# Game Design Document

Jackson Wiebe

3519635

# Game Title

Mineral Mayhem

# Backstory

"Mineral Mayhem" is a top-down 2D mining game inspired by the style of "Motherload." Players take on the role of a miner navigating through underground environments, digging for valuable minerals, upgrading their equipment, and facing challenges beneath the surface. The first implementation will be a peaceful player versus environment type of game wherein the goal is to simply finish the game without running out of money.

Example playthrough of Motherload:

<https://www.youtube.com/watch?v=ZJ_3R854yuw>

# Compile and run

## Requirements:

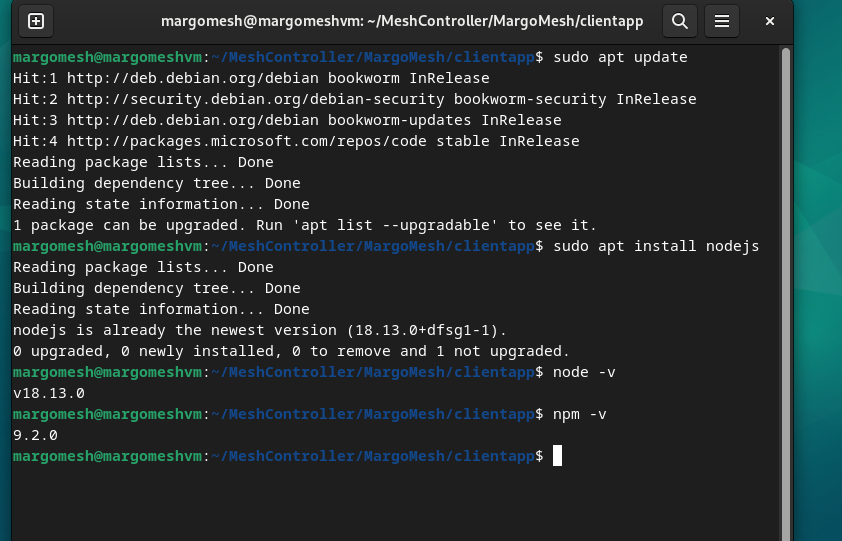


Figure Installing NodeJS

* Node JS version 20+
* Modern computer

## NodeJS instructions for Debian

1. Open a terminal window by pressing Ctrl + Alt + T or searching for "Terminal" in the applications menu.
2. Update the package list to make sure you have the latest information about available packages:

sudo apt update

1. This command installs both Node.js and npm.

sudo apt install nodejs

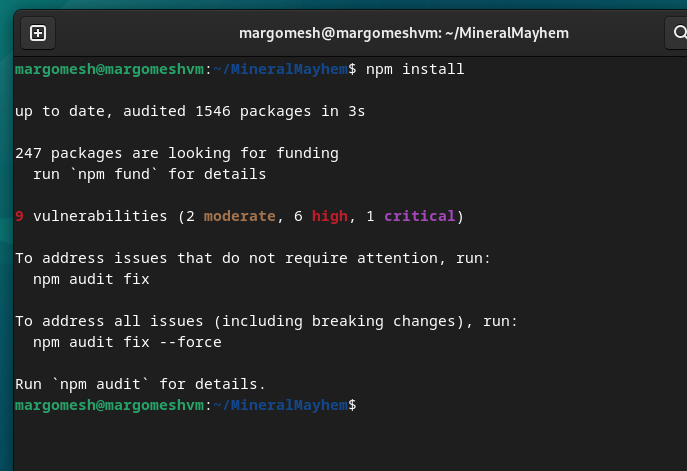


Figure - npm install

1. To check if Node.js and npm were installed successfully, you can run the following commands to check their versions:

node -v

npm -v

## Compile and run

1. Browser to the application directory with a new terminal.
2. Run the following command to pull all the dependencies and build them locally:

npm install

1. Use the follow command to compile the application

npm start

1. A new browser window should open to host the application. Otherwise the default URL is

<http://localhost:3000/>

A pre-compiled binary is also located in the /Build folder which can be hosted on any webserver equipped with nodeJS. A full working build of this game is also available at the following URL:

<https://jacky4566.github.io/>

I also created a video on how I built this game here:

https://youtu.be/RX8F9feYW-k

A screenshot of a computer screen

Description automatically generated

Figure npm start

# How to play

I have created a quick video showing how to play the game here:

<https://youtu.be/mIRE0FBIuu8>

# Characters

Player - Will pilot a mining spaceship that digs “down” to collect minerals.

# Gameplay

## Controls

* Arrow keys/ WASD for PC and finger position for mobile.
* The miner can dig when a direction is held
* The miner can not dig upward

## Mining and Collecting

* Game Cell types

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Material** | **Time to Mine (Seconds)** | **Min. Depth Block** | **Freqency** | **Damages Player** |
| Dirt | 0.25 | 0 | High |  |
| Bronze | 0.25 | 0 | High |  |
| Silver | 1 | 0 | High |  |
| Gold | 2 | 25 | High |  |
| Quartz | 4 | 250 | Med |  |
| Obsidian | 10 | 500 | Low |  |
| Diamonds | 30 | 500 | Low |  |
| Lava | Not Minable | 100 | Low | Yes |
| Water | Not Minable | 100 | Low | Yes |

* A storage system allows players to manage their collected minerals.

## Challenges

* Limited fuel for the mining vehicle adds an element of strategy.
* Player death may result from:
  + Falling too far
  + Running out of fuel
  + Environment variables such as lava and water pits

## Progression

* Different underground layers with increasing difficulty.
* Achievements and milestones for reaching specific mineral collection goals.
* Player will need collect all required items to progress

# Game World, Art and Design

## Visual Style

* Pixel art with a colorful and vibrant underground world.
* Clear distinction between soil layers and minerals.
* Block style similar to Minecraft and other crafting games

## Characters and Environment

* Customizable miner character.
* Dynamic underground environments with diverse backgrounds.
* The layout of the game will be a fixed grid width, something like 64 squares wide and possible 1024 squares deep.

## User Interface (UI)

* Clean and intuitive UI displaying player stats, inventory, and upgrades.
* Depth, pressure, and environment stats

## Level Generator

Each level will be unique and contain an array of items to randomly generate. The LevelGenerator.js class handles this by reading the level data and generating the data into placement that are returned to the level state class.

# Test Plan

For this application we can test each class for proper operation and modify the code slightly to render special cases for testing.

For the Tile generation

* Does the Sprite.js function correctly render each sprite correctly against a black and white background. Is there any clipping of the graphics?

For the Camera:

* Does the camera move correctly with the character.
* Does the camera float slightly higher at the top of the level
* Does the camera get stuck on edgese or levels of various size

For the Collision:

* Do all the functions in the collision.js class work properly.
* Can we collide with walls and solid objects but not collectables and the end goal

For the Sounds

* Do the sound effects play properly without overlapping

For the Direction system

* Can the user input directions properly
* Does the touch controls work for mobile devices

For Inventory:

* Are we collecting items properly, is there any overflow
* Does each item have a unique key for identifying

For Placement factory:

* Are we placing items properly and executing their respective TICK functions.

Lastly of course we play through the game.

* Can we finish each level with the randomly generated items.
* Are the levels of reasonable difficulty.

# Future development

I want to use this game for my final project as well so expanding the game will include:

* Adding a full combat system with enemy and a final boss
* Adding equipment for weapons
* Adding new armour types
* Expanding the mineral types
* Add AI players that can take minerals before the player.

# Appendix 1: Gui Examples

# Appendix 2: Graphics example from similar game Motherload

# Motherload - Play Online on SilverGames 🕹