, along with ensuring communication between teams will be performed to allow for newly defined design requirements to be met by the rendering team.

The game design team will be performing the top branch of the task decomposition tree while the engine team will be operating on the tasks present in the bottom branch. The game design tasks (prototype fundamentals and specifying design requirements) will be discussed and worked on in parallel, with the design requirements communicated to the engine team for planning purposes. Additionally, the prototypes will act as proof of concepts for the game itself prior to implementation over the engine.

The tasks assigned to the engine team heavily involve creating an engine that can support the specific requirements of the design team, not including additional or redundant features, similar to those found in industry standard engines such as Unity or Unreal. The underlying tasks are the general implementation of the engine, while keeping requirements in mind for creation of a straightforward and useful API that can be used during the game development phase.

3.3 Project Proposed Milestones, Metrics, and Evaluation Criteria

Game Design Team Milestones:

- Game Selection

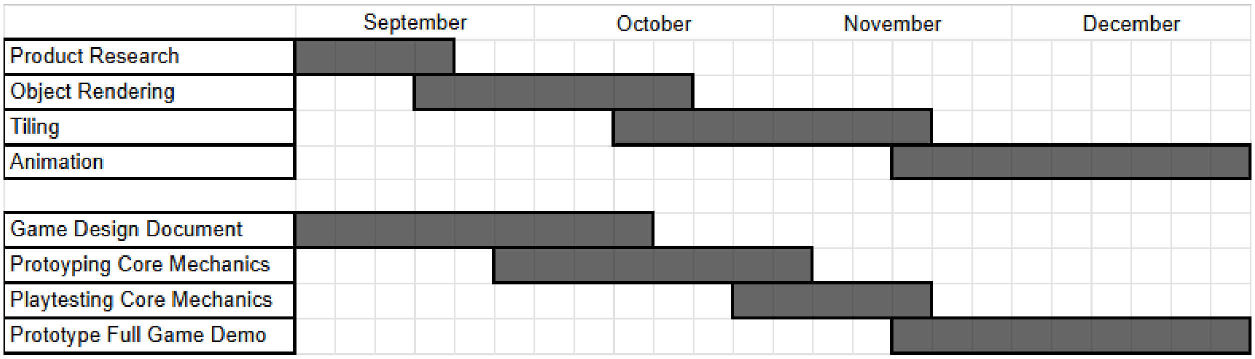
    - Choosing genre and overarching theme of the final game design.
- Design Document
    - Once a game has been chosen, the next milestone would be creating an in-depth document detailing the main features of the game; i.e., what core features we want to have.
    - Main NPCs, Biomes, Genre.
    - Develop a larger story matching the genre and theme.
- Main Prototypes Creation
    - Implement the main features, and all the core features needed to make a minimal viable product demo.
- Single Scene Creation
    - Integrating the core features into a singular scene.
- Functional Demo
    - Getting a demo of multiple scenes, demonstrating the main gameplay features and gameplay loop.

Rendering Engine Team Milestones:

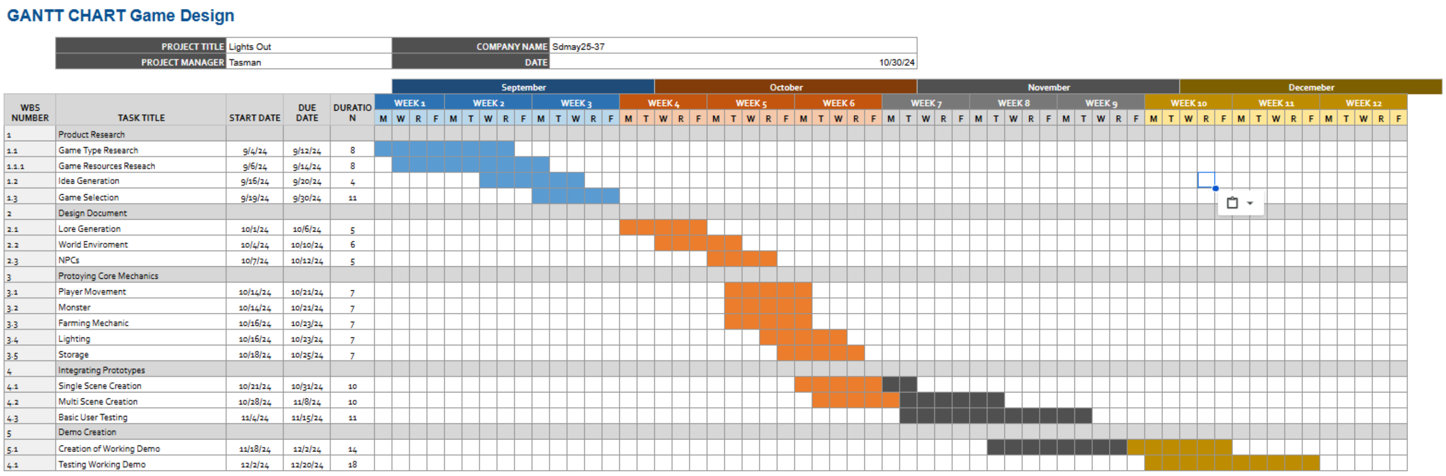
- Math Determination
    - We need to determine which form of non-euclidean math we are going to implement. This milestone is being able to render basic sprites, shapes, and features in OpenGL.
- Basic Rendering
    - This milestone is being able to render basic sprites, shapes, and features in OpenGL.
- Core Feature Implementation
    - Implement the following core features that are needed in the game link:
        - Sprites
        - Entities
        - Lighting
        - Collision
- Math Implementation
    - Ability to render the above in a non-Euclidean manner.
- Running Video Game on Engine
    - Ability to run all of the video game scenes created by the game design team

3.4 Project Timeline/Schedule

Overall Gantt Chart:



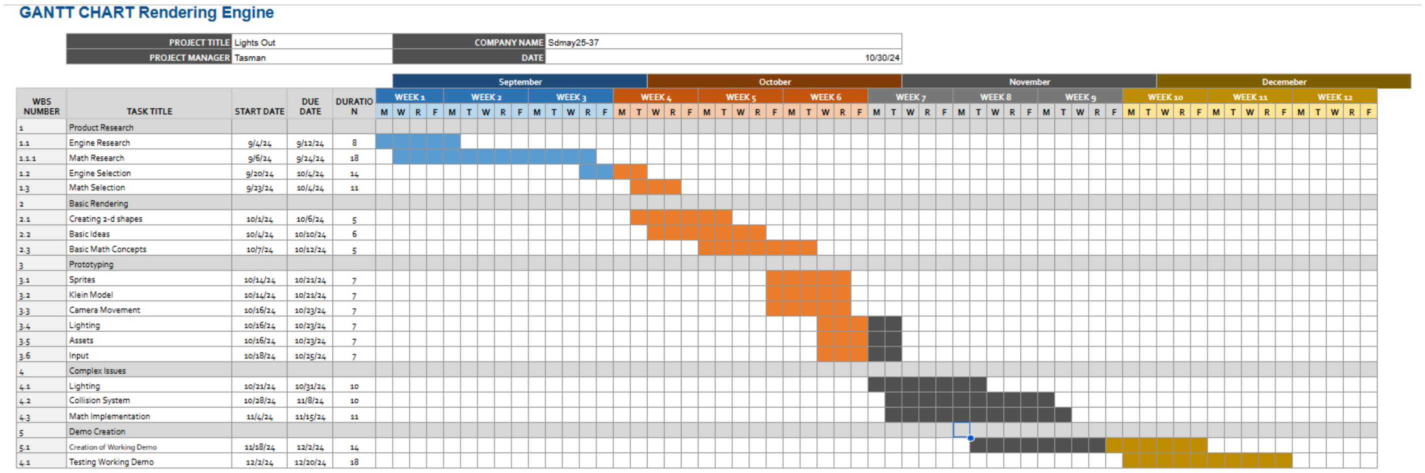
Specific Gantt charts:



GANTT CHART Game Design

PROJECT TITLE	Lights Out	COMPANY NAME	Sdmay25-37
PROJECT MANAGER	Tasman	DATE	10/30/24

WBS NUMBER	TASK TITLE	START DATE	DUE DATE	DURATION
1	Product Research			
1.1	Game Type Research	9/4/24	9/12/24	8
1.1.1	Game Resources Reseach	9/6/24	9/14/24	8
1.2	Idea Generation	9/16/24	9/20/24	4
1.3	Game Selection	9/19/24	9/30/24	11
2	Design Document			
2.1	Lore Generation	10/1/24	10/6/24	5
2.2	World Enviroment	10/4/24	10/10/24	6
2.3	NPCs	10/7/24	10/12/24	5
3	Protoying Core Mechanics			
3.1	Player Movement	10/14/24	10/21/24	7
3.2	Monster	10/14/24	10/21/24	7
3.3	Farming Mechanic	10/16/24	10/23/24	7
3.4	Lighting	10/16/24	10/23/24	7
3.5	Storage	10/18/24	10/25/24	7
4	Integrating Prototypes			
4.1	Single Scene Creation	10/21/24	10/31/24	10
4.2	Multi Scene Creation	10/28/24	11/8/24	10
4.3	Basic User Testing	11/4/24	11/15/24	11
5	Demo Creation			
5.1	Creation of Working Demo	11/18/24	12/2/24	14
4.1	Testing Working Demo	12/2/24	12/20/24	18



GANTT CHART Rendering Engine

PROJECT TITLE	Lights Out	COMPANY NAME	Sdmay25-37
PROJECT MANAGER	Tasman	DATE	10/30/24

WBS NUMBER	TASK TITLE	START DATE	DUE DATE	DURATION
1	Product Research			
1.1	Engine Research	9/4/24	9/12/24	8
1.1.1	Math Research	9/6/24	9/24/24	18
1.2	Engine Selection	9/20/24	10/4/24	14
1.3	Math Selection	9/23/24	10/4/24	11
2	Basic Rendering			
2.1	Creating 2-d shapes	10/1/24	10/6/24	5
2.2	Basic Ideas	10/4/24	10/10/24	6
2.3	Basic Math Concepts	10/7/24	10/12/24	5
3	Prototyping			
3.1	Sprites	10/14/24	10/21/24	7
3.2	Klein Model	10/14/24	10/21/24	7
3.3	Camera Movement	10/16/24	10/23/24	7
3.4	Lighting	10/16/24	10/23/24	7
3.5	Assets	10/16/24	10/23/24	7
3.6	Input	10/18/24	10/25/24	7
4	Complex Issues			
4.1	Lighting	10/21/24	10/31/24	10
4.2	Collision System	10/28/24	11/8/24	10
4.3	Math Implementation	11/4/24	11/15/24	11
5	Demo Creation			
5.1	Creation of Working Demo	11/18/24	12/2/24	14
4.1	Testing Working Demo	12/2/24	12/20/24	18

By the end of this semester, our deliverables will be a rendering demo and a video game demo. These deliverables will not be completed until the end of the semester as a large amount of build-up work must be done. By the end of the year, our final deliverables will be a working rendering engine that is supporting a video game. Because the video game deliverable depends on the working rendering engine deliverable, the rendering engine should be mostly completed before the video game is completed.

These gantt charts show the proposed schedule for our project this semester. It is broken down into two main groups we are separated into, Game Design and Rendering Engine. While we are still communicating between groups about functionality and specifications, the tasks are split as the game design team will do different tasks than the rendering engine team. Both teams start with doing project research. The rendering engine will spend more time on the non-euclidean math in this part of the project because they must use it more. The game design team will focus more on researching what makes a video game suitable to create a good solution. Both teams have a part of their schedule with learning the software due to most members not having any previous experience with these software technologies. Once each team has a reasonable amount of time learning the software, we start implementing the core functionality of the game/engine. We call these core functionality prototypes because they are separated and hopefully can be tested independently. Once we have created these base prototypes, we want to combine them to make an initial working prototype of what the game will look like / how the rendering engine will operate.

3.5 Risks And Risk Management/Mitigation

Game Design Risks:

Task	Risk	Probability	Reason
Product Research	Misidentify user needs	0.1	Most of us are part of the user group and know our own needs
Game Type Research	Lack of Quality Research	0.1	Most of us are part of the user group and know our own needs
Game Resources Research	Using not state of the art tools	0	Have people on the team who know what the start-of-the-art tools are
Idea Generation	Lack of Innovation Game Choice	0.6	See Mitigation
Game Selection	Choosing a game with too big of a scope	0.7	See Mitigation
Design Document	Bad design, so harder to build later	0.4	We are passionate about this project, so the effort will be put into place to make sure it is good
Lore Generation	Uninteresting/too complex for the user	0.3	Want the Game to be interesting, can take inspiration from other games
World Environment	Boring Gameplay	0.4	
NPCs	Lack of player immersion	0.4	Taking Inspiration from other games and seeing how people react to other games allow for better decision-making
Prototyping Core Mechanics	The game will not work properly		
Player Movement	The player will not be able to move	0.1	Basic Feature,
Monster	Inconsistent Monster Behavior	0.2	Feature already implemented
Farming Mechanic	The game will to tedious	0.2	Experience from gaming taught us what this should look like
Lighting	Lose Player Interest	0.4	More critical because it is a core mechanic of the game

Collision	Interactions between objects won't work	0.2	It is important but left to the rendering engine to figure out
Storage	Boring Gameplay cycle	0.1	Prior Experience will dictate how we design this
Integrating Prototypes	Buggy Experience		
Single Scene Creation	Prototypes won't work with each other leading to a delay	0.4	It is crucial to be able to demo/ explain to an outside person. Integrating multiple people's work is always a challenge
Multi Scene Creation	Player Information will not carry over between scenes	0.4	If we can create one scene, making more scenes is not as difficult. Have members with experience
Basic User Testing	The game does not feel enjoyable to play	0.4	Testing to make sure the game feels smooth. None of us are experts, so that bugs will happen
Demo Creation	Delays in previous steps may lead to a lack of time		
Creation of Working Demo	Too big of scope of the project, cannot fit everything we wanted to lead to not being done	0.7	See Mitigation
Testing Working Demo	The game does not work	0.4	Testing to make sure the game feels smooth. None of us are experts, so that bugs will happen

Game Design Risk Mitigation:

- **Idea Generation and Game Selection:**

  This is something that is high risk for us because it is an essential aspect of our project. Our primary user need is enjoyment, so the game idea needs to be a well-polished one that can bring high player immersion. This is a risk due to our group's limited experience. We have some players with game design experience, but we have limited experience in the entire cycle, especially the game idea generation.

  We are mitigating this risk by doing extensive research on other games and taking inspiration from those games. This way, we are not generating this entirely new game but a new spin on a type of game we like. This leads to risk mitigation because we have proof that this type of game can have high player immersion if done correctly.

- **Creation of Working Demo:**

  This will be a high-risk task for this project for many reasons. The first reason is that we may get behind on our schedule and may not be able to get the proper amount of time for this task this semester. Another reason is that tasks require all of our previous tasks to work correctly and be able to work together. We have integration tasks before this, but it still will be an issue for this task.

  The main way we are going to mitigate this risk is by keeping a hard internal deadline for our smaller tasks. We need to keep ourselves on track with the schedule we have to have the time we allocate for this task to be there. If this is not possible, we will move some tasks that are more auxiliary to the second semester to maintain the proper amount of time for this task. We also have a list of resources we can use to help us complete this task on time and give us advice on how to go about this task.

Rendering Engine Risks:

Task	Risk	Probability	Reason
Product Research	Misidentify user needs	0.1	Most of us are part of the user group and know our own needs
Engine Research	Lack of Quality Research	0.2	There is a limited number of rendering codes, so easy to make sure we cover everything
Math Research	Don’t understand the math	0.4	This is complex math, so time is needed to process, understand and then implement
Engine Selection	Choosing an engine that has a high learning curve adds delay to the creation of features	0.1	Already Chosen before the project started
Math Selection	Certain non-euclidean spaces are more complex, so math gets more complicated and worse performance	0.2	Limiting number, choose one that would be relatively simple to implement
Basic Rendering	Don't properly learn to render, so features take longer	0.6	See Mitigation
Creating 2-d shapes	Don't properly learn to render, so features take longer	0.3	There are plentiful tutorials, so it should not be a major issue
Basic Ideas	Don't properly learn to render, so features take longer	0.3	There are plentiful tutorials, so it should not be a major issue
Basic Math Concepts	Don't understand the math	0.5	See mitigation
Prototyping	Delays core development if stuck on this for too long		
Sprites	bad rendering and bloated assets	0.2	Bloated assets are not a worry until the optimization of the engine,
Klein Model	The engine won't meet technical specs	0.9	See mitigation
Camera Movement	Jagged Movement makes gameplay less fun	0.4	Tutorials exist but might be an issue given the different environments we are building
Lighting	Lose Player Interest	0.4	A core feature for Game Design needs to work well
Assets	Bad code management and poor optimization	0.3	Bloated assets are not a worry until the optimization of the engine,
Input	Cannot integrate well with game	0.2	I/O is well documented and can be

	design		tested easily
Complex Issues			
Lighting	Loss of performance	0.2	A core feature for Game Design needs to work well
Collision System	Will not be able to implement game design features	0.4	See mitigation
Math Implementation	The engine will not meet technical specifications	0.9	See mitigation
Demo Creation	Delays in previous steps may lead to a lack of time		
Creation of Working Demo	Too big of the scope of the project, cannot fit everything we wanted to lead to not being done	0.5	It's always a worry if we try to do too much, then we won't have anything to demo this semester.
Testing Working Demo	The engine does not work	0.5	This is new for all of us, we do not know what to expect when it comes to testing/verification of correctness.

Rendering Engine Risk Mitigations:

- **Math**

  Our main risk for the rendering engine is being able to render non-euclidean math. None of us are math majors. The computations to convert from Euclidean space into a non-euclidean space are not trivial. This project is not just learning how to render but how to render in a new space. The main risk is we are not doing the correct computations/conversions because we do not understand the math properly enough.

  Our primary risk mitigation strategy will be working together and sharing our knowledge together. We are all independently trying to learn this math, then coming together as a group and comparing and contrasting what we learned. By having multiple people learn from each other, we mitigate the risk that we are learning incorrect information. We also have a list of resources that we can use in case we get stuck and need advice on how to better our understanding.

3.6 Personnel Effort Requirements

Include a detailed estimate in the form of a table accompanied by a textual reference and explanation. This estimate shall be done on a task-by-task basis and should be the projected effort in total number of person-hours required to perform the task.

Game Design Hours:

Task	Estimated Person Hours	Reason
Product Research	100	Want to spend 2-2.5 Weeks coming up with a good idea

Game Type Research	20	Need to figure out what people like/dislike. Each member puts in 5 hours so all have an idea
Game Resources Research	15	We need to know what resources we have, but this can be done relatively quickly
Idea Generation	40	There should be four people = ~1 week time to create a good initial plan
Game Selection	25	~ Another 6 hours to flesh out ideas and make sure solid idea
Design Document	85	The game needs to be interesting, so we need to spend another 2 weeks making sure we have a good game
Lore Generation	25	Lore is important to keep player investment
World Environment	30	A significant part of game design is exploration, needs an interesting world
NPCs	30	Are an essential factor as to why certain games are loved/hated, so we want to create good NPCs
Prototyping Core Mechanics		Each Core mechanic is being given one person 1.5 weeks to Implement and Test
Player Movement	15	1.5-2 weeks worth of work for one person
Monster	15	1.5-2 weeks worth of work for one person
Farming Mechanic	15	1.5-2 weeks worth of work for one person
Lighting	15	1.5-2 weeks worth of work for one person
Collision	15	1.5-2 weeks worth of work for one person
Storage	15	1.5-2 weeks worth of work for one person
Integrating Prototypes	200	Very important, should be spending a month on making a solid game
Single Scene Creation	40	~ 1 week to integrate core mechanics
Multi Scene Creation	80	~ 2 weeks to create more scenes and add more mechanics. Allows for documentation and preparation for the second semester
Basic User Testing	80	~ 2 Weeks. There will be bugs, want time to test/ change implementation based on feedback
Demo Creation	150	Want to make some polished demos to make a good product Also, building some extra time in case we get behind schedule ~3 weeks

Creation of Working Demo	80	~2 weeks to get a polished demo
Testing Working Demo	70	Want to make a smooth game

Rendering Engine Hours:

Task	Estimated Person Hours	Reason
Product Research	70	~ 2 weeks to get initial research done
Engine Research	20	Want to do it right the first time
Math Research	30	We need to spend more time understanding the complexities
Engine Selection	5	Each member one one-hour meeting
Math Selection	5	Each member one one-hour meeting
Basic Rendering	80	Coupled with research = ~ 1 month time, so we have a good base
Creating 2-d shapes	25	Needed so we have baseline information
Basic Ideas	30	Time to explore OpenGL and get used to the software
Basic Math Concepts	25	Math is hard
Prototyping		Giving Member 1 week to complete each prototype
Sprites	10	1 week time
Klein Model	40	1 month because, most important, he needs to be working extremely week
Camera Movement	10	1 week time
Lighting	10	1 week time
Assets	10	1 week time
Input	10	1 week time
Complex Issues		These tasks may take longer than a week to complete because of their complexities
Lighting	20	This may take more time to implement in a non-euclidean space
Collision System	40	~ 2 weeks with two people
Math Implementation	80	~ everyone will probably spend 10-20 hours rendering complex math

Demo Creation	200	Overestimate Assuming previous tasks will take more time Want to create a solid product ~ 1 month and some
Creation of Working Demo	120	
Testing Working Demo	80	

3.7 Other Resource Requirements

In terms of game development, additional resources will be needed as discovered during the prototyping phase of game design development. The additional resources needed are:

- Sprites, Images

  For the game development itself, using sprites is required for the game development, with sprites allowing for simple drawing operations to be performed over shapes rendered with the engine. Due to the fact that none of the team are art students, acquiring sprites and images that are created by professionals would be preferable and result in an overall polished and presentable game.

- Libraries for Engine Development

  In developing the game engine, libraries must be used to ensure consistent execution of code across platforms. Due to the nature of creating windows and processing input to write textures and shapes to the screen, various data structures must be used to copy data for use by the Graphics Card. Writing custom code to perform these tasks will be impossible to do in addition to the general engine development required to be done in the timeline of the course. Therefore, using established libraries is incredibly important and useful.

- Student Innovation Center Game Lab Access

  During the game development prototyping process in Unity, members of the game design team have been unable to efficiently run prototypes on personal laptops. Gaining access to the Student Innovation Center Game Development lab would greatly help with development and viewing or creating the prototyping for Proof-of-Concepts.

- Unity Version Control

  Using Unity Version Control (UVC) is necessary for the Game Design team as many of the game assets are extremely large files and will get merge conflicts with other version control systems. The Game Design team has been able to use the free version of UVC that allows up to 5GB of storage space. However, it is possible that we need funding for the pro version of UVC as the final game may exceed this 5GB limit.

Overall, art, libraries, game lab access, and UVC may be necessary resources for our project. Well made sprites, and images are needed to satisfy the aesthetic user requirements. Libraries are necessary to ensure we are not writing unnecessary code. Student Innovation Center lab access would assist members of the game design team with prototyping on Unity. Finally, paying for UVC would increase the productivity of our team.