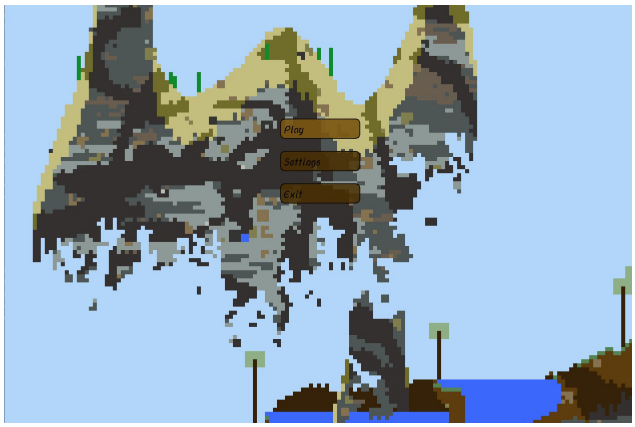
Aeronef (*github.com/mikedillender/Aeronef*)

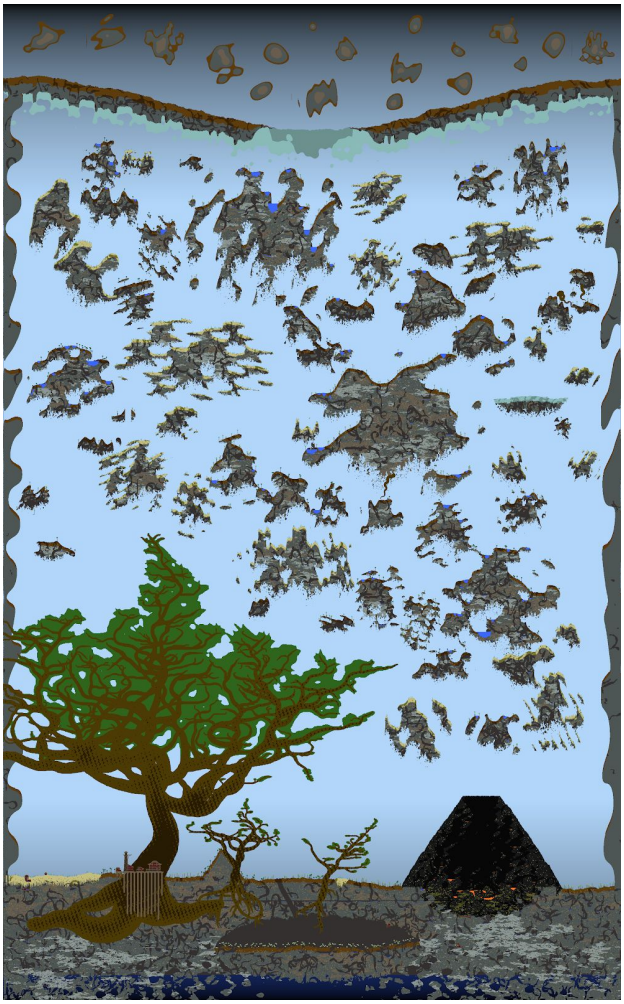
Video documentation can be found on the "Mike Dillender" YouTube Channel



This has been my most creative and long-running endeavor, currently sitting at 32,477 lines of code. As such, it would be nearly impossible for me to describe the creation of it within a reasonable word-count. Instead, I will explain some of the interesting and complex features/algorithms that are incorporated into it.

Notable Algorithms:

- Random World Generation
  - The most complex algorithm is the world generation. Every time a player starts a new game, a new world is procedurally generated, one of these maps is shown to the right. This process alone is around 8,000 lines and takes most computers around 45 seconds to complete.
  - Noise Generation (*github.com/mikedillender /MiscApplets/tree/noisegencontroller*)
    - I independently developed a process for noise-generation using an array of sinusoidal vector functions. This is demonstrated by the unique shapes of the islands in the above images of map generation.
  - Cave Generation
    - Caves are created like worms, tracking their path and removing tiles they pass through.
    - This algorithm was adapted to create the world tree in the bottom left of the map.



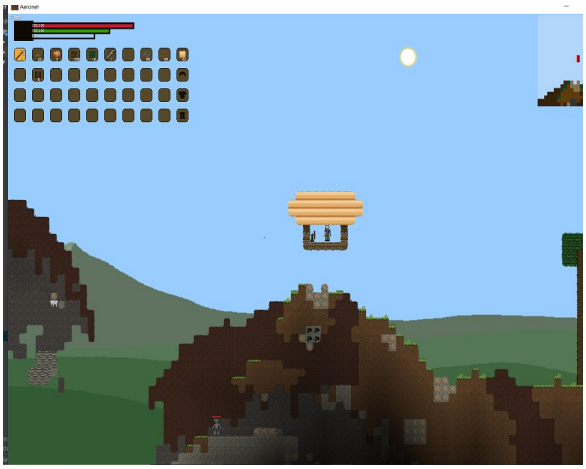
- Island Generation
  - Uses noise-generation to create a unique shape, then runs a recursive decay algorithm on the bottom to make it seem as if the islands have eroded and fallen off. After this, caves are generated all throughout the island.
  - After erosion and cave-gen, grass and trees (or sand and cactus) are placed on top of each island, making them look more natural and pleasant.
  - Finally, each island is placed into the most empty region of the map.
  - Each map consists of around 50 islands of varying sizes.

- Pre-Built Structures
  - Structures can be drawn in *Photoshop* and saved directly as a .png, I have written a script that allows the game to convert this .png data into a structure, and can be placed into the world. This is used for the village at the base of the world-tree.

- Enemy Pathing
  - The enemies use an adaptive pathing system to follow the player
  - They can not only navigate through mazes (finding the fastest possible route), but jump over pitfalls if they are in the way.
- Lighting
  - The lighting system is very efficient, and unlike many similar games, is extremely smooth.
  - The system allows for updates to the lighting every frame, making entity lights possible.
  - Of course, blocks and tiles block light.
  - Certain tiles in the open-air act as light-sources along with torches, but their brightness changes based on the time (i.e. it is dark during the night).

- Fluid Dynamics
  - Water flows from tile to tile to find its level.
  - Obviously, the fluids cannot flow through blocks.
- Animations
  - All entities are animated to some degree
  - All sprites can have up to two degrees of rotation, which is rather difficult to achieve.

- Airships
    - Players can not only create their homes from materials they gather but can turn those homes into airships.
    - Players can control the amount of air in the balloons, controlling their upward lift.
    - If not powered by propellers, the ship is controlled by the wind (which can be determined based on the movement of the clouds in the background)



- Steam Power
    - This is highly integrated into the aforementioned airships. Taking heavy inspiration from the steampunk genre and gilded age technology, the player can create a steam boiler and use it to power an engine.
    - Steam engines can rotate gears, which in turn, can rotate propellers.
        - These propellers, when placed on airships and used with valves, can turn the player-built ships into thermal airships, allowing for full control of its motion.
- Efficient Tile Rendering
    - World files are 2,520x4,050 tiles (for reference, the player is 1x2 tiles), but it only renders the tiles that are on screen, and visible.
    - If the tile is too dark, it will not render, making it much more efficient.
- Automatic World Saving / Loading
    - Whenever the player changes the environment, the change is seamlessly saved to the hard-drive, so no matter when the user exits the game, their data is saved.
    - It also independently saves the player's location and inventory data, and minimap data.
    - Along with player change saving, the biggest save file is the full map.
        - Most world files are 2,520x4,050 tiles, but large worlds are 7,560x12,150 tiles.
        - Each tile consists of five layers: block, wall, complex block, pipe, and liquid.
        - Uncompressed, these files are around half a gigabyte, but with my compression algorithm, they are saved at a mere 17 megabytes.