Journal / Log

2022-10-21

- Think about a game, pulling ideas from games like Space engineers (thematically and graphical dream) , Aurora 4x (for the depth of gameplay), From the depths (for the advanced physics simulations and freedoms)
    - Very little from Aurora, maybe also look into how we can make orbits fun
        - Orbital maneuvers? Like Kerbal Space Program?
- Aircraft physics
- In depth weapons systems
- In depth propulsion systems
    - Fusion torches require power, resources, cooling
    - Possibly implement something like the Elite Frame Shift Drive
        - Exclude witchspace as it gives too many freedoms and would make the project too large
        - Include *supercruise* as it enables travel at speed but still leaves the player vulnerable to encounters and such
            - *Supercruise interceptor?, AI Pulling players out of supercruise? Players pulling players out of supercruise?*

- Player is free to move like FtD and SE

    - Player does things to kickstart the major processes like SE
    - Maybe make production of things more factorio-like than SE like

    - **TODO**

        - Figure out how to do air sim (Airgear?)
        - Figure out how to do planet orbits *efficiently*
        - Decouple the camera from being a player so we can have TV Screens unlike SE

        ### Progress report

    - Create project and begin a new godot scene

    ## 2022-10-24

    - Should support N-body physics if time allows for such

- Basic Kepler's laws if unable to do so

2022-12-06

- Begin pulling ideas for inventory management systems from SE, Mindustry, and Factorio

- Find a developer's creation of factorio belts on github

2022-12-30

    - Find spacescape, an MIT-licensed space-cubemap generator
- Allows me to make programatically generated sky images
- Generate skybox and use python cube2sphere to convert it into a panoramasky as **godot cannot take cubemaps**
- requires `convert -rotate 180 in.exr out.exr` on the top and bottom faces as spacescape outputs them in the wrong order for cube2sphere by default

2022-12-31

    - Use ideas from Dr. Chris Thorne's papers to implement a form of floating origin

```
.
└── Viewport
    ├── Player Camera
    └── World Root
        └── Object (e.g. an asteroid or ship)
```

2023-01-25

- Finalise the structuring of the client and server architecture that is required to combat floating point origin
    - Server computes nearly everything
        - Due to engine limitations **two** godot projects will be used (server, client)
        - Server uses high precision numbers (64/128bit) and uses an absolute coordinate system
            - Sends it to the client in client-relative coordinates (remote player is 0,0,0) to combat FPPE on the client
                - By default most game engines use F32 (godot can be recompiled with F64), but **all GPU's** use F32
    - Client is a near perfect thin client
        - Current arch still allowing for the client to be thickened later to act similarly to Quake and contemporary games that can still be played even after losing a server connection

Solving Problems

- Large distances from world origin **will** cause issues with FPPE (Float Point Precision Error)
    - Can be solved by Floating Origin described by Dr. Chris Thorne
    - Split the world into large clusters (a very small resemblance to MC chunking system but avoid FPPE)
        - Space Engineers' Implementation
        - Floating Origin series of papers by Dr. Chris Thorne.

Progress

- General
  - ~~Chunking system~~ (Rejected)
  - [ ] Player Movement and Camera
    - [ ] Fall damage / General Velocity Damage
    - [ ] Player movement is limited - [ ] All done via *forces* as player weight matters - [ ] ~~Should be able to have target inertia relative to nearest object / planet~~ (Rejected)
      - [ ] Should have velocity of planets and objects when sent, be relative to the player
        - [ ] If grounded then do not rotate vertically when looking up and down
        - [ ] If not grounded freelook as long as there is ΔV left to consume
        - [ ] Use an acceleration to look so it *feels* like you are using real RCS
          - [ ] Should (rotational) deceleration be instant?
      - [ ] Else limit range of look to like ±30°
- [ ] Planets
  - [ ] Planet Gravity
  - [ ] Planet Orbits
    - [ ] Wikipedia N-Body Simulation
    - [ ] GitHub Simulation projects
  - [ ] Voxel planets
  - [ ] Optimise planet orbits with a Barnes Hut tree
- [ ] Graphics
  - [ ] Star cubemap as background
  - [ ] Particle effects in godot for things like thrusters
- [ ] Audio
- [ ] AI players
  - [ ] Can pilot ships
  - [ ] Can engage in foot combat w/guns
  - [ ] Can pilot rovers
- [ ] Save Reload System
  - [ ] Save and reload planets
    - [ ] Get position and velocity working
    - [ ] Get planet textures working
- [ ] Resource Management
  - [ ] Basic transmission of items into and out of blocks
  - [ ] Mining resources with voxel deformation like space engineers

Useful Resources (and uses in game)

- nVidia GPU Gems (CUDA TARGET) - Physics Sim Chapter + N-Body
- Simple implementation of astrophys universe
- Sebastian Lague's series on procedural planet generation
  - Spacescape
- Main space background

    - Community Godot Shaders
    - FLECS for a C/C++ ECS system that is able to work with Godot
    - ECS allows for a large (10-100x) speed increase in processing as it reduces events such as cache misses and is data oriented

Creation of documentation (del after maybe?)

- CTAN Animation Package

Documentation of submodules

Pipeline (/conveyor) system

- Conveyor networks need to be analysed to find possible paths
    - Single conveyors can be grouped into transport groups, with 2 I/O ports
    - Do something like a layer 3 router to optimise paths?
    - Time must be taken and the maximum throughput of a path must be determined

Main ideas

- The game doesn't go heavy on plot
    - Relies on light environmental storytelling and short snippets of text
        - Simulation games like Minecraft, Factorio, Space Engineers, follow a similar style
        - Reduces authoring workload while keeping the game fun by not lore-dumping (large chunks of text shown to the player which cause disengagement)
- Make a game with the complexity and shipbuilding of space engineers and factorio
- Make a game with good AI that is legitimately a threat to the player
    - To the level where playing against AI is fun and a constant threat