Legend

Friday, September 23, 2016 8:34 PM

Legend

- Things in normal text are part of a specification. They can be highlighted as in the "Specifications" section:
  - Green items have been completed.
  - Blue items have an untested implementation.
  - Yellow items are incomplete.
  - Magenta items require more information before work can be done.
  - Non-highlighted entries have not been worked on.
- Things in *italics* aren't part of a specification or are optional elements.

Mathematical Notation

Thursday, September 15, 2016 6:30 PM

- c indicates a real number without a unit or with a monetary unit. This is generally added to something, or multiplied with a factor then added.
- k indicates a normalized non-negative real number in the range [0, 1]. This is generally a factor on some parameter.
- n indicates a non-negative integer with or without a unit.
- m indicates a mass, represented by a non-negative real number.
- d indicates a distance, represented by a non-negative real number.

Table of Contents

Sunday, September 18, 2016 8:54 PM

Summary

Design documentation and specifications for the game.

Table of Contents

- Package - Engine
- Package - Game
- Section - Pipeline Design. Specifies pipeline operation and layout.

Legend

- Project as a whole is a product.
    - The product contains packages, which are groups of items that are installed together to the user's system.
        - Packages contain artifacts, which are distinct executables or libraries.
        - At any of these levels, the documentation for a level can have sections to compress the page count.
        - A level can also have package/artifact groups if the number of items involving the topic aren't yet known.

Resources

Friday, September 23, 2016 4:51 PM

- *https://blog.molecular-matters.com/*
- *http://www.geomerics.com/blogs/quaternions-rotations-and-compression/ - quaternion compression!?*
- *MMO Server Architecture: https://gameserverarchitecture.com*
- *Networking: http://gafferongames.com/*
  - *http://gafferongames.com/2016/05/10/building-a-game-network-protocol/*
  - *http://gafferongames.com/networking-for-game-programmers/what-every-programmer-needs-to-know-about-game-networking/*

Implementation Choices

Thursday, July 28, 2016 5:29 PM

- *PhysX 3 is open source and has been for a while now. Use that and not Bullet if possible, since Bullet's API is all over the place. ODE is also possibly acceptable?*
- *If you're going to do any multiplayer **and** you want large numbers of units (more than a couple hundred), there's a couple of options:*
  - *If player count is small (2-4?), you'll need deterministic lockstep, which means you need deterministic physics. There's no good, reliable way of doing this cross platform, but people have tried and you can look at their remarks: http://gafferongames.com/networking-for-game-programmers/floating-point-determinism/*
    - ***Important Metric:** How much bandwidth given some number of units?*
  - *If player count is large, you'll need a client-server system with interpolation. Lockstep is as slow as the slowest player. Again, blog entry: http://gafferongames.com/networked-physics/snapshots-and-interpolation/*
- *Binaural audio is really, really immersive. Not clear how good vertical or depth positioning sounds, but it sounds vastly different from just having 5.1/7.1. Main problem is that you have to configure for speaker or headphones, can't have a transformer that does both. Secondary issue is that you might need to make a transformer yourself if the available libraries aren't compatible.*
  - *Without this you basically can't make the game accessible.*

Physical Scale

Friday, September 9, 2016 2:35 PM

- *NASA-STD-3001 VOL 2 spec: Humans can take ~3 seconds of 17g eyeballs in, which translates to 500 m/s delta-V. This can be considered this world's version of a "boost". We can take sustained accelerations of 4g, or 39.2 m/s^2.* ***Movement is on the scale of kilometers for strike craft at the very least****, but without something to dampen experienced acceleration it'll be very jousty: you boost boost boost to the target, hit, then boost boost boost away. There's very little space to change your mind since the initial approach forces any course correction to use up lots of time with more boosting.*
  - *This is raw g, before adding inertial dampening. How much does an actual fluid chamber dampen inertial effects?*
  - *Eyeballs out is a bit better at 3 seconds and vastly better at 1 second, but you'd experience rotational forces backwards which would be incredibly disorienting.*
  - *What are the caps for thrust at different size classes? Remember that all of this generates heat too.*
  - *In addition, hard control limits should probably be set to keep new players from flinging themselves uncontrollably into a suicide charge. With the range on weapons, going too fast relative to the battlezone means you can't really turn to evade after a certain point.*
  - ***Is it possible for ships of one size to move at speeds expected one size lower (in exchange for huge heat/supply issues)?***
- *http://worldbuilding.stackexchange.com/questions/32549/attenuation-of-a-laser-in-space thinks ~200km effective range for single large beam that generates similar effect as 1.3 PW laser array (https://web.archive.org/web/20080614224504/http://newton.ex.ac.uk/aip/physnews.401.html#3).*
- *Most likely basic ranges will be dictated by the range at which a laser turret can track and instakill a bomber; that in turn decides what weapon a bomber carries. The range and time to impact for the bomber's weapon then decides how much engine power can be dedicated to the bomber's attack run, and the time taken for the bomber to reach the launch point after enemy detection decides how quickly an interceptor must be able to move.*

Design Questions

Friday, September 23, 2016 8:32 PM

- *How do we test the products?*
- *How do we package the products into installable units?*
  - *People may want the packages as stand-alone archives, or they might not have permission to install the package properly on their systems.*
- *How do we update packages?*
  - *How do we handle add-ons?*
- *In general - remember that **whenever you have an external dependency you have a wrapper.** May be worth printing on a wall, to be honest.*
- *Redmond has a game development guild.*
- *Some group in SVC does advising if you want to sell a product as a Microsoft product?*

Project Layout

Friday, September 23, 2016 5:47 PM

Build and Distribution System

Distribution system split into three major systems:

- ### Credentials Storage
  - *Securely retains administration credentials.*
  - Credentials stored on a private repository to synchronize credential updates.
- ### Server Complex
  - *Constructs and distributes product.*
  - Consists of a set of DigitalOcean instances. See Pipeline Topology & Price Estimation.
  - Server configuration stored on a private repository to synchronize changes.
- ### Code Storage
  - *Maintains source code changes to the product.*
  - Product elements may have different licenses; because of this there are two repositories:
    - Public repository for open-source packages/artifacts.
    - Private repository for closed-source packages/artifacts.

Product

Product is split into two packages, which are distributed as archives/installers:

- ### Game
  - *This gets distributed to end users.*
  - #### Game Server
    - *Executable or package of executables that simulates the game world.*
    - Artifact should be closed source.
  - #### Game Client
    - *Executable that connects to the server and sends/relays state requests. Users use this to play the game.*
    - Artifact should be closed source.
  - #### Game Tools
    - *External tools needed to process products used by or made by the server or client.*
    - Artifact can be open source.
- ### Engine
  - *Libraries and executables that provide functionality common to all products in the game package.*
  - Package can be open source.
  - #### Engine Modules
    - *Libraries that provide said common functionality.*
    - See [Specifications: Engine Milestones] for the full specification of engine subsystems and modules.
      - **All engine modules must be libraries, no executables are allowed.**
  - #### Engine Tools
    - *External tools needed to process products used by or made by engine modules. Generally executables, though they could be libraries if silo-ing them from the engine itself seems acceptable.*

Table of Contents

Friday, September 23, 2016 7:44 PM

Summary

This is all of the products needed to develop the game; add-ons thus require the engine tools in this package.

Table of Contents

- Artifact - Engine Modules. Libraries that provide functionality common to all products in the game package.
- Artifact - Engine Tools. External tools needed to process products used by or made by engine modules.

Design Questions

Friday, September 23, 2016 8:41 PM

- *Engine is responsible for describing *how* to do everything, not necessarily what should be done.*
- *Since the game is planned as a client-server node system, different nodes don't need the whole engine in memory. If a machine only does site updates it needs physics, event transmission and the like, but it *doesn't* need AI, for example...*
    - *But then that means we need some way for the build system to know what engine modules each node type depends on.*
    - *Alternately we could use dynamic libraries, but we need to then read up on exactly how they behave:*
        - ***How does Rust generate dylibs (if it does)?***
        - ***How does each OS load dylibs? Is it possible for multiple copies of the same version to load?***
        - ***What overhead do dylibs generate versus static libraries?*** *Remember that the static overhead can add up on the server side since you're supposed to spin up nodes to scale to the game world.*
        - ***How do we maintain dylibs as the codebase evolves?***
        - ***How do we update dylibs?*** *Although this is a question for the entire application.*
- *In either case the engine must be split up into separate libraries now unless the whole thing is that small of an overhead.*
    - ***What libraries will we split the engine into?***

Milestones

Friday, September 23, 2016 5:18 PM

Table of Contents

Friday, September 23, 2016 8:37 PM

Summary

These are all of the libraries, whether dynamic or static, that compose the common functionality used by the Game package.

Table of Contents

Design Questions

Friday, September 23, 2016 8:44 PM

Description

Friday, September 23, 2016 8:38 PM

- *These are all of the libraries, whether dynamic or static, that compose the common functionality used by the Game package.*
- *See [Specifications: Engine Milestones] for the full specification of engine subsystems and modules.*
  - ***All engine modules must be libraries, no executables are allowed.***

Milestones

Friday, September 9, 2016 11:52 PM

Engine Subsystems

Highlight key:

- Green items have been completed.
- Blue items have an untested implementation.
- Yellow items are incomplete.
- Magenta items require more information before work can be done.
- Non-highlighted entries have not been worked on.
- Optional items are in *italics*.
- (Comments are in parentheses.)

1. **Platform Independence**
    1. Datatypes
        1. (Sorta have this in the f32, i32, etc. types.)
    2. OS Dependent Wrapper
        1. File System. Use existing library if possible.
            1. File Open/Close (std::fs)
            2. Directory Enumeration (std::fs)
        2. Timer. Must be performance counter.
            1. (Not really a good speed guarantee for this; crate chrono can do this, but is focused on time correctness. You will probably have to implement this yourself.)
            2. Windows: QueryPerformanceFrequency()/QueryPerformanceTimer()
            3. OSX: mach_absolute_time()
            4. Linux: clock_gettime(CLOCK_REALTIME_HR)
        3. Input Device I/O (Raw Input). No existing library found.
            1. Input API Specification
        4. Supported OSes
            1. Windows
            2. *Linux*
            3. *OS X*
    3. Threading System. Use existing library if possible. May be OS dependent, may not. Very hard to tell right now.
        1. (Rust appears to have a threading system; see docs. Make a thread object with std::thread::spawn(), which takes an anonymous function of the format "move || {…}". Join the thread with [thread name].join.)
    4. Graphics Wrapper
        1. Specification
            1. Function Loader - if wrapper needs to load functions, do so before marking as ready.
            2. Function Wrappers - renderer makes these raw calls and doesn't see the actual implementation.
        2. Supported Frameworks. Use existing libraries if possible.
            1. OpenGL. Either Glium or gfx-rs will handle this.
            2. *DirectX*
    5. Audio Wrapper. Check https://github.com/RustAudio .
    6. Physics System/Wrapper. May need to hook to C++ library if none in Rust is viable.
2. **Core**

1. Memory Allocation
2. Module Startup & Shutdown
3. Assertion System. Use existing library if possible.
    1. (std::assert!() provides basic functionality with custom comment.)
    2. Comments
    3. Failure Location
    4. Failure File
    5. *Failure Function*
4. Debug Logging. Use existing library if possible.
5. PRNG. Use existing library if possible.
    1. (Crate rand is supposed to handle this, but it hasn't been updated in half a year. But then you really just need a Mersenne Twister, nothing cryptographically secure.)
6. Math Libraries. Use existing library if possible.
    1. High speed functions:
        1. Square root
        2. Power
        3. Logarithm
        4. Sin/Cos/Tan
        5. Arcsin/Arccos/Arctan
        6. (Holy crap, the standard library just has it on the numeric type; not clear how fast the implementation is though, so we need to consider implementing this ourselves.)
7. Strings & Hashed Strings. Use existing library if possible.
8. Engine Configuration
9. Profiler/Stat Gatherer. Use existing library if possible.
    1. (Remotery can do this, and it has a Rust binding through remotery-rs. Displaying in-game requires us to do some wrapping for our own implementation though.)
10. Object Handles & IDs. Intend a DoD design; this implies a component system and sparse arrays.
11. *Curves & Surfaces*
12. *Reflection*
13. *Serialization*
    1. Available via crate rustc-serialize.
14. *Localization*
15. *Async File I/O*
16. *Movie Player/Replay System*
17. *Debug Menu/Console*
    1. (Could just be a literal ~~Lua~~ Squirrel interpreter.)
3. **Resource Manager**
4. **High-Level Input**
    1. Control Abstraction
        1. Keyboard
        2. Button
        3. Axis
    2. Command Abstraction
        1. Button-Button Combinations
        2. *Button-Axis Combinations*
    3. I/O Filtering
        1. Deadzones
        2. *Noise Filtering*
            1. *(Some broken controllers will randomly jump from actual position to some false position. Normally this is an obvious shift in amplitude and should be easy enough to*

filter.)

3. *Axis Curves*
    1. *(3rd party drivers can do this outside of the game, so it may be better to just post a link to it since it's an advanced topic and would centralize the player's settings.)*

5. **Collision Detection/Physics** Try to use an existing library for the backend.
6. **Low-Level Renderer**
7. **Scene Graph**
8. **Front End**
9. **Gameplay Base**
    1. Game Flow System
    2. Scripting
        1. (Incomplete, see LeEK 1: “Scripting Notes”. Will likely redo with Squirrel.)
    3. Static World
    4. Dynamic Object Model (Actor and Components)
    5. Agent System (AI hooks)
    6. Event / Messaging System
        1. (Need to get this right the first time; just using a notification hub leads to a tangled mess.)
        2. *Dynamic Music*
    7. World Loading

Table of Contents

Friday, September 23, 2016 8:37 PM

Summary

These are the external tools needed to process products used by or made by engine modules.
Remember that these do not have to be written in Rust!

Table of Contents

Design Questions

Friday, September 23, 2016 8:44 PM

Description

Friday, September 23, 2016 8:36 PM

- *These are the external tools needed to process products used by or made by engine modules. Generally the tools are executables, though they could be libraries if silo-ing them from the engine itself seems acceptable.*
- ***Remember that these do not need to be written in Rust; in fact it is probably better to work in a much higher-level language so these tools can be prototyped faster.***

Milestones

Friday, September 23, 2016 8:40 PM

- 3D art to model file processor
- 2D art to texture processor
- Audio processor
- Script processor?
- Packaging system to pack processed assets into monolithic asset files
    - The server might need packages too, hard to say.

Table of Contents

Friday, September 23, 2016 7:57 PM

Summary

Package used by the end user to play and host the game.

Table of Contents

- Artifact - Game Client. The view/controller of the game.
- Artifact - Game Content. The audio/visual assets used by the game.
- Artifact - Game Server. The model of the game.
- Artifact Group - Game Tools. Auxiliary programs used to manage the client or server.

Design Questions

Friday, September 23, 2016 8:44 PM

Overview

Friday, September 23, 2016 4:48 PM

- *Client-server system. Client responsible for displaying and requesting changes, server responsible for validating and communicating changes to client.*
- *Server authoritative; beginning with this in mind will prevent a lot of headaches if we ever do multiplayer.*
- *Single player is thus the client connecting to a server on the same machine; we'll call this local mode. When connecting to a remote machine we call it remote mode. If the remote machine is part of a distributed cluster it's remote distributed mode.*

Communications Methods

Friday, September 23, 2016 7:14 PM

- *Client-server communications use UDP remotely.*
  - ***What does it use locally?*** *Threads? IPC?*
- *In-server communications need to use some kind of IPC. If we're in local mode we can instead use the local client-server comms method.*
  - ***What remote distributed mode IPC would we use?***
- ***How much physics do we actually need? Every bit that can be disabled or removed will be vital.***
- *The rest is for multiplayer:*
  - *What packet-level synch method will we use?*
    - *What we know:*
      - *The number of actual players in a given site is always low (2-10).*
      - *The number of NPCs taking commands from the players is really high (5-100 per player = 10-1000 NPCs)*
      - *Haven't decided if players need to directly order NPCs around.*
      - *Haven't decided if players can change ship while in a site.*
    - *Choices:*
      - *If player count is small (2-4?), you'll need deterministic lockstep, which means you need deterministic physics. There's no good, reliable way of doing this cross platform, but people have tried and you can look at their remarks: http://gafferongames.com/networking-for-game-programmers/floating-point-determinism/*
        - *Important Metric: How much bandwidth given some number of units?*
      - *If player count is large, you'll need a client-server system with interpolation. Lockstep is as slow as the slowest player. Again, blog entry: http://gafferongames.com/networked-physics/snapshots-and-interpolation/*
    - *It's not clear if we can use some weird hybrid. Probably not, one player seems to always get one connection.*
  - ***Can we use a library for the packet-level communication?*** *If not, we need to test it.*

Factions

Monday, September 12, 2016 12:02 AM

- *Republic*
  - *High alpha damage, short range.*
  - *Short FTL range. Focus on smaller craft; capitals aren't very good.*
    - *Their idea of a subcapital is smaller than an Empire subcap, more akin to a strike craft with an interstellar drive. C.f. fast attack craft, torpedo boats.*
    - *Strike craft role often fulfilled by drones partnered to a subcapital. Leaving them to SI means they're not as alert or manuverable as a remote fighter, but they're still fast in sublight, can be placed on anything, and can't be jammed.*
  - *Use self-guiding weapons and drones. Missiles are a favorite.*
    - *Specialty weapon is the hilarious/horrifying railgun-launched missile. Yes, these might be seen on their strike craft.*
      - *Main problems are loading and maintaining the barrel; you're firing a much larger mass and it must be launched without damaging the missile.*
  - *Heavy use of SI automation.*
    - *Fleets do not suffer much morale problems.*
    - *Fleets do not gain many morale bonuses, either.*
    - *Fleets are vulnerable to decapitation strikes - if a commander is lost, it may drive commanders of other fleets in the fight to retreat. SI in a headless fleet are also much less effective.*
    - *All that automation means less crew and lower Supply costs.*
- *Empire*
  - *Long range, less focus on alpha damage.*
  - *Long FTL range (this doesn't mean they are always faster in-system!). Focus on larger craft to increase overall range of fleets.*
    - *Strike craft are smaller and focused on agility, both sublight and in FTL. Almost always remoted from a carrier/base.*
  - *Use turrets and spinal-mount weapons. Carriers focus on craft capacity.*
    - *Specialty is the phased array turret, which doesn't need to physically turn (though it should, range weakens at the sides). Laser turrets are available at smaller sizes compared to Republic ships.*
    - *Working with Republic tech leads to warp torpedos, extra large missiles that can do in-system warp. See [General:Weaponry].*
  - *Heavy use of crew. Automation is rare and centralized in advanced AI.*
    - *Fleets benefit heavily from morale bonuses.*
    - *Fleets fall apart quickly if morale is low!*
    - *Fleets can carry on without their commander; depending on the faction they may fight harder.*
    - *Larger crew counts mean higher Supply costs.*

Electronic Warfare

Friday, September 9, 2016 6:39 PM

- *Detectors/Navigators can attack each other "psychically". This is very key since shutting down a Navigator in deep space halts their whole fleet, and shutting down a Detector leaves their fleet strategically blind.*

Basic Concepts

Saturday, September 17, 2016 11:35 PM

- Celestials *are natural constructs in space.*
- Entities *are artificial constructs in space.*
- *Two main categories of entities:*
  - *Ships*
    - *Can move under their own power.*
  - *Stations*
    - *Cannot move under their own power.*
    - *Must be placed on or relative to a celestial.*
- *All entities have a Size, which abstracts its overall mass and volume. When inside a container, they take up that many size units.*

Civilian/Logistics

Saturday, September 17, 2016 11:32 PM

- *Ships*
  - *Subcapital*
    - *System Hauler. Carries small freight quickly inside a system. Can travel between systems but is slow about it.*
  - *Capital*
    - *Bulk Hauler. Carries significant amounts of cargo between systems. A bit like a big empty carrier, not many defenses if any.*
  - *Supercapital*
    - *Mobile base. Self-sufficient facility that generates Supplies.*
    - *Colony constructor. Builds colonies out of asteroids or on planets.*
- *Structures*
  - *Each category can be represented by a station that has the given structures in it.*
  - *Industry*
    - *Power plants/collectors*
    - *Mines*
    - *Factories*
  - *Detection systems (radar, etc)*
    - *FTL detector requires Crew and thus more Supplies.*
  - *Communications*
    - *Com tower. Decodes FTL signals and broadcasts the signal to com relays that send over EM bands.*
      - *Requires lots of crew to keep the FTL decode going and lights the system up on FTL detection. If you're not a large faction that can defend itself you should stick to fleets and point-to-point comms.*
      - *Can be hacked; see BlueDot design info for this.*
    - *Com relays. Handles communications to and from a system's com tower within a certain lightspeed latency radius.*
      - *Can be completely unmanned since it's all EM muxers/demuxers, but it's a bad idea if boarding is a thing.*
      - *Can be hacked; see BlueDot design info for this.*

Military/Combat

Saturday, September 17, 2016 11:32 PM

- *Ships*
  - ***Important: Are interdictors something that only a dedicated ship can do?***
  - *Strike craft. Do not have an interstellar drive.*
    - *Bomber. Attacks ships larger than strike craft.*
      - *Maximize damage.*
      - *Maximize forward agility.*
    - *Interceptor. Attacks bombers and other interceptors.*
      - *Maximize agility.*
    - *Warp Torpedo. SI-guided missile that uses intersystem drive to strike at range.*
      - *Maximize damage or agility.*
        - *Can be faster than piloted craft or hit harder, but not both at the same time.*
      - *Requires multi-faction research; Republic and Empire can't develop on their own.*
      - *Guidance type heavily affects use cases.*
  - *Subcapital. Have an interstellar drive and perform one specific role.*
    - *Artillery. Attacks structures and capital ships.*
      - *Maximize range.*
      - *Maximize damage.*
      - *Minimize mass.*
    - *Defender. Protects an area from strike craft and missiles.*
      - *Maximize tracking speed.*
      - *Maximize one of the following:*
        - *Range*
        - *Agility*
    - *Destroyer. Attacks point defenders and artillery.*
      - *Multiple roles:*
        - *Sniper*
          - *Maximize range.*
          - *If needed, maximize rotational agility for fixed weapons.*
        - *Brawler*
          - *Maximize agility.*
          - *Maximize one of the following:*
            - *Armor*
            - *Damage*
    - *Light Carrier. Launches strike craft.*
      - *Minimize time to ingress for launched craft.*
      - *Minimize time to egress from combat zone.*
      - *Maximize hangar size.*
      - *In general you have no room to defend against enemy attack, so your only response is to run after dropping your ships and come back when the fight's over.*
  - *Capital ships. These all have point defense like a defender and do one extra role.*
    - *Siege Carrier. Launches strike craft and has point defense.*
      - *Maximize hangar size.*
      - *Minimize time to ingress for launched craft.*
    - *Dreadnought. Anti-subcapital like a destroyer and has point defense.*
      - *Doing both jobs makes the ship massive; it has to be a sniper.*
      - *Maximize range.*
      - *Maximize two of the following:*

        - *Agility*
        - *Armor*
        - *Damage*
    - *Being massive means slow dreads are vulnerable to artillery!*
  - *Classes that are probably bad ideas:*
    - *Combat Carrier. Anti-subcapital and launches strike craft.*
      - *Maximize agility. Anti-subcapital role means weapons are either short ranged or are fixed. Means you need to…*
      - *Minimize mass, which means you need to…*
      - *Minimize hangar size.*
      - *<u>Problem:</u> What happens when this needs to fight a defender and a destroyer at the same time? Or artillery and a defender?*
    - *Siege Vessel (come up with better name!). Anti-structure like artillery but has point defense.*
      - *<u>Problem:</u> Why have the ship come close enough that it needs point defense?*
- *Supercapital ships. Have point defense and do two extra roles.*
  - *Having all these parts generates lots of heat, so now they're easily detected and can't operate quickly. Two options:*
    - *Focus on the payload, which makes you horribly slow in realspace at least.*
    - *Focus on the thrust, which makes you even easier to see and costs even more in supplies.*
  - *Classes that are probably bad ideas:*
    - *Siege Supercarrier. Anti-structure with carrier capability.*
      - *<u>Problem:</u> Suddenly has a minimum operation range (the anti-structure weapon's max range).*
      - *<u>Problem:</u> Anti-structure means it's not agile, but at the same time you're reducing range???*
    - *Combat Supercarrier. Anti-subcapital with carrier capability.*
      - *<u>Problem:</u> Everything with a siege supercarrier, and now the primary weapons probably have bad tracking since the ship's so big.*
- *Stations*
  - *Defense towers*
    - *Being immobile means these things can have insanely high damage and ranges. If hooked up to an asteroid it has a large heatsink to dump into to make it even bigger, **so asteroids may be a strategic resource.***
    - *Weapon limits are mostly down to tracking speed.*
  - *Barracks/Hangars*
    - *Recruitment of troops and storage of military equipment.*
  - *Strategy centers*
    - *Remote command of ships and fleets.*

Weaponry

Friday, September 9, 2016 5:17 PM

There's definitely going to be:

- *Lasers*
  - *Good range.*
  - *At combat ranges, is a hitscan weapon.*
  - *Attenuates with distance!*
- *Missiles*
  - *Great, possibly best range…*
  - *But the delta-V is a problem.*
  - *Can carry all sorts of payloads, so it could be good for armor, for shields if that's a thing, or for EW.*
  - *Where it gets sensor data is also important, since if it's all onboard it may have trouble in a crowded environment.*
  - *Targets can also warp away if the missile takes too long!*
  - *Extra mass to the ship, so it may not get a lot of ammo at small sizes; big ships don't need to worry so much. How does this affect balance?*
    - ***How does losing the mass of the missile affect combat?***
- *Railguns*
  - *Good range.*
  - *Many options for messing up armor.*
  - *Generates a lot of heat!*
  - *Rails take a beating launching payloads, may not have good fire rate.*
- *Warp Torpedos*
  - *Strike craft-sized missiles that can do in-system warp. These can be mounted as a secondary or used as the primary on a carrier hull, turning the thing into a mobile missile silo.*
  - *Two types: piloted and unpiloted.*
    - *Piloted torpedos are remoted to a pilot. Don't need a spotter, but you're limited to launching as many torpedos at a time as you have pilots.*
    - *Unpiloted torpedos have onboard SI and are guided to position by a spotter ship near the target. However you can launch as many as is practical for your ship to launch simultaneously. It's possible to launch without a spotter, but the torpedo will arrive at the very edge of battle and will probably be intercepted.*

There might be:

- *Ballistics*
  - *Not definite since it's not clear if their effective ranges are large enough for any type of combat.*
  - *But would have much better fire rate than a railgun.*

Equipment

Monday, September 19, 2016 6:47 AM

- *Hangar*
  - *Carries craft of the maximum specified size.*
    - *These do not have to be limited to strike craft; capital Imperial carriers can hold smaller subcapitals, for instance.*
  - *Have Launch Bays that take a certain amount of time to launch a single craft. This is mainly done so that carriers can't lag the game out by dumping their entire hangar in one go.*
    - *Each bay has a maximum size; if a ship of a smaller size launches from the bay it only takes up that many size units in the bay.*
    - All launch bays in the game have their launch time multiplied by a server variable $k_{sLaunch} = 1.0$ by default.

Travel

Friday, September 9, 2016 6:35 PM

Overview

- *Big ships tend to have range, little ships don't. The larger volume means larger ship classes usually have more fuel and larger FTL drives that can compress space more strongly.*
- *Bigger FTL drives compress space harder, but being able to do so reduces their ship's agility in-system due to the nearby masses. Smaller drives aren't affected so badly but they don't have the compressive power for interstellar travel.*
- *The more mass you have, the longer it takes to charge for warp. This is the big difference between ship classes, and you can't compensate with more smaller engines or anything like that.*
- *Fleets have the speed of their slowest ship!*
- *You can turn in warp (so you can intercept other people in warp).*
- *Because normal sensors don't work in warp, you need one crewman to act as Navigator if you want to do more than move from system to system, such as moving around in deep space or intercepting other fleets in warp.*
- *If your Navigator is incapacitated while you are in FTL and your destination is not a celestial object, you must drop out of warp to wait for them to wake up or to ready another Navigator.*
- *Interdiction fields are a thing; they have a far larger power draw than a warp field however and thus multiply the Supply cost while they're active.*
    - *You can't interdict and warp at the same time. You can, however, join another warp bubble and force it to stop by generating an interdiction field in it (this is how field interceptions normally work).*
    - *You can alternately have your Navigator try and tackle the enemy's Navigator. This is very dependent on their respective mental strength and MMIs.*
    - *You can't move while interdicting! You can rotate but that's it.*
- *When you leave warp your ship immediately loses all velocity.*

Specifications

- Basic values of an FTL drive:
    - Max Speed, $c_{ftlMaxSpeed}$. Must be positive.
    - Drive Size, $c_{ftlSize}$. Must be positive.
        - Affects warp agility. Increasing size decreases agility.
        - Affects scaled max speed near masses.
- This allows functions on the drive, given distance d from celestial mass m:
    - **Max in-system speed $s(d, m)$**
        - $s(d,m) = k_{speedScale} * c_{ftlMaxSpeed} * clamp(k_g \frac{d^2 c_{ftlSize}^2}{m_{ship} m}, 0,1)$
        - In deep space, asssume $s = k_{speedScale} * c_{ftlMaxSpeed}$
        - Where:
            - $k_g = 1$; it's just there as an adjusting constant for now.
            - $k_{speedScale}$ = a scaling factor to generate the final speed.
        - If $s(d,m) < 0.01$, you're considered locked and you must drop out into normal space.
        - Remarks: This needs to cut hard into a large ship's max speed when in system; k_g will likely need adjusting.
    - Acceleration $a(d,m)$
        - $a(d,m) = \frac{k_{accelScale} c_{ftlMaxAccel}}{c_{ftlSize}^{0.5}} * clamp(k_g \frac{d^2}{m_{ship} m}, 0, 1)$; in deep space assume this is $a = k_{accelScale} \frac{c_{ftlMaxAccel}}{c_{ftlSize}^{0.5}}$
    - Time $t(d,m)$ to charge the drive in seconds

- $t(d, m) = \max(c_{ftlSize}, k\frac{c_{ftlSize} m_{ship} m}{d^2})$
- If $t(d, m) > 20$, you're considered locked and you can't jump.

Detection

Tuesday, September 13, 2016 10:41 PM

- *One crewmember must act as the ship's Detector (come up with better name). Anything within a certain radius of the ship can then be detected, no matter if it's in FTL or not.*
    - *The accuracy and range of the Detector depends on their mental strength, their experience, and their MMI.*
- *If you don't have a conscious Detector, all you'll see is where a ship enters and leaves warp, as well as STL objects.*

Morale

Sunday, September 11, 2016 5:29 PM

- Represents the psychological capacity to fight for an entire fleet. This is normally at 100; if it goes over 100 you start getting bonuses, and if it goes under 100 you start getting penalties.
- If fleet morale hits zero in battle the fleet will retreat, no exceptions, and a zero-Morale fleet outside of battle can be convinced to defect.
- Increasing Morale:
  - Win battles.
    - $c = 10$
  - If Morale is under 100, destroy enemy ships.
    - $c = 1$, gain for one ship.
  - Dock at stations. The effect is greater the more comfortable the station is.
  - Rest at stations. The effect is greater the more comfortable the station is.
- Losing Morale:
  - Lose ships.
    - $c = 1$, loss for one ship.
  - Lose battles.
    - $c = 10$
- Bonuses:
- Penalties:
- Faction effects on Morale:
  - Republican ships…
    - AI equipment does not have Morale. This means they can't suffer negative effects but can't get bonuses either.
    - AI equipment can still be Suppressed or Shocked.
    - Get a boost to starting Morale when they are defending an inhabited station/planet.
      - $k = 1.5$
    - Get a boost to starting Morale when they are outnumbered.
      - $k = 1.25$
    - Morale loss from **manned** ship losses is higher than normal.
      - $k = 1.25$
  - Empire ships…
    - AI equipment does have Morale, but it is much harder to degrade compared to human morale. It is also a bit easier to raise compared to human morale, especially if given a War motivation.
      - $k_{loss} = 0.5, k_{gain} = 1.25$
    - Get a boost to starting Morale when they are led by a noble.
      - $k = 1.125$
    - Get a boost to Morale when they have won a battle. This effect additively stacks to a limit.
      - Boost is additive: $c = 10$, with the maximum boost being $c_{max} = 50$.
      - This boost decays the longer you go without a victorious fight.
        - Specifically the boost $c(t) = \frac{c(0)}{2^{t/2}}$, where t is the time since the last victory and $c(0)$ is the boost immediately after that victory.
      - This boost is removed if you retreat from a fight, but not if you lose a fight.
- Specification
  - Organic crewmembers take 100% of a morale gain/loss.
  - AI crewmembers take 50% of a morale loss and 125% of a morale gain.
  - Thus the ship morale/loss factors can be calculated as
    - $k_{moraleLoss} = \left(n_{organic} + 0.5n_{AI}\right)/n_{total}$

- $k_{moraleGain} = \left(n_{organic} + 1.25n_{AI}\right)/n_{total}$
- Where:
  - $n_{organic}$ = number of organic crewmembers
  - n_{AI} = number of AI crewmembers
  - $n_{total} = n_{organic} + n_{AI}$

- To calculate ship morale/loss $moraleChange_{final}$, let $moraleChange_{raw}$ be the unmodified morale change. Then do:
  - If $moraleChange_{raw} > 0$
    - $moraleChange_{final} = k_{moraleLoss} * moraleChange_{raw}$
  - Else
    - $moraleChange_{final} = k_{moraleGain} * moraleChange_{raw}$
  - This could be made into a continuous function, but the edges near 0 would be seriously messed up. If branching is costly, sort on the morale change events and do the losses separate from the gains.

Command Ships and Crew

Thursday, September 15, 2016 10:50 PM

- *Each fleet is led by a Command Ship (CS); in the player's case this is the ship they directly control.*
- *CSs have an inventory shared by the whole fleet.*
- *CSs can customize the weapons they mount.*
    - *This is a slot system consisting of weapon size, type and fixture; see [Design:General:Weaponry].*
- *Most importantly, CSs have specific roles that must be filled by crewmembers. This includes your character, who always has full Morale.*
    - *The roles are:*
        - Captain
            - *Issues commands to the fleet. Special role; taking this role doesn't prevent you from taking another role.*
            - *Adds bonuses to fleet morale. Can add bonuses to recruitment costs.*
        - Navigator/Detector
            - *Handles FTL sensors and (usually) pilots the ship when in warp.*
            - *Adds bonuses to FTL acceleration, FTL sensor range and FTL sensor accuracy.*
        - Pilot
            - *Pilots the ship in realspace.*
            - *Can add bonuses to realspace acceleration, inertia dampening, and sensor signature (since more efficient maneuvers generate less heat).*
        - Gunnery Officer
            - *Manages fire control and weapon-related engineering.*
            - *Can add bonuses to weapon fire rate, accuracy, laser damage and sensor signature.*
        - Engineering Chief
            - *Manages defensive systems and maintains ship internals.*
            - *Can add bonuses to shield strength, FTL speed, realspace acceleration, and sensor signature.*
        - Intelligence Officer
            - *Analyzes sensor data to figure out what's happening on the battlefield/strategic theatre.*
            - *Can add bonuses to FTL and standard sensor accuracy, and supply cost.*
    - *On subcapitals the non-Captain roles can be consolidated to Navigator/Detector, Pilot, Systems Officer (who handles Gunnery and Engineering and can provide all of their bonuses), Intelligence Officer.*
        - *Republic subcaps can bring it down to Navigator/Detector, Pilot, Systems Officer.*
    - *Strike craft have at most a Pilot and a Systems Officer, since they don't have interstellar FTL; the pilot takes up navigation while the Systems Officer handles detection, but they don't get N/D bonuses. The smallest only have a Pilot, and the ship systems automate everything beyond FTL and flight control.*
    - *Crewmembers have properties:*
        - *Skill at a given role*
        - *Morale (there are bonuses for high and low morale)*
        - *Equipped MMI, if any*
            - *This boosts their role skill according to the skill's growth function.*
- *What happens when the CS goes down?*

Logistics

Sunday, September 11, 2016 5:08 PM

Supplies

- *Fleets run on Supplies, which represents all the food, fuel, and armaments that get used up. If you don't have enough Supplies to fuel your ships, you lose ships until you do.*
  - *The lost ships can't be picked back up, they are immediately removed from the fleet.*
- *Factors affecting Supply cost rate:*
  - *Warp*
    - *More range = more Supplies required.*
    - *More max speed = more Supplies required.*
  - *Weapons*
    - *Normally is just part of the Supplies draw. Special ammunition can be bought for your command ship.*
    - *More weapons = more Supplies required. Should the weapon use in battle be recorded or just assumed?*
  - *Crew*
    - *More crew = more Supplies required.*
    - *However, AI do not need supplies on their own. AI/SI count should be separate from crew count.*
    - *Bigger ships usually require more crew to maintain all of the different modules.*
      - *Republican ships…*
        - *Crew counts are lower than normal since SI is everywhere.*
      - *Empire ships…*
        - *Crew counts are higher compared to a Republican ship.*

Crew

- These are the minimum number of people and *AI* that must be aboard a ship in order for it to run. You don't need to pick them up or train them yourself, they come with the ship. However they affect how a particular ship type consumes Supplies and how its morale works.
- SI is *not* counted as crew since it's not sapient; if a ship is all SI and has no organic or AI crew, it is counted as having 0/0 crew.
- Supplies
  - Let $c_{supplyCrew}$ be the cost to supply the entire crew:
    - $c_{supplyCrew} = n_{organic}$
    - Where:
      - $n_{organic}$ = number of organic crewmembers
- Morale
  - See [Design:Fleet Mechanics:Morale].
- Recruitment Cost
  - Crew applies a multiplicative factor to a ship's recruitment cost; this is calculated as
    - $k_{crew} = \left(1.125n_{organic} + 2n_{AI}\right)/n_{total}^{0.5}$
    - If $n_{total} < 1, k_{crew} = 1$
    - This is then multiplied on the ship cost so it can scale with size.

Weapons

- Each weapon affects supply cost as a factor:
  - $k_{weapons} = [k_{laser}c(T_{laser}) + k_{railgun}c(T_{railgun}) + k_{missile}c(T_{missile})]/c_{total}$
  - If $c_{total} \leq 0, k_{weapons} = 1$
  - Where:
    - $c_{total} = c(T_{laser}) + c\left(T_{railgun}\right) + c(T_{missile})$
- The subtotal $c(T)$ for a weapon type T is

- $c(T) = k(T, M_{turret})n_{turrets} + k(T, M_{fixed})n_{fixed}$
- Where:
  - $k(T, M)$ = the cost factor for a weapon mount of type M that carries a weapon of type T

- Supply Cost
  - $c_{supplyWeapons} = k_{weapons}c_{total}$

FTL

- Values defined in [Design:FTL:Travel].
- Supply factor, $k_{FTL} = 1.2^{c_{ftlMaxSpeed}c_{ftlSize}}$
- Supply cost, $c_{supplyFTL}(t, d, m) = k_{FTL}c_{ftlMaxSpeed}c_{ftlSize}$
  - Where:
    - t is the unit of time over which the supplies are being used
    - d is the distance from the most affecting celestial in system
    - m is the mass of said celestial

Ships

- Have weapons.
- Also have a tech level to indicate the overall complexity of the hull and the equipment it carries.
  - The tech level is an integer, $n_{techLevel} \in [1, +\infty]$.
- Recruitment Cost
  - Cost for a single ship is calculated as
    - $c_{recruitTotal} = m_{ship}\left(k_{FTL}^{b_{hasFTL}}\right)k_{crew}k_{weapons}k_{techLevel}$
    - Where:
      - m_{ship} = the mass of the ship
      - $b_{hasFTL} =$ 1 if the ship has a FTL drive, 0 otherwise
      - $k_{techLevel} = 1.5^{n_{techLevel}-1}$
- Supply Cost
  - Supply for a single ship is calculated as
    - $c_{supplyTotal}(t, d, m) = c_{supplyCrew} + c_{supplyWeapons} + c_{supplyFTL}(t, d, m)$
    - Where:
      - t is the unit of time over which the supplies are being used
      - d is the distance from the most affecting celestial in system
      - m is the mass of said celestial
- Ships in another ship's hangar still cost supply, but at a reduced rate. Typically $k = 0.5$.

Fleet Ships and Crew

Monday, September 19, 2016 6:39 AM

- *Fleet ships have much simpler crew division:*
  - Staff *that man the ship itself.*
  - Craft Pilots *that control craft launched by the ship.*
- *Craft launched from a ship's hangar count against the number and type of Craft Pilots in the ship's roster. If there aren't enough pilots of the right type, you can't launch more craft.*
  - *This is why SI ships count as having 0 crew, to simplify the launch algorithm.*
  - *They also take up space in a Launch Bay while they take off; if all Bays are in use the next craft has to wait.*

Dashboard

Wednesday, September 21, 2016 6:26 PM

- *World landmarks split up into:*
  -
  - *Domains (**y** ly-radius regions of houses)*
    - *Houses (**x** ly-radius regions of stars)*
      - *Stars*
        - *Orbitals (planets)*
          - *Orbitals can have an arbitrary number of nested orbitals (moons, moons of moons, etc.)*
            - *Only orbitals have structures, however.*
          - *Orbitals can also have rings.*
        - *Rings (asteroid belts, rings around a gas giant).*
          - ***Rings can have orbitals inside them (like an especially large asteroid), but rings can't have nested rings.***

- *A site is any temporary point of interest on the strategic map.*
- *Types of sites:*
  - *Battles*
  - *Salvage Sites*

Battles

Wednesday, September 21, 2016 6:28 PM

- *Always between fleets; if it's an orbital siege it's between the attacker and the orbital's defense force.*
- Involves two sides, reinforcements have to decide which side they're joining.
- *If you reach a certain radius from the fight your fleet retreats.*
- If all ships of one side are destroyed or retreating, the other side wins.
- If all ships of both sides are destroyed or retreating, it's a draw.
- *Loot*
    - *Surrendered ships*
        - *Retreating ships have a chance to be captured if their morale was zero by the time victory is declared; this is different from salvaging, as the crews are still aboard. You might not want to pick them up anyway since their morale will be low.*
        - To surrender, a ship must:
            - Be retreating
            - Be hostile
            - Be manned
            - Have 0 Morale at the time of victory.
        - Any surrendered ships you don't recruit disappear from the map.
    - *Any bailed-out hulks will also be part of the loot. They can come from any side.*
    - *Wreckage has a chance to be generated from destroyed ships.*
        - *What does Wreckage do?*
        - *Are these usable as/convertible to Supplies?*
    - *Destroying or capturing an enemy command ship puts its loot and equipment on the loot table. If the command ship was destroyed, only some of its inventory will drop.*

Salvaging

Wednesday, September 21, 2016 6:01 PM

- *How to Salvage*
  - *Make Target Bailout*
    - *Requires two things:*
      - *Target Morale is zero*
      - *Target is still taking damage*
    - *Weapons that generate heat in particular are good to cause a bailout, and there may be short-range disruptor weapons that do garbage damage but greatly increase bailout chance.*
    - *Then the pilot bails out…*
      - *Unmanned ships just have the pilot disconnect/the SI melts.*
      - *Crewed ships eject and the escape pods warp out.*
        - *Can you capture the escape pods???*
  - *Capture the Target Ship*
    - *Need to send a control unit to the target. This places it under rudimentary control, but the ship won't move or fight nearly as well as it should; it can do in-system FTL well enough though. Send the ship to a dock to get it repaired and ready for field use.*
    - *All ships can move up to an abandoned ship and capture, so you can order a member of your fleet to do the capture; alternately you can fit a launcher that will let you do the capture at range, but this takes up a weapon slot.*
- *Salvage Sites*
  - *After a battle, anything you don't take with your fleet/send to get refurbished can be left in a Salvage Site that others can enter.*
  - *For the most part you can't see a Site unless it's of high enough value to you, and* there's a 50% chance that a battle won't generate a Salvage Site at all. You also can't see Salvage Sites generated by battles you were involved in.
    - This doesn't mean you can't enter them if someone else decides to stop by.
  - *When you enter a Site, you can choose to stay and salvage*; this acts like resting and at the end of it you get some percentage of loot at the site, displayed as a post-battle loot table. You will always get at least one item that is not Wreckage if it exists.
  - *Other fleets can enter this site and try to salvage the hulks or gather wreckage.*
    - *If they think they can take you, they'll demand you leave, then fight you if you don't.*
    - *If they think you're too strong but not strong enough to be a problem, they'll stay and reduce the amount of salvage you get. You can choose to attack them.*
    - *This can lead to police coming by but probably won't, given that this was generated by a battle.*

Dashboard

Saturday, September 24, 2016 11:56 PM

- *The different factions exist as actual, dynamic entities. They have goals that can change with world events (like a war breaking out/ending), deploy operations based on those goals, and maintain territory as needed.*

Goals

- *Each faction has a set of goals that their AI uses to decide their Stance. Their Stance is then used by advisor AI to decide what events need to occur (get Total Supplies to 1 million by Year 30, for example), and that is used by regional AI to decide what specific operations should occur where.*
- *This can go the other way - a region may report that they need supplies and the advisors can decide if the region is important enough to make the request a goal.*
    - ***Refusing requests may or may not have consequences, this hasn't been decided.***

Operations

- *Eventually the advisor goals translate into a specific action for a fleet/group of fleets to do. This is called an operation.*
- All operations have:
    - A list of fleets assigned to the op. You can add a fleet, but you can't remove a fleet once it's been added.
    - A start time for the op. The operation can't start unless it has fleets assigned.
    - An end time for the op; if the operation isn't completed/failed by this time, it is reported as a timeout instead.
    - An operation type, for the sake of sorting
    - A status value:
        - Unassigned (created but no fleets are assigned)
        - Assigned (created, has fleets, but isn't started)
        - In Progress (created and started)
        - Completed (all objectives completed)
        - Failed (sufficient objectives failed to cause the op to fail)
        - Timed Out (somehow did not complete or fail; possibly nobody was assigned for too long)
- When a fleet is assigned to an operation, it becomes one of the fleet's objectives. The player sees it in their mission list, fleet AIs refer to their list during their high-level decision loop.
- Performing operations generates world events*, and finding out someone's assigned to an op might generate world events if word gets out.*

Territory and Logistics

- *Celestials are the geography, structures are how factions claim geography as territory. By placing a Com Tower in a system a faction claims a star, and by capturing towers they expand their territory. Destroying towers renders a system neutral, basically the space version of razing a town.*
- *Factions can contain vassal factions; a star is owned by the faction's topmost owner (the first faction in the faction tree that isn't itself a vassal). However it's administered by whatever faction actually built the tower.*
- *Structures can notify their faction that they need supplies, which can lead to supply operations being generated as mentioned in the previous section.*

Diplomacy

- *Factions do not necessarily want to dominate the world (most don't, expansionists still prioritize world safety over control); they have standings with other factions that are changed by world events and operations. Standing changes are themselves world events to help the AI's decision-making.*

- *CKII bases everything off Opinion and shows the player what is altering an agent's Opinion; look into that.*
- ***There may be special ops that just increase the standings of everyone involved, like a feast or a festival.***

Overview

Friday, September 23, 2016 7:31 PM

- *Everything exists in the game world, and everything happens in the game world.*
- *The game world is the data source or collection of sources that describes the state of all elements simulated by the game.*
- *Elements relevant to the game world include:*
  - *Physical. Have positions in some kind of space.*
    - *Celestial objects, entities and the like.*
  - *Logical. Are conceptually distinct from each other.*
    - *Factions and the properties of each faction.*
  - *Historical. Are chronologically distinct.*
    - *World events. These provide a way to deterministically describe everything that's happened, allowing for replays among many other things.*
- *Each element's in-memory and on-disk representation may be different, and the frequency they get moved to/from disk will definitely be different. Site entities aren't going to get pushed to disk ever until downtime, factions may get pushed to disk every hour.*
- *It's especially important to compartmentalize all of this, since the server is a bunch of distributed nodes. Avoid shared state wherever possible, then the shutdown step's merge is a relatively simple operation.*
- *Avoid hitting to disk wherever possible because it is super slow even with an SSD.*

Table of Contents

Friday, September 23, 2016 7:48 PM

Summary

Executable/set of executables that act as model for the game world.

Table of Contents

- Data Definitions
  - Description of all data that gets saved to disk as files eventually.
- Gameplay
  - Game mechanics.

Design Questions

Friday, September 23, 2016 8:14 PM

- *How do we test this server?*
  - *In particular, if we have multiplayer how do we stress test high user count?*
    - *Test account generation*
    - *Automated login and activity*
      - *Means creating an automated client, what does that entail?*
  - *Each node type needs unit tests.*
- ***How do we implement local nodes efficiently?***
  - ***What is the most efficient way to dummy IPC for local mode?***
- *How do we take advantage of all hardware on the node's machine?*
  - *A node *might* have a GPU, but we can't make that assumption. Unless we're on a local mode server in which case we're in single player and need to assume there's a GPU.*
  - ***Is GPU computing worth doing?***
    - ***If so, how do we platform-independently check and use the GPU for general computing?***
  - ***How do we coordinate nodes?***
  - ***Do we dynamically-balance load?***
    - ***If so, what balancing architecture do we use?***
- ***What is the update cycle for the final server layout?***

Description

Friday, September 23, 2016 7:14 PM

- *Is the game world's model; all data and mechanics are handled by this server.*
- *Composed of nodes that perform one specific role; these should be designed with the assumption that they're on separate processes and possibly separate machines, although local versions can be implemented with threads and their IPC shortened considerably.*

Node Hierarchy

- *A login node goes in front; this validates that the user exists and has credentials to play on the server. Passes off to connection nodes.*
- *Connection nodes act as a demux for processing nodes. Users only ever touch these nodes, they don't get deeper than this.*
    - *(In other words your single-player client can't get any deeper than this either.)*
- *Processing nodes perform the actual world update and communicate state updates back to connection. This is hugely, hugely complicated and there are many different architectures.*
    - *The world is logically split up into a giant graph where each node is a tree though, so it lends itself to a responsibility-oriented design. Several clear roles:*
        - *Site node, for tactical-level activity in one system. Might be split into:*
            - *Physics node*
            - *Ship AI node*
            - *Fleet AI node*
        - *System node, for handling activity in a system that' s not individual ships flying around.*
        - *Faction AI node, to update each faction's campaign-level decision making.*

Milestones

Friday, September 23, 2016 8:15 PM

- Startup
  - Open step
    - Server can accept connections, they just can't enter the game world until the server says so.
  - Start step
    - Load assets
    - If a world file exists:
      - Load world file
    - Once start step is complete, it notifies any connected clients that they can start playing.
- Load from world file
  - Validate the world file.
    - If the result file is corrupt or couldn't be read:
      - Report load failure
      - Stop the load
    - **How do we report a corrupt world?**
  - (distributed mode) Split the world file out to the nodes.
    - If a node reports load failure:
      - Report load failure
      - Stop the load, which means sending a drop command to all nodes.
- Save to world file
  - (distributed mode) Merge the node states into one world file.
    - If a node can't send:
      - Report that node's failure
      - Keep taking chunks
      - **Save the chunks as a "broken changes" dump**
      - Report overall save failure
      - Stop the save
  - Validate the world file we got.
    - If the result file is corrupt or couldn't be written:
      - Do not save changes
      - Report save failure
      - Stop the save
    - **Do we want to save the results as a "broken changes" dump that the World Manager could look at?**
- Handle clients:
  - Connection
    - If connection is blacklisted (DDoS IP, banned IP, etc.):
      - Reject connection
    - Check client's version; if version is under server minimum:
      - Reject client
    - Authenticate the client. If authentication fails or user is banned:
      - Reject client
    - Setup client's presence on server. If this fails:
      - Politely reject client
    - Notify client/subscribing dashboards that the client is in the game
  - Client Request Execution (world update)
    - *This is literally all of gameplay, so it needs its own document. Requests aren't exactly*

commands to do one thing or another so much as they are encapsulated input. The server then applies that input like a player pressing buttons on a controller; that changes world state.

- World State Broadcast
  - *The world state changes generate events that the server can send to the clients; the clients can then use those events to update their display of the game world.*
  - *The question is how we send these events, since they're probably very high volume. This ties heavily into your netcode implementation.*
- Disconnection
  - Notify subscribing dashboards that the client has left

- Shutdown
  - Disconnect all clients
    - Dashboard should remain unaffected, but it naturally can't see into the world after this step.
  - Stop step
    - Save the world file
    - Unload assets
    - Server cannot accept new connections.
  - Closed step
    - *The game is inactive but the server is still running. Sort of like a main menu for the server.*
    - Server cannot accept connections.
      - *Is Dashboard an exception?*
  - Quit server program

Table of Contents

Friday, September 23, 2016 8:00 PM

Summary

Executable that connects to the Game Server to play the game; acts as view/controller for the game world.

Table of Contents

- Interface and Experience
  - What interface devices should be supported?
  - How is the user interface laid out?
  - How is the user experience laid out?
  - How do we make the client accessible?

Design Questions

Friday, September 23, 2016 8:20 PM

- *Should we preload a local server so people can go in and play faster?*
    - *Adds to loading time, after all.*
- ***Should server nodes be able to know if a client node is on the same machine (and dial down GPU draw accordingly)?***
- ***How do we predict and interpolate state?***

Dashboard

Thursday, September 22, 2016 4:15 PM

- *Strike craft need to be completely operated by HOTAS. There's no question here.*
- *Subcapitals have all of their different roles and a player has to be able to quickly switch between them since they're mobile, medium-range elements of a battle; is this possible on a HOTAS alone? Elite has its menus but the mental lag will add up fast.*
- *Supercapitals don't need to worry so much about twitch, but they do have multiple systems. At this point there's not a lot of need to have it all on HOTAS but the player may switch to a class that does.*
- *Commanding large numbers of units traditionally requires mouse and keyboard, nobody's come up with an interface that would work completely on HOTAS, never mind one that would work well. How do we solve this? Or should we go the other way - figure out how to have the units command themselves and make the player more of a guide?*

Dashboard

Thursday, September 22, 2016 4:21 PM

- *Make the game visually accessible? Is this even possible?*
  - *Implies that all important state needs sound cues:*
    - *Your ship*
      - *Velocity*
      - *Health*
      - *Weapons*
        - *On target/displacement from target*
        - *Ammo*
    - *Other ships*
      - *IFF*
      - *Position*
      - *Velocity (does position imply velocity?)*
      - *Health*
      - *Are they attacking (or give ordinance sound cues?)*
    - *Mission objectives*
      - *Objective succeeding/failing*
      - *Objective about to fail*
  - *A lot of those properties could be encapsulated by one sound and the distortions on the sound (sound type indicates IFF, binaural for position, pitch for health?)*
  - *How would you handle large numbers of entities in the space?*
  - *How would you handle collision warnings?*

Description

Friday, September 23, 2016 7:14 PM

- *A big view/controller for the game world. Does the following cycle:*
  - *Takes server requests*
  - *Displays state*
  - *Takes user input*
  - *Turns user input into server requests*
  - *Sends server requests*
- *Because of this it doesn't need physics or any world update info, but it does need to do rendering, local input and remote I/O very efficiently.*

Milestones

Friday, September 23, 2016 8:18 PM

- Login
  - Server can reject you at any time, if it does report rejection reason.
  - If connection was accepted:
    - If server is ready to play:
      - Enter game
    - Otherwise, if this is multiplayer:
      - Wait at the main menu
    - Otherwise, the game's broken.
      - Assert failure
      - Put error text on main menu that the game's busted!
- Game loop
  - Update view with new server state
    - Update scene tables as needed
  - World rendering
    - Filter scene tables into draw calls
    - Batch the draw calls
  - Handle input
    - Ideally platform independent, but on Windows at least you're notified of new raw input state.
    - Input module turns this into uniform input ranges.
  - Turn input into server requests
    - This is basically a lookup table.
  - Post requests to server
    - *Implementation heavily dependent on netcode library architecture.*
- Logout
  - Tell server you're leaving
  - Go back to main menu
- Single player-specific functions
  - Save/Load (another kind of request to single player servers, all multiplayer servers ignore this)

Table of Contents

Friday, September 23, 2016 8:02 PM

Summary

Auxiliary programs used to manage the client and server. **Remember that these do not have to be written in Rust!**

Table of Contents

- Artifact - Server Dashboard. Used to view and control a server.
- Artifact - World Manager. Used to view and find state errors in a saved world state file. Also used to load state from backups or rollback state.

Design Questions

Friday, September 23, 2016 8:45 PM

Description

Friday, September 23, 2016 7:16 PM

Table of Contents

Friday, September 23, 2016 8:09 PM

Summary

Tool program used to administrate a server without running the full client. Reports server status and allows top-level control such as starting/stopping the server remotely.

Table of Contents

- Accessibility
  - How do we make the server dashboard accessible in particular?

Unanswered Questions

Friday, September 23, 2016 8:11 PM

- Local control is OK, do we want to do remote control where the dashboard isn't on a machine with a server node at all?
- How do we coordinate the nodes? Especially how do we tell them to shut down?

Description

Friday, September 23, 2016 8:23 PM

- Auxiliary tool used to control a server.
- Shouldn't have any 3d interface, else you could just log in as an admin via the client.
    - **Goofy, but should this have a CLI since it doesn't need rendering? Would make it easy to remote.**

Milestones

Friday, September 23, 2016 8:25 PM

- View server state
  - Node status
  - Connection status
- Control server
  - Startup/Shutdown
    - *Remarks*
      - *Implicitly an order of start/stop to all of the nodes.*
      - *Players need to be notified that their node's going down.*
        - *Can they be warned in advance?*
      - ***Can startup/shutdown be scheduled?***
    - Startup
      - Notification of startup
      - Data load. Servers are distributed, so data load must be distributed too.
      - Mark server as up
    - Shutdown
      - Notification of impending shutdown
      - Distributed data save
      - Mark server as down
  - Event handling
    - Logging events. Needs to be quick.
      - **How do we buffer I/O so logging doesn't clog up everything else?**
      - **How do we dump when an error occurs?**
    - Reporting events. The admin may not actually be at the machine or awake.
      - Email?
      - IRC?
      - **If in local mode, can we post to client?**
  - Manipulate world elements
    - Create an object
    - Destroy an object

Table of Contents

Friday, September 23, 2016 8:09 PM

Summary

Tool program used to analyze and repair world files.

Table of Contents

- Accessibility
  - How do we make the world manager accessible?

Description

Friday, September 23, 2016 8:25 PM

Milestones

Friday, September 23, 2016 8:25 PM

- Load world files
- Save world files
- View world files
    - Search for specific elements?
- Validate world files
    - List all errors and invalid state.
- Sync from backup?
- Rollback?
    - Implies world events can be rolled back and that this doesn't require actually running the server.

Table of Contents

Tuesday, September 27, 2016 2:09 PM

Summary

The audio/visual assets used by the game.

Table of Contents

Dashboard

Tuesday, September 27, 2016 2:10 PM

- A lot of work goes into DSP mixing; look at Elite's sound design.
- Need following tools:
  - Audio production software (professional grade is Vegas)
  - DSP effects vital to dynamic audio (Elite uses WWise).
  - Middleware like WWise also handles audio asset loading, although you probably have something else in mind for that.

Dashboard

Tuesday, September 27, 2016 2:10 PM

- You need:
  - Models
    - They don't *need* to be animated, but support would be very appreciated.
  - UI
  - Textures
    - Color
    - Normal?
- Implies you need following tools:
  - 3D:
    - Final modeling (Blender, etc.)
    - Sculpting, if possible (Zbrush, etc.)
      - Sculpting is preferred since it lends much better to mocking up the topology and textures; finishing tools can then derive final assets from concepts the artists draft in the sculpting app.
  - 2D:
    - Texture finishing (Photoshop, GIMP, etc.)
- PBR is way way off in the stratosphere; just get some meshes drawn with standard shading methods.

Testing the Products

Friday, September 23, 2016 5:55 PM

Each product has specific elements that need testing:

Game

The client and server have these tests in common:

- *100% CPU load*
- *100% GPU load*
- *Low memory*
- *Power throttling (user is probably on a laptop)*
  - *Weak GPU suddenly switches in*
  - *CPU suddenly drops in power*

Game Server

- *Unit tests: interface*
- *Unit tests: functionality*
- *Stress tests:*
  - *Common tests, see above*
  - *Multiplayer only:*
    - *High load*
    - *Unbalanced load*
    - *Anomalous conditions*
      - *DDoS on login node*
      - *DDoS on connection nodes*
    - *Cheating*
      - ***Going to need outside help on this one, the point of hacking is that it's about what the programmer didn't expect.***

Game Client

- *Unit tests: functionality for each node.*
- *Integration test for whole server.*
- *Stress tests:*
  - *Common tests, see above*
  - *No GPU*
    - *Is probably a screen that says "GET A GRAPHICS CARD"*
  - *Multiplayer only:*
    - *Bad connection*
    - *Intermittent connection*

Game Tools

- *Unit tests: interface*
- *Unit tests: functionality*
- *Each tool should specify the tests it needs.*

Engine

Engine Modules

- *Unit tests: functionality for each module. Not necessarily split by output library, but may naturally end up splitting that way.*
- *Integration test for the whole engine.*

Engine Tools

- *Unit tests: interface*
- *Unit tests: functionality*
- *Each tool should specify the tests it needs.*

Making Packages Installable

Friday, September 23, 2016 5:55 PM

- ***Means picking a build system.*** *This takes the package artifacts and puts them in a platform-specific installer. Seeing as this must be automated the old VBscripts won't cut it.*
  - Each package has an installer and a delta patch from the previous release to the current release.
- *Also means knowing* ***what installer system to use for each platform****.*
  - *Windows of course prefers MSIs.*
    - *Best practices are on MSDN. Notably MSI can and should be built such that they can be run from the CLI.*
    - ***How do you test this?***
      - Automated validators: http://releaseengineering.blogspot.com/2008/03/automated-install-testing-msi.html
  - *OSX does DMGs and more rarely installer executables.*
    - *DMGs seem to be preferred whenever a product can be delivered in one .app. The game package is more naturally two .apps, but it can still work.*
    - ***People also expect the app to offer to move to Applications if it's running inside a DMG.***
    - ***How do you test this?***
  - *Linux uses a distro-specific packaging system. This is either RPM (Red Hat/Fedora/Centos) or APT (Debian/Ubuntu). Honestly you might just not support Linux for the time being if it seems to be a headache.*
    - ***How do you test this?***
- Common Testing Scenarios
  - Installation Testing – Test the installation with all possible combinations of application features. Test all types of installation, including Administrative Installation, Rollback Installation, and Installation–On–Demand. Try all possible methods of installation, including clicking on the .msi file, command line options, and installing from the control panel. Test that the package can be installed by users in all possible privilege contexts. Try installing the package after it has been deployed by all possible methods. Enable Windows Installer Logging for each test and resolve all errors found in the installer log and event log.
  - User Interface Testing – Test the package when installed with all possible user interface levels. Test the package installed with no user interface and with all information provided through the user interface. Ensure the accessibility of the user interface and that the user interface functions as expected for different screen resolutions and font sizes.
  - Servicing and Repair Testing – Test that the package can handle Patching and Upgrades delivered by a Small Update, Minor Upgrade, and Major Upgrades (MSI-specific functionality). Before deploying the package, write a trial update of each type and try applying it to the original package.
  - Uninstall Testing – Verify that when the package is removed it leaves no useless parts of itself behind on the user's computer and that only information belonging to the package has been removed. Reboot the test computer after uninstalling the package and verify the integrity of common system tools and other standard applications. Test that the package can be removed by users in all possible privilege contexts. Test all methods to remove the package, click the .msi file, try the command line options, and try removing the package from the control panel. Enable Windows Installer Logging for each test and resolve all errors found in the installer log and event log.
  - Product Functionality Testing – Ensure that the application functions as expected after the installation, repair, or removal of the package.

Updating Packages

Friday, September 23, 2016 6:03 PM

- *Hugely intricate process.*
- *How do we do this in a space-efficient way? Patches need to be small, after all.*
- *How do we handle add-ons?*
- *How do we handle server customization? Clients should be able to connect and automatically download the add-ons they need.*
    - *If there's central repositories, they may be faster to pull from than the server. Should this be a situation we handle?*
    - ***Does this mean server code must be forward compatible?***
- *Do we want the ability to:*
    - *Check for updates?*
        - *Means having a central server. Central servers can be DDoSed…*
    - *Automatically pull official updates?*
        - *Means having everything needed to check for updates.*
        - *Means knowing what patches to pull given the user's version. If they're really far behind, do they need to pick everything leading up or can they just pick the latest?*
            - *Or do we handle that behind the scenes?*
        - *Need an automated way to run the patches. Rollback and notify the user if patch update failed for some reason.*
- *Should there be a difference between purely gameplay patches and infrastructure patches (as in the user can choose what balance they want)?*
    - *If gameplay features gets added this becomes kind of a pain…*
- *If an update messes up, how do we roll it back?*
    - ***Usually the installer of our choice has rollback systems. Each one needs to be understood.***
- *Zero-reboot patching - this is actually a doable thing if you use dynamic libraries.*
    - *See kpatch and kgraft; kgraft waits for old threads to finish then patches, kpatch sleeps the old threads then patches (so the system freezes for a second while the patch goes through). Kgraft is truly zero-downtime, but kpatch is faster.*
    - *Crazy but possible - an example would be upgrading a server, though clients would have to leave until the upgrade is done. The server unloads the world, swaps the changed engine modules, then loads the world back in.*

Add-Ons

Friday, September 23, 2016 6:03 PM

- *Modifications to the game are done by extending the existing installation.* ***Mods don't get to delete or modify anything from the core game****; instead you change settings or specify overrides.*
- *Modifications to behavior are done through script files.* ***The script engine must not be overridable to prevent arbitrary code execution.***
- *This implies some kind of package system for add-ons, something supported both by client and server. In other words this is engine code.*
- An add-on pack consists of the following:
    - The add-on's version
    - The add-on's overrides
    - The add-on's assets, files that aren't allowed to be executed
    - The add-on's scripts
    - Optionally a list of dependencies?
        - *It'll be hard work debugging that, but modders would absolutely appreciate it.*
    - Optionally a validation certificate?
        - *How would we make this worth using at all?*
- Applying an add-on then consists of these steps:
    - Check if overrides require client/server stop
        - If so, check that user is allowed to stop the client/server
            - If **not**, reject apply request
    - Load assets
    - Load scripts (don't execute anything yet)
    - Check if overrides require client/server stop
        - If so, stop the client/server
    - Apply overrides, switching to add-on assets as needed
    - Check if the client/server was stopped
        - If so, start the client/server
- *Can we have add-on repositories? If so, how do we validate those?*

Between Products

Monday, September 26, 2016 2:37 PM

- Not all products will be Rust, so we need a common language to speak with. Rust can export to C as a static library ('crate-type = ["staticlib"]') or a dynamic library ('crate-type = ["cdylib"]', Rust 1.11+); it won't necessarily generate headers for you though.
    - Bindgen (https://github.com/Yamakaky/rust-bindgen) performs this. It should also generate bindings for most managed languages you'd think of using.

Between Platforms

Monday, September 26, 2016 3:16 PM

- The biggest hurdle to multiplayer is the fact that **floating point values are not portable**. No matter what we do there's no guarantee for this, we can at most design everything around carefully packing floats to hopefully stay consistent across our protocols.
- Beyond that, endianness needs to be watched when transferring across the network but that's not really a design problem.

Basic Devops Concepts

Saturday, September 24, 2016 12:43 AM

Things are packaged into artifacts *that then get deployed. An artifact is different from a regular file since the tool that manages it knows how to create it, deploy it, and update/rollback that particular artifact if necessary. Generally this is what goes into a repository; if it's not used by a tool, it doesn't get versioned.*

Tools

- *Jenkins/Bamboo - continuous integration software.*
  - *Jenkins is open source, so that's a big point in its favor.*
- *Flyway - makes databases into artifacts?*
  - *Specifically it's a migration tool, also open source.*
- *Maven can actually do arbitrary pipelining by specifying the "pom" artifact; at that point you can use plugins.*

Deployment Formats

- *Installer*
  - *Contains the full installation.*
  - *Should not overwrite current or newer versions.*
  - *Should ask to upgrade older versions.*
- *Patch*
  - *Contains a delta from one version to another.*
  - *Given this, should definitely not overwrite current or newer versions.*
  - *Doesn't need to ask to upgrade older versions.*
  - *Since it only works from one specific version to another specific version, each needs to be downloaded and applied separately. A client requesting update will need to go in sequence, and if any of the sequence fails the whole thing must be rolled back.*

Build/Patch Servers

- *The developer's machine definitely shouldn't be the machine that makes production or ship builds - in the field a specialized build server takes the current head, runs the full pipeline, then (if it passed build) sends the build to a network share. Stricter servers refuse to send build if tests fail or don't fully pass.*
- *Some particularly powerful places let the dev upload to a build cloud when they're doing dev modifications before pushing to origin; Google does this, for instance. For very large projects this is actually faster than leaving it to build on the dev's box, assuming you can get enough nodes together to reach the necessary processing scale.*
- *Of course this means you need to secure the server; the moment more than one person's working on the project you can't just run builds off your laptop alone. This means authentication and the like, of course.*

Content Servers

- *The content server is what is actually accessed by end users to download the product. This shouldn't be the same server as your build server for this reason.*
- ***This server is what gets asked for patches**, but it should again be some kind of application, not a SFTP request.*

Web Page

- *Last is the web page, the frontend that everybody sees. This provides a link to the latest build, possibly a patch (but your products should really have a "Check for Updates" feature instead).*

Server-side Build Structure

Tuesday, September 27, 2016 10:23 PM

Structure

1. Code repository (Github/Bitbucket). Sends code updates to…
2. Build complex. Performs build and distribution of updated product.
    - Administered by:
        i. Security Monitor. Uses an IDS and monitor like tripwire and Nagios to check and report component status.
            1) See Authentication and Security: Overview for details on the basic security decisions.
            2) Note that this does not implictly have control over any other node; it will need a login but does not need control.
        ii. Docker Layer. **Installed on each server**, runs any nodes operating on the server. Composed of:
            1) Docker daemon
            2) Images for servers
            3) Private registry for images
    - Composed of:
        i. Build Node (Docker instance). Sends packaged builds to…
        ii. Content Node (Docker instance). Links installer to…
        iii. Web node (Docker instance?). Displays the installer link.
3. Separate image repository (Github/Bitbucket/whatever) stores Docker images on backup.
    - **Isn't the registry basically itself a backup? Is this unit necessary?**
        - If you're on a public service like GH or BB they probably have a registry you can upload your images to.
        - **DO's droplets *are* made into images, you can specify backups.**

Remarks

- **Keep in mind that the nodes don't have to be on separate servers.**

Build Pipeline

Saturday, September 24, 2016 12:07 AM

- *Figure out the full pipeline that it takes to build the project.*
- *You know how to build a library of a product, and that's about it; never mind how to build a package, then pack it for distribution.*
- *Blogpost on build system basics:* http://ngnghm.github.io/blog/2016/04/26/chapter-9-build-systems/

Requirements for the Build Pipeline

- It must be remotely triggerable.
- It should be something you can run from a command line.
  - Because of this, on Windows it's probably going to require PowerShell.
  - Everything else uses bash, so you're okay.
- It must be automatable - git has a feature where it'll run your build command when you push to a branch for instance.
- It must not wipe prior builds, since you might need the debug symbols from a past build!
- It must automatically report success and failure.
  - How do we automatically send emails?
- It should be able to test the build too. The results must be automatically reported.
  - This means each product needs some test harness, not just unit test functions.
  - Riot had an interesting way of doing this - tests fired off RPCs. https://engineering.riotgames.com/news/automated-testing-league-legends

Pipeline Algorithm

- Devs should be able to build individual products locally in order to test them.
- The build system must have a build server that nightly runs the master head through the pipeline. On every commit to master the build server also runs a full pipeline.
- A separate content server contains the installers for the built products, and is where end users download the product.
- The build server's pipeline:
  - The pipeline shouldn't assume it's running on a full server; it should **consider that it may be running on a dev box too**.
    - In this case, we can't assume the build steps will just automatically work, we have to *check* that the build tools are available and are properly set up. If they're not:
      - If this is a build server, fail because we need the full pipeline.
      - Else if the products are the requested products, fail because we need that build tool.
      - Otherwise, warn the user and continue.
  - Get the current head of the requested branch.
  - For each package requested:
    - For each product requested:
      - Build the product.
      - Test the resulting product. This should go in stages, and if any fail we email and abort there:
        - Unit tests
        - Integration tests
        - Functional tests
    - If the request is for a full pipeline run, deploy the package to the output share.
      - The deployment step involves packaging the executables into the distribution artifacts - this will typically be an installer.
      - This can also include a delta patcher from the most recent ship build to this build. Then when a build is approved for ship you send it to the content server.

            - All artifacts should be signed, so the build server also needs a certificate.
        - If a failure or error occurs at any point in this pipeline we email and abort at that point.
        - **Should commits to other branches get run?**
            - It'd be very good to have full audit coverage.
            - If the build server isn't set up well it'll be very slow, too.
- This leads to the Content Pipeline.

Tools Needed By Product

- Engine Package
    - Modules Product
        - Rust - Cargo. Builds a library artifact.
    - Tools Product
        - Probably non-Rust. Builds executables.
        - Links:
            - Engine Modules
- Game Package
    - Server Product
        - Rust - Cargo. Builds an executable.
        - Links:
            - Engine Modules
    - Client Product
        - Rust - Cargo. Builds an executable.
        - Links:
            - Engine Modules
    - Tools Product
        - Server Dashboard Subproduct
            - Non-Rust. Builds an executable.
            - Links:
                - Engine Modules
        - World Manager Subproduct
            - Non-Rust. Builds an executable.
            - Links:
                - Engine Modules

Implied Tools Needed for Pipeline

- **Jenkins** should be the build job runner.
- Some kind of **test harness framework** necessary per artifact. Is this part of the Jenkins job?
    - If so, each harness needs to be individually configured. Rust has a rudimentary test system, any higher level language we pick will either definitely have one or we can specify a package to load. OTOH each product probably needs to be configured for Jenkins anyway so this might not be much extra work.
- **Installer packager** per platform.
    - **What does Linux use? There's no real generic system.**
- **Languages** used by products (so Rust, possibly C#)

Content Pipeline

Tuesday, September 27, 2016 7:12 PM

Pipeline Algorithm

- The build server should be able to automatically SFTP the ship build, but in a pinch a ship can be requested manually and the server then moves its latest build over.
    - Note that this is Build -> Content; if the connection credentials are compromised an attacker just has access to the most recent build as opposed to every single build.
- The content server then moves the build from the drop to a publicly visible folder. If you don't have some JS magic on your page, this is when you'd manually update download links.
- For update requests:
    - The requestor sends their version via API call, which the content server sends to some app, the update resolver.
    - The resolver figures out where on the patch chain to start, possibly finding a cumulative patch to start at. The app then sends the sequence of patches that must be installed to the content server.
    - The content server then sends those patches to the requestor to install.
    - The requestor installs the patches. The requestor can report success or failure here.

Implied Tools Needed for Pipeline

- An **update resolver.** This should really be a solved problem, but you may need to create a script of your own for this.
- **SFTP**. Should come with distro but just in case.

Client-side Update Structure

Saturday, September 24, 2016 12:34 AM

- *For most games, there's one client process and one server process. Patching is a matter of deploying to the client install, then the server install, and that's it.*
- *Delta patching is smaller only if you're not throwing everything into an archive. If you're doing archive packs, it's more efficient to take all the deltas, put them in a patch archive, then deploy by unarchiving and applying delta to each affected file. Or so the idea goes.*
    - *The asset archives aren't supposed to be directly moddable, they're supposed to be extended (see add-ons). Can we delta the entire archive instead so we can do the patch in one go?*
    - *And what of the individual engine libraries? The game executables?*
    - As long as the core application isn't touched (that is, the only directories that are modifiable are the config and addon directories), then we can assume the rest of the application directory is a constant that matches a checksum. Then we have the following process:
        - If the installation version is more recent than our patch's version, report too old version and abort.
        - If the checksum doesn't match expected version's checksum, report an invalid version and abort.
        - We can then apply a binary patch.
        - If the patching was not allowed or was somehow partly applied:
            - Report the exact error and abort.
        - Checksum the patched installation against the expected patch value.
        - If the checksums don’t match:
            - Rollback the patch.
            - If the rollback failed for any reason:
                - Report the failure reason and abort.
            - Report a failure to properly patch and abort.
        - Otherwise the patch was successful; report success.
    - The client can't assume the server isn't compromised though, so update protocols should make sure the patches are signed.
        - Open source certificates are very cheap, sometimes free.
        - Anything that's not open source gets kind of pricey though, starting at $100/yr.
- *If we ever do multiplayer though, the server might be on a bunch of different machines…*
    - *This is where something like Docker or the services that use Docker come in. They rely on deploying images as opposed to modifying individual files in an instance, so they're not as resource efficient but are much more consistent, much easier to audit.*
    - *In remote distributed mode you need to be ready for the update deploy to fail. In that case the server must be totally unaffected by the update.*

Pipeline Topology & Price Estimation

Tuesday, September 27, 2016 6:43 PM

Price Estimate

- Standard DO droplet is 5/month, giving you:
  - 512 MB
  - 1 core
  - 20 GB SSD, 1 TB transfer space.
- We can estimate the demands of each node:
  - Security Monitor
    - Medium update (since it's scanning every node)
    - Medium data
    - *Candidate for being a separate droplet due to security isolation concerns.*
  - Build complex
    - Build Node
      - High update (must do build on commit)
      - High data (needs to store every build ever made)
      - *Candidate for being a separate droplet due to resource demand on node.*
    - Content Node
      - Medium update (being hit for installer & patch downloads)
      - High data (needs to store and transfer installer & patches)
    - Web Node
      - Medium update (public face for installer downloads)
      - Medium data (image content)

Implies two possible structures. In both cases only the droplet with a web node needs a domain name.

Proposed Topologies

- ### Two-droplet solution
  - Structure:
    - D1: Private ops
      - Security Monitor, D1->D2
      - Build Node, repo->D1, D1->D2
    - D2: Public ops
      - Content Node
      - Web Node
  - Cheaper @ $10/mo, but not safe; D1 must be WAN-visible to check the monitor remotely. if D1 is hit all of your builds are gone.
- ### Three-droplet solution
  - Structure:
    - D1: Private ops
      - Security Monitor, D1->D2, D1->D3
      - Repository Passthrough Node, Repo->D1, D1->D2. A dummy node with Jenkins that listens for a POST from Bitbucket, then passes the payload to D2 and requests a build.
    - D2: Internal build
      - Build Node, D2->D3
    - D3: Public ops
      - Content Node

        - Web Node
    - Extra $5/mo, but D2 never needs direct access by humans; it can be hardened much more strongly and will not immediately be compromised if D1 is compromised.

Access Control

Tuesday, September 27, 2016 10:42 PM

Pipeline Interactions

Given a 3-node topology, devs should be able to interact as follows:

- Build updated product
  - Input: Edit repo's master branch and commit.
  - Output: Deployed products or an error.
- Reconfigure build pipeline
  - Input: Edit the repo's "ci" directory from any branch and commit.
  - Build node loads pipeline code from ci, so it has the new pipeline without admin intervention.
  - Output: Nothing or an error.

Admins should have the following interactions:

- Log in to nodes
  - Must have SSH cert for node.
  - **Logins are reported via email.**
- View network status via security monitor node

There should be an account just for updating the web content. Following actions:

- Upload and modify web assets
  - This is SFTP access.
  - Should it require a cert?

Servers are configured via Ansible, so we should have a repo to store the scripts on. **Should that repo be separate from the code repo?**

Points of Access

The interactions imply some concrete roles that should exist. Each role must have only the rights specified and no more.

Roles

- Repository access controls the build server (via ci/). Anybody with write access can effectively reconfigure the build server. Is *not* a user on any server.
- Security monitor access allows viewing any server's state, but *not* necessarily any node's state. Should be a single user.
- **Admin access allows control over any server. Should be a single user per server.**
- Web asset access allows SFTP read/write access to web node's public directory. Should be a single user.

Overview

Saturday, September 24, 2016 2:05 AM

Server-side

- **All servers should have:**
  - Security decisions
    - If possible, do not let the server be publicly visible; have it on a private subnet. You probably don't have this much control however.
      - Digital Ocean allows this with a private network. This may cost more…
    - SSH
      - Password-based SSH disabled
      - Root SSH disabled
        - This should be really obvious, but you need to patch it up immediately or get bruteforced within like a month: http://bsdly.blogspot.de/2013/10/the-hail-mary-cloud-and-lessons-learned.html
      - Log sudo commands
      - Explicitly whitelist SSH users; nobody else should have access, and if they somehow do they should be reported.
      - The admin cert…
        - Should be unique to each server.
        - Should be tightly guarded. it's really hard to back this up in a trustworthy way, but unless you want to sync thumbdrives all day you'll need to come up with something.
          - You could make it so these certs are Truecrypted onto backup servers and the password is something you remember/is stored in KeePass. Not like you're losing KeePass lockers left and right, after all.
    - SELinux policy in place
      - DigitalOcean tutorial.
    - fail2ban set up
    - Firewall set up
      - DO tutorial.
    - Log every time the admin logs in; for that matter email any time the admin logs in. Record the IP and time.
    - Disable all services that aren't used by the server. This means X, printers, Samba… see https://www.digitalocean.com/community/articles/how-to-migrate-linux-servers-part-1-system-preparation for some details on how to find what services you have enabled.
    - Default permissions should be specified: https://www.digitalocean.com/community/articles/linux-permissions-basics-and-how-to-use-umask-on-a-vps
    - Use maldet to scan for malware. Rare but you never know.
  - Monitoring and reporting systems
    - You probably can't monitor the logs easily yourself; get something like logcheck to do it for you.
    - It might just be better to have a monitoring solution like Nagios (to check whole network status) or a IDS like tripwire (for watching individual servers); they're still open source and you'll have something comprehensive instead.
  - **Automatic system patching**
    - Remember - none of this matters if your patches are out of date. Only update via distro's official signed security channels.

    - **If possible, these decisions should be packed into a Docker instance that can be deployed whenever the server gets broken.**
- Watch for obviously suspicious behavior:
    - Port scans
    - Hammering from a single address (fail2ban should automatically catch this)
    - Multiple addresses hammering the same user

Client-side

- For clients, a user/password is enough, but for administration of anything that really won't cut it in this day and age.
- **Can we have certificates?**
- **Can tools transmit via SSL? This would make dashboards a bit less scary.**
- **Can the *client* transmit via SSL?**

Build Server

Monday, September 26, 2016 3:40 PM

- See Overview for common requirements.
- The internal build system should have a separate user with rights to run; it should not be able accessible via SSH. If it does email the admin.
- If using the 3-node topology, no human logins should be necessary. **Logins from any user besides the monitor or the passthrough should trigger an email.**

User Roles

- Admin does any installing.
- Devs should only be able to drop new code in via commit and read output builds if available, they have no access otherwise.
- The admin does not have rights to the drop point or build point.
- The internal build system should have read rights to the drop points, write rights to the output point and execute rights on tools. It also has access to email so it can email pipeline status. No access otherwise.
- Everything needs to be logged; apparently cron can tag its tasks.

Content Server

Monday, September 26, 2016 5:15 PM

- See Overview for common requirements.
- **Completely separate server from the build server.**
- The public folder should be the only folder visible to the WAN.

User Roles

- Build system should have write access to the drop folder and some way to notify the content server; no access otherwise.

Resources

Tuesday, September 27, 2016 7:45 PM

- https://www.digitalocean.com/community/tutorials/an-introduction-to-securing-your-linux-vps

Operating System

Tuesday, September 27, 2016 11:02 PM

- **Operating system is CentOS, whatever the most recent version is.**

Server Deployment & Configuration: Ansible

Wednesday, September 28, 2016 3:13 PM

- Used to reproducibly set up servers.
- **Our tool is Ansible.**
- Testing tutorial: http://docs.ansible.com/ansible/test_strategies.html

Node Deployment & Configuration: Docker

Tuesday, September 27, 2016 6:07 PM

- **Our tool is Docker.**

Basic Concepts

- Instance templates are called images. Images are stored on registries so Docker systems can pull whichever template they need.
- Those get installed into actual instances called containers. The containers and such are administrated by a server process called the daemon.
- The actual processes running the containers have an arbitrary architecture; all you care about is that they all are controlled via one central point, their service. Daemons let you control services, which run the containers.
- Docker's really meant for deploying and isolating *applications*, not servers. Don't use this to set up a new server, use Ansible playbooks instead.
    - *Do* use this when you're doing testing though, since you can have a bunch of different configurations all saved to images you swap in. For installation testing in particular this is useful.

Advice

- Seeing as how the images start at hundreds of megs and go up from there, it's probably best to store them on a separate repo.
- Important to reduce number of instructions in a Docker image: http://developers.redhat.com/blog/2016/03/09/more-about-docker-images-size/
    - Keep your RUN calls for initial setup to a minimum, since the image does COW to maximize speed over size.

Continuous Integration

Tuesday, September 27, 2016 6:08 PM

- **We will use Jenkins.**
- Pipeline tutorial: https://jenkins.io/doc/pipeline/

Network Monitor

Tuesday, September 27, 2016 10:26 PM

- **Nagios is the pipeline monitor.**
  - DO tutorial.

Security Package

Tuesday, September 27, 2016 10:27 PM

- **Iptables is the firewall.**
  - DO tutorial.
  - **Keep in mind IPv6 is handled by ip6tables; set both up in the initial image.**
- **Tripwire is the intrusion detection system (IDS).**
- **fail2ban is the connection filter.**
- **SELinux is the system policy enforcer.**
  - DigitalOcean tutorial.
- **maldet is the malware scanner.**

Web Display Framework

Tuesday, September 27, 2016 10:30 PM

- **This has not been chosen yet.**

Locker Specifications

Wednesday, September 28, 2016 3:42 PM

- Login credentials need to be placed in known secure and backed up locations.

Credential stores

- All credentials **(store in a locker)**
  - Administrator (locker optional). Place anything responsible for control over the entire system or over any hardware here.
    - u/p locker
      - Server admin logins
        - Admin login
        - Admin certificate passphrase
        - Root
      - Repository logins
        - Code admin
        - Code admin certificate passphrase
        - Server config admin
        - Server config admin certificate passphrase
      - Misc logins
        - Inaccessible test user login
    - Certificate locker
      - Server admin certificates
        - Admin **(has passphrase)**
      - Repository certificates
        - Code admin **(has passphrase)**
        - Server config admin **(has passphrase)**
      - Misc logins
        - Inaccessible test user
  - Service Management (locker optional). Place non-administrative credentials here.
    - u/p locker
      - Server logins
        - Security monitor
        - Web asset
      - Service logins
        - Security monitor remote view
    - Certificate locker
      - Server certificates
        - Security monitor
        - Web asset

Node Specifications

Tuesday, September 27, 2016 10:14 PM

- Right now these don't need to be implemented as Docker images, but they should be specified in playbooks for server deployment.
- **Deploy scripts should report/email on failure.**

Distinct nodes:

i. Security monitor node.
   - Software:
     - Network monitor **(Not Started)**
   - Configuration:
     - Monitor user. Used by monitor to watch other servers. **(Not Started)**
       - If it needs a SSH cert for other servers, this is not the same cert as an admin's cert. **(Not Started)**
     - Monitor view user. Used to remotely view monitor; is a user in the monitor's system, not a user in the server itself. **(Not Started)**
   - Policies:
     - Network monitor is listening to web endpoint **(Not Started)**

ii. Repo passthrough node.
   - Software:
     - Continuous integration framework (CIF) **(Not Started)**
   - Policies:
     - HTTP/S inbound from repository open **(Not Started)**
     - CIF:
       - Is listening to POST on web endpoint **(Not Started)**
       - Can POST to Build node's web endpoint **(Not Started)**

iii. Build node.
   - Software:
     - CIF **(Not Started)**
     - Build tools for products, see Build Pipeline: Implied Tools Needed for Pipeline. **(Not Started)**
   - Policies:
     - HTTP/S inbound from passthrough open **(Not Started)**
     - CIF:
       - Is listening to POST on web endpoint **(Not Started)**
     - SFTP outbound to content node open **(Not Started)**

iv. Content node.
   - Software:
     - Update resolver, see Content Pipeline: Implied Tools Needed for Pipeline. **(Not Started)**
   - Policies:
     - HTTP inbound and outbound is public **(Not Started)**
     - Update resolver:
       - Is listening to POST on web endpoint **(Not Started)**
       - Can HTTP respond to client **(Not Started)**
     - SFTP inbound from build node open **(Not Started)**
     - Can update web node with links to new installers (web endpoint?) **(Not Started)**

v. Web node.
   - Software:
     - Display framework; if you use Wordpress again I will be very cross from the past **(Not Started)**

- Assets:
  - Web layout and base content **(Not Started)**
- Policies:
  - HTTP inbound is public **(Not Started)**
  - Content node can update site links to new installers (web endpoint?) **(Not Started)**

Server Specifications

Wednesday, September 28, 2016 2:50 PM

See Pipeline Topology & Price Estimation, 3-node topology.

Playbooks Needed

i. Base playbook

All implementations derive from this. Has security installed and configured.

- Software:
  - Operating System **(Not Started)**
  - Security Package **(Not Started)**
- Configuration:
  - Operating system:
    - ☐ Automatic patching is configured. **(Not Started)**
    - ☐ Default permissions are specified. **(Not Started)**
    - ☐ Root user has a unique password; this password is not stored in the same locker as admin logins. **(Not Started)**
  - Security package:
    - ☐ Firewall is configured for default restriction. **(Not Started)**
    - ☐ System policy enforcer is configured. **(Not Started)**
    - ☐ Connection filter is configured. **(Not Started)**
    - ☐ IDS is configured. **(Not Started)**
    - ☐ Malware scanner is configured. **(Not Started)**
  - Admin user. Is not itself a root user, but has sudo permissions.
    - ☐ Login triggers an email noting the IP and time. **(Not Started)**
  - Inaccessible user. This is for test purposes to verify that the SSH remoting is only allowing specified users.
    - ☐ This user has no rights to anything.
    - ☐ This user must never be on any whitelists.
- Policies:
  - All ports closed except where otherwise noted. **(Not Started)**
  - All services closed **(Not Started)**, except those needed to run ports **(Not Started)**.
  - SSH:
    - ☐ Port inbound/outbound is open. **(Not Started)**
    - ☐ Password-based SSH disabled. **(Not Started)**
    - ☐ Root SSH disabled. **(Not Started)**
    - ☐ SSH from non-whitelisted users disabled. **(Not Started)**
  - Know where sudo command logs are. **(Not Started)**
  - su is completely forbidden. **(Not Started)**

ii. Implementation Playbooks

These inherit software and policies from (i):

a. Private ops.

- Logical nodes (do not actually load Docker images):
  - ☐ Monitor
  - ☐ Passthrough
- Software:
  - ☐ Continuous integration framework (CIF) **(Not Started)**
  - ☐ Network monitor **(Not Started)**
- Configuration:
  - ☐ Operating system:
    - ◆ Root user has a unique password; this password is not stored in the same locker as admin logins. **(Not Started)**

        - Admin user. Is not itself a root user, but has sudo permissions.
            - Login triggers an email noting the IP and time. **(Not Started)**
        - Monitor user. Used by monitor to watch other servers. **(Not Started)**
            - If it needs a SSH cert for other servers, this is not the same cert as an admin's cert. **(Not Started)**
        - Monitor view user. Used to remotely view monitor; is a user in the monitor's system, not a user in the server itself. **(Not Started)**
    - Policies:
        - Network monitor is listening to web endpoint **(Not Started)**
        - HTTP/S inbound from repository open **(Not Started)**
        - CIF:
            - Is listening to POST on web endpoint **(Not Started)**
            - Can POST to build node's web endpoint **(Not Started)**
b. Internal build.
    - Logical nodes (do not actually load Docker images):
        - Build
    - Software:
        - CIF **(Not Started)**
        - Build tools for products, see Build Pipeline: Implied Tools Needed for Pipeline. **(Not Started)**
    - Configuration:
        - Operating system:
            - Root user has a unique password; this password is not stored in the same locker as admin logins. **(Not Started)**
        - Admin user. Is not itself a root user, but has sudo permissions.
            - Login triggers an email noting the IP and time. **(Not Started)**
    - Policies:
        - HTTP/S inbound from passthrough open **(Not Started)**
        - CIF:
            - Is listening to POST on web endpoint **(Not Started)**
        - SFTP outbound to content node open **(Not Started)**
c. Public ops.
    - Logical nodes (do not actually load Docker images):
        - Content (maybe this should actually be an instance)
        - Web
    - Software:
        - Update resolver, see Content Pipeline: Implied Tools Needed for Pipeline. **(Not Started)**
    - Configuration:
        - Operating system:
            - Root user has a unique password; this password is not stored in the same locker as admin logins. **(Not Started)**
        - Admin user. Is not itself a root user, but has sudo permissions.
            - Login triggers an email noting the IP and time. **(Not Started)**
    - Policies:
        - HTTP inbound and outbound is public **(Not Started)**
        - Update resolver:
            - Is listening to POST on web endpoint **(Not Started)**
            - Can HTTP respond to client **(Not Started)**
        - SFTP inbound from build node open **(Not Started)**
        - Can update web node with links to new installers (web endpoint?) **(Not Started)**

Test Specifications

Wednesday, September 28, 2016 4:49 PM

- Nodes
  - All node install scripts:
    i. Install the given software.
    ii. Set the given policies and OS configuration.
    iii. Create the given users, if necessary.
      1) The users have the specified permissions.
- Servers
  - All servers:
    i. Have the policies described by the node install scripts used on the server.
      1) Do *not* have additional policies that were not described.
    ii. Have the given software described by the node install scripts.
    iii. Are accessible via described access points with that access point's credentials only.
      1) Are *not* accessible via alternate credentials (i.e., SSH password login should always fail).
      2) Are *not* accessible via any other access point.
      3) Invalid access points are immediately rejected.
  - *(i) and (ii) are probably directly testable with the server deployment tool; the deploy scripts can include validation code. (iii) may require a separate test script.*
- Pipeline
  - Test Building
    i. When changes are pushed to code repository's master branch:
      1) The change is built into a product on the build server.
      2) The packaged product is ready for distribution on the public ops server.
  - Test Reconfiguration
    i. When changes are pushed to code repository's master branch:
      1) The build server uses the configuration specified in the master head.

Dashboard

Wednesday, September 21, 2016 4:28 PM

- You've got the superpowers, each has some subfactions:
  - Republic
    - League Members. The individual nations that make up the League.
  - Empire
    - Imperial Houses. These are the heirs to the throne.
  - Superpower ops are on a common page.
- Then the international groups:
  - Peace Commission. The space police! They also sometimes help with disaster relief, but don't count on it.
  - Crime syndicates:
    - Imperial Mafia. Organized, tough, and very old.
    - Republic Syndicates. Younger than the Mafia but a lot less predictable too.
    - Crime ops are on a common page.
  - Cults:
    - NOW. A very new "life improvement seminar" that's gotten popular with new money and young entrepreneurs in the Republic.
    - Core Servants. Relatively young, relatively nonthreatening cult descended from Imperial nomad cults.

Republic

Sunday, September 11, 2016 5:27 PM

Overview

- Descended from groups that stayed in the original Eden systems. These systems were deeply reliant on imports however, so they quickly turned to fighting each other for resources. The resulting age of war is called the Fall of Eden.
- The outbreak of war locked the Fall nations into using **short-ranged equipment with high damage yields**. Anybody that thought they could flee and had the equipment to do so left long ago. As combat dragged on and ruined generations of men and women, **the warring states turned to cheap non-sapient AI, or Scripted Intelligence (SI)**, to fill their skeleton-thin ranks. War became faster and ever more wasteful, marked by moments of inconceivable bloodshed when an automated attack finally pushed to an inhabited system. If you are a Republican your homeworld has at least one of these massacres in its history and caused at least one, there are no exceptions.
    - A result of this is that almost all Republican equipment has some kind of SI guidance, and to this day they follow the high-damage, short range doctrine. **They love missiles and drone support craft, using these to overwhelm and confuse attackers.** The lack of people means their MMIs aren't very good, though, and the close ranges of their war mean that **they have weak FTL range and don't have very good capital ships**.
    - Some states *did* make high-power AI. Curiously all of those AI either killed themselves, vanished, or stole a FTL ship and ran away from Eden.
- The systems that survived united to prevent such a tragedy from happening again; the resulting Republic works hard and has a lot of harmony among its member nations. However they are very much distrustful of outsiders, believing them to be subversives that caused the original Fall.
- This was not helped by first contact with the Empire, which led to the Reunion War. The "scouting fleet" that the Republic defenders encountered would've been enough to defend or assault any homeworld, and it led to an intense buildup of arms.
- Still have about 80% of their population based on planets, with the remaining 20% on massive station complexes usually in lower orbit. Having habitable planets greatly reduces logistics concerns.

Empire

Monday, September 12, 2016 12:01 AM

Overview

- Descended from nomads that went coreward for various reasons. The worlds they found weren't hospitable, but were flush with resources. They learned to build big FTL mobile colonies, and the mineral supplies combined with powerful solar output led to large food production. They thus have a large population despite the harsh environment.
    - The nomadic life and mobile colony construction means they have a good understanding of **long-range on large vessels**, while the demands of combat between the original tribes and just living near the core led to a **deeper understanding of man-machine integration** and bioengineering.
    - That same nomad life and lack of unity stifled research development however - they have engineered their ships well by this point but **don't have much automation** or self-guiding weaponry. **They prefer low-guidance weapons on turret and spinal mounts; contact with the Republic would lead to a focus on range to kite damage dealers. They do have carriers, but usually don't see the point of using them unless they totally maximize carrying potential.**
        - This doctrine turns out to be a very good idea when warp torpedoes are developed, since the **carriers can then be retrofitted into mobile tactical silos**; one of these boomers can ruin an entire system's day.
- Eventually the tribe that would become the Empire became strong enough that they intentionally sought out other tribe-fleets to conquer them. Unlike normal conquests however, they did not pillage surrendering fleets and allowed all conquered fleets to maintain their identities. Once word got around that there was a dominant group, weak tribes intentionally joined to put a stop to raiders. This eventually snowballed into all of the major tribes uniting under the Empire.
- Once it became clear they had no real competition, they sent scouting fleets in several directions. One went further coreward, another along the arm to look for unusual artifacts… and the most famous one went rimward to see what happened to Eden. This led to the Reunion War between the Republic and the Empire.
- Their planets aren't generally inhabited since most are radiation-blasted wrecks - Imperials set down an AI Waystation that maintains a set of drone workers and only visit if they need to take control of the drones.
    - The waystation managers are generally treated kind of like the weird uncles of the family: they're highly respected for their experience and independence but aren't understood too well because they're so physically remote. One AI is usually enough to run a whole waystation and they don't really need emergency rescue if their core is properly placed. On the other hand they're usually not really isolated since they have a com tower linking them to the rest of the Empire.
- Asteroids are often colonized since the sheer mass makes for an excellent defense; as with a lot of Imperial equipment colonized ones tend to be mobilized so they can stay close to the area they're intended to supply.
- Empire ships tend not to have automation, but they have superior mind-machine integration since they need to rely on people so much. The AI they do have are advanced, on the level of a human but not quite the same psychologically.

Superpower Operations

Saturday, September 24, 2016 11:27 PM

Description

Common operations performed by superpowers. Superpowers include:

- Republic
- Empire

Operations

Superpowers in general have the following operation departments:

- Intelligence. Enter the other side's ranks or territory to obtain intelligence and assets. Divided into departments:
  - HUMINT. Retrieval of important people and gathering information by talking with people.
    - Remember that people aren't necessarily organic anymore.
    - Requires some sense of social skills.
  - TECHINT. Capture and analysis of enemy ships and equipment.
  - GEOINT. Basically recon; enter opposing territory and get geographic/technical scans of some feature in the territory.
  - SIGINT. Probably not available for players, but analysis of communications is of course essential.
- Logistics. Ensure faction facilities and fleets are stocked. Generally is something you donate to rather than do missions for directly.
  - Civilian Logistics. Operate in core systems for the most part.
  - Military Logistics. Operate in border systems or between FOBs and active fleets.
- Security. Fight space crime, basically.
  - Low-intensity. Handled by the faction's police.
    - ?
  - High-intensity. Handled by the faction's military.
    - ?

Peace Commission

Saturday, September 24, 2016 11:07 PM

Description

Diplomatic NGO established by the two powers to find non-destructive ways to resolve their disputes. Eventually also chartered to defend all humans from internal or external threats, no matter their political stance. In other words this is the space police.

Operations

- Customs Patrol. Go through systems scanning transports, and intercept suspicious transports.
- High Threat Operations. Neutralization of groups that the local faction does not have the equipment to handle.
    - Threat Location. Scout systems for criminal hideouts and bases.
    - Threat Neutralization. Neutralization of criminal facilities.
- High Value Operations. Arrest of specific fugitives wanted for crimes against humanity. Fewer targets than HTO but the targets are either much more heavily guarded or far more dangerous.
    - Counterterrorism. A facility has been compromised with a bomb/boarding party; secure the area and neutralize the internal threat. If your command ship's not carrying the defuser/CT team, you have to guard whoever is.
        - Win conditions:
            - Defuser performs Defuse action and timer runs to completion.
        - Lose conditions:
            - Operatives have no Defusers
            - Enemy threat timer runs to completion
            - Operatives are defeated or retreat
- Emergency Response. An international disaster is about to or has happened; provide relief and rescue to victims.
    - Disaster Prevention. Prevent an international disaster from starting, such as preventing a colony reactor from going critical.
        - Similar to Counterterrorism except the timer is way shorter or outright unknown; usually there's no enemies in exchange.
        - Failure is super super bad for public relations…
    - Site Remediation. A disaster is underway and has contaminated or is consuming an area; clean the area of contaminants/fire.
        - You're given special weapons to remediate with; it's hard to lose but you are ranked for speed and it's very tough to get a good rank.
        - The higher your win rank, the better PR boost the Commission gets.
    - Search and Rescue. Find people that have been reported missing, or search a site for survivors. There's not a lot of time and the areas involved are usually still being remediated.
        - You're given close-range scanners to find survivors and a long range hot/cold meter; you have very little time to save a single person, but picking up a survivor increases the time.
        - If the timer runs out the survivor you're looking for died, but this isn't itself a mission failure.
        - Mission failed if too many survivors die in a row. Like most Emergency Response missions, very hard to get good ranks.
    - Disaster Relief. Supply an affected area with food or other materials.
        - Only way to fail is to lose the supply ship, extremely easy from your end.
        - Rank is easy to max out too, similar to Logistics ops.
        - High rank provides PR bonus to the Commission.

- PC only operates against groups or fugitives wanted by the Republic *and* the Empire; otherwise they don't get involved.
- Working with a faction's security can eventually lead to an invite to the Peace Commission.

Imperial Mafia

Saturday, September 24, 2016 11:30 PM

Description

The crime families that plagued the Empire in their exodus have gained huge amounts of influence with the more permanent settling of systems and the move back into the Eden systems.

- They prefer covert operations, diplomacy and economic strength. Organized crime should pay better than disorganized crime, after all.

Republic Syndicates

Saturday, September 24, 2016 11:30 PM

Description

Unsanctioned corporations and cartels in the Republic that have established relatively stable alliances. They are nowhere near as cohesive as the Mafia but have the home field advantage in Republic systems.

- Focus on overt operations and the more violent public relations ops with superpowers. Of course the syndicates like money, but they like their dominance more and will do the most brutal acts if it means their turf is safe.

Crime Syndicate Operations

Saturday, September 24, 2016 11:12 PM

Description

The common operations performed by criminal organizations. Crime organizations include:

- Imperial Mafia
- Republic Syndicates

Operations

- Syndicates in general have the following operations:
  - Dominance. Attack enemy facilities or operatives.
    - Covert. Remove the target quickly and quietly.
      - If you reach a detection threshold, you fail. Harder missions may also fail if you've gotten too much police attention *after* the strike.
    - Overt. Intent is to make an example of the target; you need to attract as much attention as possible.
      - Can't stay long or PC will steamroll you, though.
  - Work. Distribute supplies and smuggle products.
    - Distro. Move legal supplies between sites. Your goods are clean but this is still extremely suspicious.
      - Special subtype is Diplomacy, where you're transporting a VIP under heavy guard. You're expected to be spotted or attacked by rivals; your only goal is to ensure the VIP's safety in and out of the destination.
    - Import/Exports. Smuggle contraband to or from a destination. Less suspicious but much bigger problems if you get caught.
  - Public Relations. Random missions from the superpowers, the result of them calling in favors to your syndicate. Can be any type, including standard syndicate missions in the opposing superpower's territory; almost never Logistics though.

NOW

Saturday, September 24, 2016 11:07 PM

Description

A strange cult; all they seem to do is draw in money and "improve lives". Something's definitely up but nobody knows what.

- They turn out to be by *far* the largest threat since all they want is to be the ones in charge, NOW can't be bribed like the Mafia/syndicates can. Eventually the superpowers have to step in to stop their attempt to seize control.

Operations

- □ Pacification. Basically Covert Dominance missions against very heavily guarded targets.
- □ Recruitment. Long-range versions of Import Work.
- □ Preparation. Mixture of GEOINT and Export Work; infiltrate a guarded location to deliver some item or person.
  - Sometimes this is a Violent Preparation, and the "delivery" turns out to be a bomb or a box full of marines; in this case you need to guard the package until it's done, then (for your sake) escape.

Joining them makes you a huge, huge jerk, but if you want to take over the whole map they're probably the easier way if you can keep the takedown from triggering. Alternately you can let it play out and make your move while the superpowers are busy fighting NOW.

Core Servants

Saturday, September 24, 2016 11:07 PM

Description

Relatively young monotheistic cult. Believes that the creator can be communicated with and that entanglement has something to do with it.

- Mostly this is just a matter of faith and it causes a lot of trouble where Republic/Imperial officials that believe in the Core are accused of being saboteurs. For the most part this isn't true beyond embezzlement, and the tenets of the Core actually make them more loyal.
- On the other hand they're not that helpful to the outside world; they're working on very bizarre FTL technology that only gets used by their leaders to escape from NOW. This leads to a post-war schism in the Core faith.

Operations

- Research. Scanning of jump data and FTL drives. A lot of following people; sometimes you'll follow them somewhere you really shouldn't be in, which actually adds a bonus for the intel you found.
- Logistics. Retrieval of supplies and finances to fund all this research.
  - Can be pretty tough to do, but isn't suspicious from the start like syndicate Logistics is.
- Diplomacy. Do favors for other powers so they don't wreck the Servants.
  - Kind Words. Favors for the superpowers.
    - Any of the Espionage types, rarely Civilian Logistics.
  - Watchful Eyes. Favors for the Peace Commission.
    - Generally Threat Location, very very rarely High Value Operations.
  - Helping Hands. Favors for the syndicates.
    - Generally Work, very very rarely Covert Dominance.

Joining them gives you weird equipment and ways to travel.

History

Wednesday, September 21, 2016 4:16 PM

- Factions are created
- Factions reestablish contact
    - Reunion War
- Contact ends with a relative balance, nobody can really establish dominance that can last
- Subfactions begin to make diplomatic connections and relationships now that world relations are relatively friendly. Neutral peacekeeping organizations and crime syndicates form, further solidifying the world.
- Technology advances to the point that a small team can man and maintain a fleet with system-wide range; it's not practical to keep the information out of their hands and eventually the infrastructure required spreads too. This leads to a lot of chaos as every budding warlord or cult leader tries to split off from their local colony/planet/whatever, and this is the era that the player finds themselves in.

Reunion War

Wednesday, September 21, 2016 2:09 PM

- Or the "First Contact War", but that's considered a misnomer given that the Empire is descended from Eden too. A Republican calling it this is sort of like "The War of Northern Aggression".
- In any case, this started when an Imperial scouting fleet jumped into a border system. A Republic patrol wasn't too far away and tried to interdict the jump flare they spotted. Instead the Imperials caught *them* for a short while before they escaped the interdiction bubble and returned to HQ.
- The incident put the Republic on the highest alert. It wasn't quite panic as we would understand it; after all the Republic had just been through the end of the world, this was just another apocalypse and they wouldn't let it kill them. Fleets were pulled out of peace vaults, orbital defenses calibrated. Every eye and sensor looked to the borderlands for a hint of what these new invaders planned to do…
- Which wasn't much, of course. The Empire didn't become great by destroying, it became great by conquering. *And out in the wastes there wasn't anything to conquer.* The scouts reported their findings to Central Command and moved on towards the central systems. They didn't know what awaited them, but they still had the legends their parents and their parent's parents told about Old Eden.
    - Eden, at least the way the story usually goes, was a great place. The cities were made of gold, the glass was made out of diamonds and all the food was real food, not synthfood. In Eden there was so much to go around that people didn't need to live on a colony, they could live on a regular planet and not owe anyone anything…
- Which made the first jump into a core system disillusioning. The scouts couldn't stay long before thousands of ships vectored to swarm them, but the glimpses they caught of the Republic homeworlds was disturbing. Massive craters pockmarking dead cities that themselves covered continents; once blue oceans turned dusty grays and lurid yellows. Wreckage still in orbit, tearing up still inhabited colonies - but that didn't stop those colonies from arming and launching whatever interception fleets they had. The scouting fleet quietly placed their probe drones then left, destroying whatever ships that managed to reach them.
- Not every ship was on point of course, and the stragglers stayed behind to distract the enemy, keep them from interdicting the main fleet. Those that stayed behind surrendered, horrifically missing the difference between how the nomads treated war and how the Edeners treated war. To be fair most of those ships were captured instead of destroyed, but the treatment they received afterwards was considered a shameful crime by both sides.
- So the scouts headed back to Empire territory, and the actual warfleet headed out to Eden. The War Council decided that while they couldn't just conquer the Eden systems in one go, they could make a start by destroying their communications; without coordination they wouldn't be able to keep all those planetary defenses supplied, and then they could eat the enemy piece by piece. The com towers themselves were heavily defended even by Imperial standards, but they would find a way. This was before torpedoes, so the plan was simple.
- One April day a couple thousand dreadnoughts and carriers jumped into each core system of the Republic. Every major com tower reported enemy contacts, specifically bombardment by laser; eventually one went down. SDC wouldn't know exactly what happened for several days but that didn't matter - the command went out to focus fire and throw everything they had at the other strike fleets. Remarkably enough this actually had an effect: for some reason the enemy's cohesion broke down much faster than expected once they took some casualties, and eventually the attack was rebuffed. All of the core towers took heavy damage, but no others were actually destroyed.
- But then it became a little clearer why they were so easily turned away. Repair crews and resupply squadrons kept disappearing. Nobody could catch the enemy in the act thanks to all the damage the first assault caused, and by the time a patrol could react to a distress call they were long gone.

Republic forces weren't taking *bad* losses, but at that rate the enemy could win out of sheer attrition. And after all that they didn't even really know who they were fighting…

- There were prisoners, but they could barely be understood; naturally they stopped cooperating once they experienced how the Republic treats POWs. Intelligence was able to determine that the enemy was some kind of Empire, that they came from outside Eden, and that all of their ships were manned. That was about it. The Imperial ships were much more informative, and would prove key to winning the war. Their designs pointed towards ways to make Republic vessels stronger, faster, and more efficient; meanwhile the captured communications equipment would give a peek into how the Empire coordinated itself. Notably there was much more integration of the crew into ship equipment - people didn't man the ship so much as become it.
- SDC was able to reverse engineer the MMI interfaces; more importantly the comms systems to Imperial central command were apparently still transmitting. This led to a terrible idea by Intelligence that really never should've worked: hook a *Republican* soldier into the comms system. At first they just listened, but eventually they moved to transmitting, letting the Imperials know that their sons and daughters were very much alive. This didn't lead to the hoped-for Imperial rescue attempt that SDC could then ambush though - instead it led to a confused response from Imperial command. Roughly translated, it read:
- *<<Why are you bargaining with your children?>>*
- The ship Intelligence had been using as the transmitter then started powering up, and the engineers were just barely able to keep it from revealing its location to the Imperials.

Travel

Saturday, September 17, 2016 11:47 PM

- Works by exaggerating the ship's spacetime presence: compress distances in front, extend distances in the back. The drive can only move in the direction it is mounted but can turn on any axis.
- Spacetime is being moved, not you, so **when you exit you have no velocity**. And when you're "moving" you have no local velocity; the global velocity being experienced is called *warp speed*.
- The faster your warp speed, the larger the bubble around your ship. Contact between this bubble and realspace distortions like mass causes feedback that the FTL drive must work to overcome, sort of like drag on a wing. **You can therefore fly through masses, but they will always slow you down at the very least.**
    - This makes in-system warp slower almost by definition.
- If the feedback is too strong for the drive, you get forced out of warp, or *locked*; getting locked is very bad for the drive and will generate a point flare at the point of lock. For your ship this means a giant ball of radiation hits you and you die. For the celestial you hit it doesn't usually mean much given their scale, but it might cause some radio crackling on a vacuum world.
    - This is why you can't warp at interstellar speeds inside a system; you'd almost immediately get locked by a planet before you could do anything, never mind the system's star. Also why using this as a bomb i s impractical, since you'll get slowed down by very small masses like people and machinery and won't generate enough of a flare by the time you actually lock at the desired size.
    - This is also why you can't automate combat-speed warps with star charts and no FTL sensors, since they're so large that any error would again result in a lock. **If you have FTL sensors, your ship will emergency exit if its warp vector will lead it anywhere near a lock - you'll get hit by a flare but it'll at least be survivable.**
- Safely exiting warp involves drawing down the drive over time; it's a fairly quick action on the order of seconds, but not like a lock where the "draw down" is over millionths of a second.
- Exiting generates a warp speed-dependent flare in the front - interstellar warps are huge relative to a ship (not so much to an asteroid) and will kill ships caught in the flare, in-system warps are tactically significant but not especially dangerous to present-day craft.
    - In general you should exit near friendlies as slowly as possible and exit near hostiles as quickly as possible. Ship computers will precompute safe exit speeds for you.
- FTL drives can also generate a reverse bubble, which doesn't lock on contact but neutralizes the warp effect. FTL drives still run, they just won't move the ship unless your drive is that much stronger. Since the interdictor's field dominates, trying to attack with a high-speed exit won't do anything since you're not warping at any significant speed.
    - Since the bubble is in reverse the interdicting ship can't move, but it can still turn. Usually the bubble is huge and you can't warp at all in it (otherwise you'd have left), so **the only reasonable way to escape is to blow up the interdictor**.
- The interdictor effects lead to these implications at least:
    - Rear-ending another warp bubble just sort of slows you down, it's not much of a collision.
    - Head-on colliding with another warp bubble is hellish - the compressed fronts merge, you haven't locked yet so if you don't slam into other ships there's a good chance you'll lock into them, then the extended rears ensure you stay stuck (assuming anyone's alive).
    - Colliding from the sides is similar to colliding with a mass generating the same-magnitude distortion: you lock and everyone dies.
    - All of this requires precisely hitting in deep space though, which you can't do without FTL sensors, and the defending ship can always emergency exit or turn very slightly to completely obviate the attack.

Communication

Tuesday, September 13, 2016 9:52 PM

- Remotes
  - Remotes are pairs of entangled particles; **they don't have a maximum range and don't appear to have any latency**.
  - However **communication via remotes requires a living creature**; doing it completely via a non-living device results in no control over the signal.
    - **The sender, or *speaker*, does not have to be sapient or organic**, as long as it is sufficiently complex to be alive.
      - This isn't much good philosophically, since you can hook a rock up to it and get a non-random signal. Interpreting these signals is a common hobby for numerologists and other dorks that read too much into things.
      - This is very useful practically, since it means **you can use a MMI as a coprocessor** to help you switch the remote. Having a MMI interpret the base-level signals takes a huge load off a speaker and lets you do a lot more with much smaller bandwidth. Certain scary factions use specially-designed organs instead of cybernetics.
    - **Once a remote pair's formed, you can't reentangle the pair.** Normally this is no big deal, it just means you take the destination's remote out and hook in a different remote linked with the person you want controlling it. Just make sure the pair isn't super far apart!
    - **If one end of the remote is destroyed, the other end is still intact; you just can't entangle it with anything now.**
      - The feeling of losing a remote is unpleasant but not uncomfortable; you sorta feel disappointed in yourself.
  - Relaying and mux/demuxing remote streams doesn't scale in a human brain very well, much like trying to hold more than one conversation at a time. Because of this switching time is paramount once you've reached a concurrency limit. A human can reliably mux two or three streams without an MMI; with an MMI this goes up to 6 or 8 but it requires putting all of their focus into managing the streams. Purpose-built AI or GELFs can usually handle way more streams in parallel.
  - Humans in particular can do strange things with streams though - sometimes they can predict some of the incoming data and often they compress/decompress it much more efficiently than synthetic life. However none of these effects are the same from person to person, so they can't be integrated into the design of a communication system.
- Com Towers
  - By far the most important communication structure in the modern world; without these towers interstellar communication is pretty much impossible and none of the superpowers would exist.
  - Contain at least one lifeform acting as the relay, usually far more than this to even out the load of a modern communication network. A small tower has at least two dozen people on the relay crew, and a large one several thousand. Factor in the security, repair and sanitation teams, the cooks and so on and you have a small town centered around these towers.

Detection

Saturday, September 17, 2016 11:48 PM

- Detection
  - **Normally when a ship warps you see an entry flare, then a directional exit flare** where it drops out of warp… eventually, when the flare's radiation reaches you. This is clearly garbage since seeing the enemy mostly means that you now know where they *aren't*.
  - Instead it's possible to get a sense of where a thing is right "now" (it's not clear that this means there's one special frame of reference). Again, **this requires a living being to feel out the signal.**
  - Things moving about messes with spacetime, and these changes can be detected instantaneously, they just can't be interpreted with a nonliving device. Plug the signal into a creature's brain and they *can* figure out what's where. Normally this is way too much data and gives a human-sized brain a seizure, so an MMI acts as intermediary. **The MMI performance limits the range and accuracy of the detection.**
    - If you hooked it up to a really large AI, it might be possible to get crazy ranges with absolute accuracy. This requires making a superintelligence that would probably go do something less boring, though.
  - The really messed up thing is that **you can feel the thoughts of others using the detection nearby**; it is *not* a good sensation unless you know they're friendly. You can't truly hide from a detector but you *can* communicate with them mind-meld style; **disorienting enemy detectors is the first order of signal warfare in space**.
  - **In FTL the detector is also your navigator**, since there's no normal sensors that can see what's going on. Without a navigator you really should plan your course before you jump, since you're literally flying blind.
  - **Finding and disorienting enemy *navigators* is a really great way to stop a fleet dead in its tracks.** You might not even need to interdict them. Once disoriented a fleet will need to either have a backup navigator or be really really lucky if they want to run, since you've usually caught them in deep space where nobody knows range precise enough to preplot a warp path.