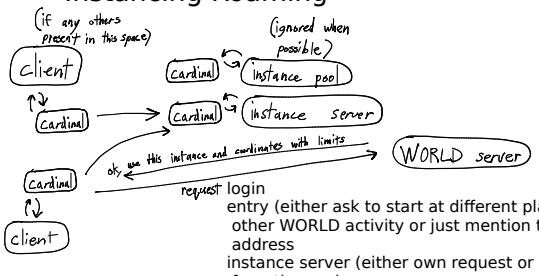
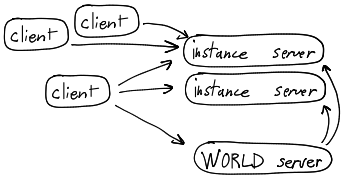
Algorithms

Instancing Roaming



entry (either ask to start at different place perhaps based on other WORLD activity or just mention the 'assign me whatever' address
instance server (either own request or 'assign me whatever' from the pool

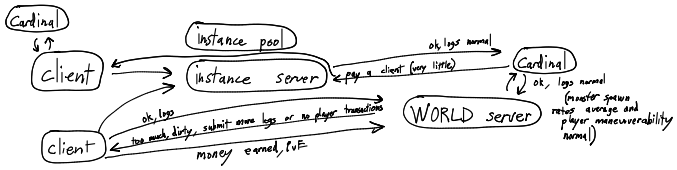


Clients connect to multiple instance servers when 'roaming' into nearby occupied spaces. In case of suspected impersonation, WORLD server connects to instance servers to request dropping of source IP addresses, signatures, names, etc, unassociated with recent authentic logins.

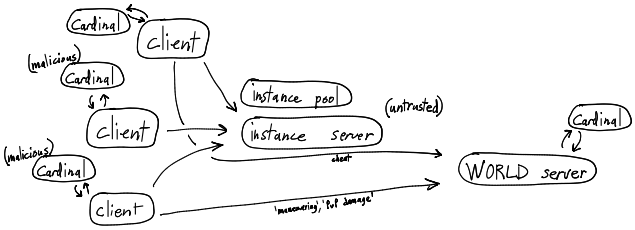
Players must have a visible indicator of connection quality from other players and which instance servers are connected.

Instance server bandwidth during popular events may be reduced by filtering traffic sent by players not causing PvP damage (ie. not participants in the contest), degrading such other players by reducing bones, F/s events, and if necessary disconnecting players for adding excessive traffic (eg. disconnecting any player adding a new 'microphone' to the already crowded space). 'Dumb' instance servers relaying a shared channel with no processing may force clients to degrade by dropping, delaying, or corrupting (by added analog noise) packets as a hint. Thus, duels and such may be observed by a large number of players in the same instance while server bandwidth is limited.

Anti-Cheat



Cheating PvE may occur with no other players present, or offline. When informed of supposed PvE progress, WORLD server may demand logs before acceptance.



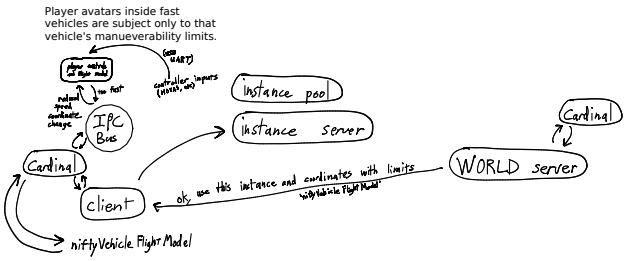
Cheating PvP between reputable players has much higher possible consequences, WORLD server may always require logs, other players are always present, and statistically detected cheating may be silently accepted for some random interval to prevent exploiting.

Statistical abnormality in logs or allegations of cheating, is cause to - after some random interval to prevent exploting - mark a a player 'dirty' for increasing durations with each offense up to 7years (5years being arguably short in place of permanent ban, 10years being arguably longer than serves any purpose, and some ability to catch 'Newton's Apple' proverbially implying some substantial multiples of that response time is enough for reasonable cognition to rationally change behavior).

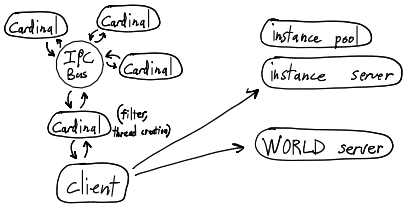
Players marked dirty may be unable to participate in many activities, may be reset, and may be reset more promptly for any suspicion of cheating. Newly created accounts may have similar limitations until sufficient proof-of-cognition has accumulated to mitigate rapid bypassing of such anti-cheat limits. Accounts marked 'dirty' for the maximum duration (7years) may be much further from participating than newly created accounts (months of limitations expected for most activities), and this is an intended effect.

Shared Channel

IPC Bus, UART Controller Peripherials, Flight Model Limits, Latency Compensation, Developer Interaction



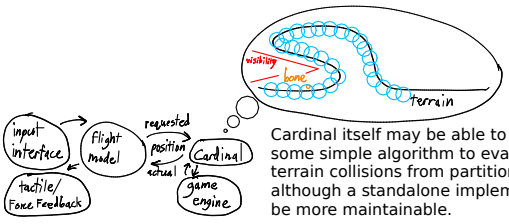
Player given program, taking input from player given controls (eg. HOTAS with USB UART having no 'DirectInput' or keyboard binding compability) may move player's vehicle 'playerItem' within manueverability limits from WORLD server. Position/rotation changes are sent to IPC Bus shared channel, forwarded by CARDinal, within Client, to instance server.



CARDinal is inherently multi-threaded, despite being a monolithic set of functions, due to the ability to call these functions over an IPC bus after calling the '_bin_IPC' function (otherwise disallowed).

Developers may interact directly with the IPC Bus.

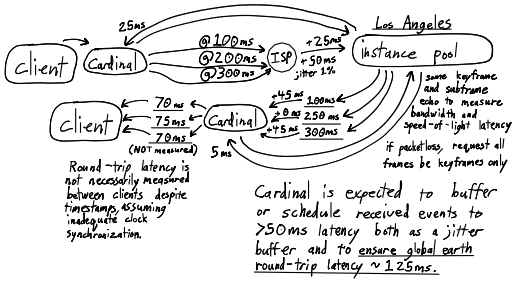
CARDinal always sends a 'playerItem' to a 'shared channel'. An instance server is expected to copy any acceptable 'playerItem' messages to all other connected players. An IPC bus is expected to copy all input to all outputs, which may change ad-hoc, exactly as a shared pair of wires hardware bus (ie. similar principle to CAN bus). Messages are NOT directed to specific recipients, as may be the case for some existing software bus implementations (eg. dbus, named pipes, etc). Suitable IPC bus reference implementations are part of 'ubiquitous bash'.



Cardinal itself may be able to include some simple algorithm to evaluate simple terrain collisions from partitioned space, although a standalone implementation may be more maintainable.

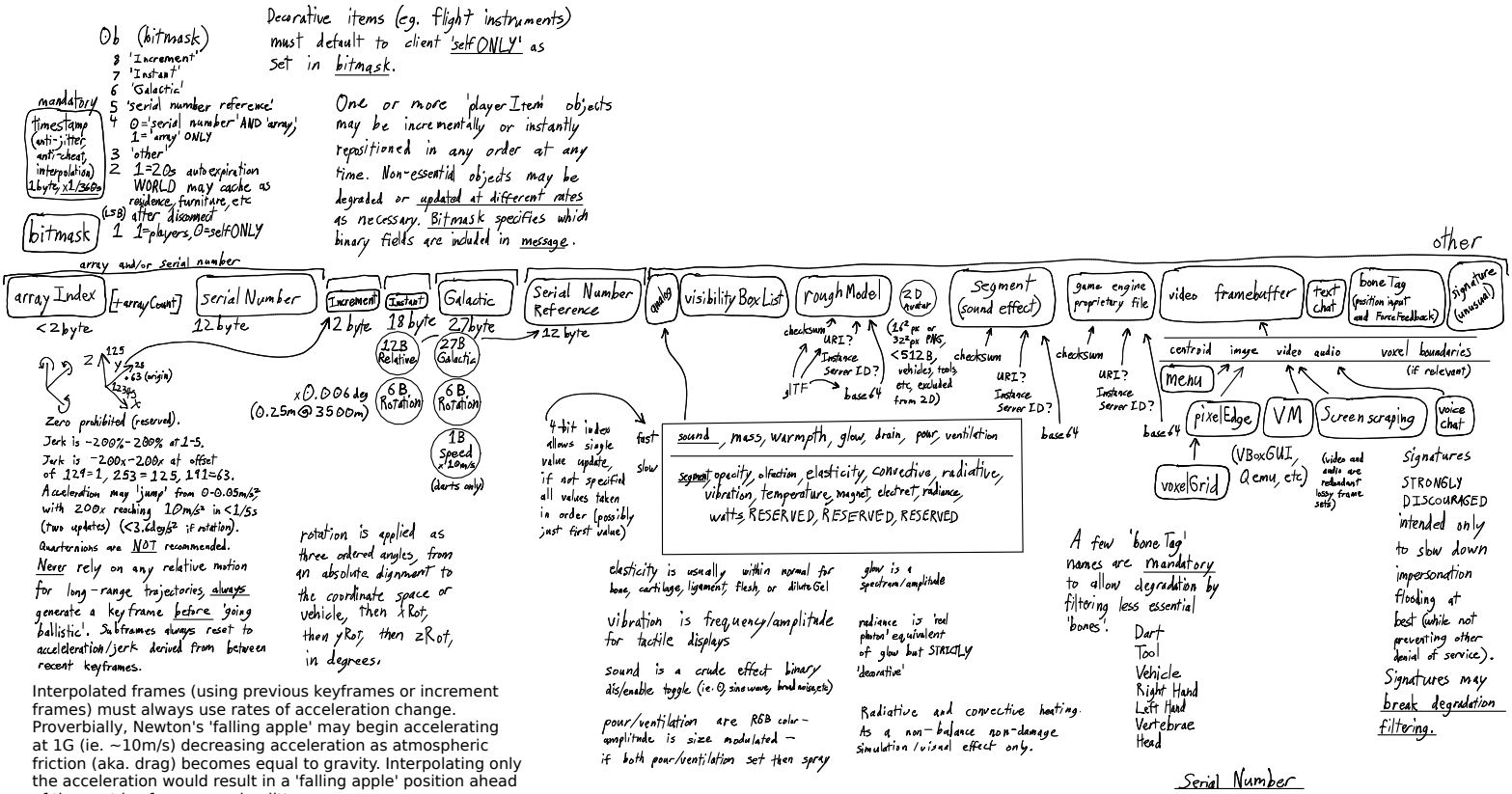
Rough terrain model may correct bone 'bounce', compensating for latency which may otherwise cause objects to appear below terrain. Especially prevents aircraft appearing to 'land undergound'.

Before PvP damage events occur, avatar/vehicle/tool/etc vehicle centroid coordinates may not be sent to prevent cheating (however, guessing with splash damage is allowed). To request other players announce their visibility, a player may declare their own presence in a semi-random announced visibility bounding box (which still prevents accurate visibility and aimbot cheating).

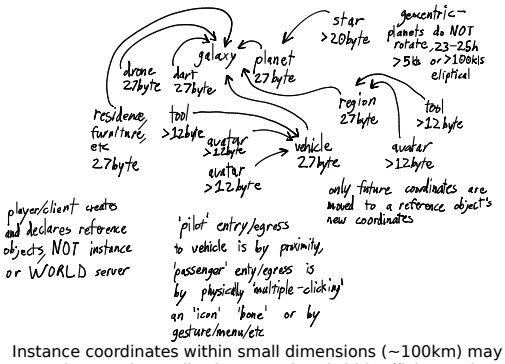


Latency compensation. A buffer must ensure latency remains approximately constant, rather than as little as possible, to mitigate jitter. Packet loss degrades acceleration rate change compression and must force keyframes only.

Cardinal playerItem Message Byte Allocation



Interpolated frames (using previous keyframes or increment frames) must always use rates of acceleration change. Proverbially, Newton's 'falling apple' may begin accelerating at 1G (ie. ~10m/s) decreasing acceleration as atmospheric friction (aka. drag) becomes equal to gravity. Interpolating only the acceleration would result in a 'falling apple' position ahead of the next keyframe, causing jitter.



Instance coordinates within small dimensions (~100km) may be referenced to 'region' markers, allowing more bandwidth efficient 12byte position keyframe coordinates. Similarly, a player avatar may use 12byte coordinates relative to a vehicle.

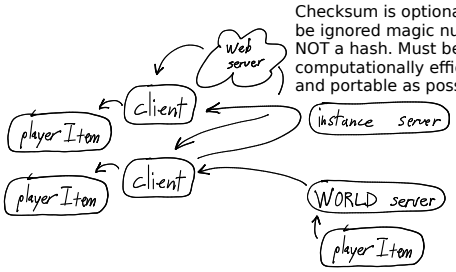
Vehicles must use 27byte position coordinates, as these may frequently move through ~100km boundaries, and players in vehicles will not gather as closely in large numbers as player avatars.

All objects near planetary surfaces must move when periodic astronomic position updates occur. If this is performed as a proximity rule, objects need not be marked or processed relative to planets.

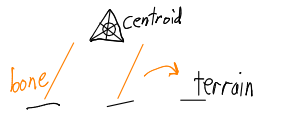
CARDinal may provide default rules to match location and prompt Task/Quest begin/intermediate/end 'playerItem', and PvE monster spawn 'playerItem'. Clients may alternatively encounter an interactive 'playerItem' which a local program is waiting for and manages. WORLD and instance servers have no part of Task/Quest/PvE events, progress of which is accepted on the basis of (if demanded) plausible logs.

Task/Quest begins as a 'playerItem' (possibly non-visible) which must be brought with a player avatar to a location, may include retrieving a 'playerItem' (possibly non-visible) from a location (eg. which may appear to the player as a conversion from a raw material to a manufactured material), and is completed when a 'playerItem' is brought to a location (eg. where the Task/Quest was given or immediately completed at the location where the 'playerItem' was retrieved).

Checksum is optional (may be ignored magic number), NOT a hash. Must be as computationally efficient and portable as possible.

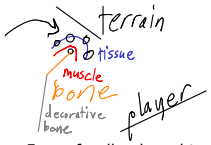


Special objects (eg. terrain, rough terrain, planetary surfaces, default vehicles, PvE items/monsters) may be available as 'playerItem' from WORLD server or mirror. Any real-time positioning, model, texture, or other changes, do NOT depend on WORLD or instance server - clients must *always* be *completely standalone* after login and instance 'relay' assignment. Large files (eg. terrain, planetary surfaces) must not be sent through instance server, instead should be downloaded by reference to an appropriate file server.



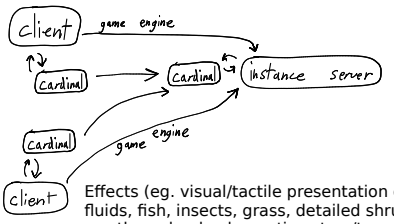
Paw bone collision event average distance and frequency are a better basis for player avatar centroid maneuverability limits. A velocity advantage is allowed for excellent footwork, whereas vehicles usually have less complicated acceleration limits from thrust/lift.

Paw placement patterns when repeated well, may be saved, replaying snaps to the original acceleration, speed, and accuracy. Such replayable patterns may be traded to other players (ie. as an 'Original Sword Skill' tradeable item mechanic). External programs may of course implement such replayable gesture snaps, modifying player avatar bone positions through IPC to CARDinal.



Force-feedback and tactile sense may be detected by collisions with non-shared objects. Game engines are apparently more able to detect such collisions than interactions between 3D model surfaces.

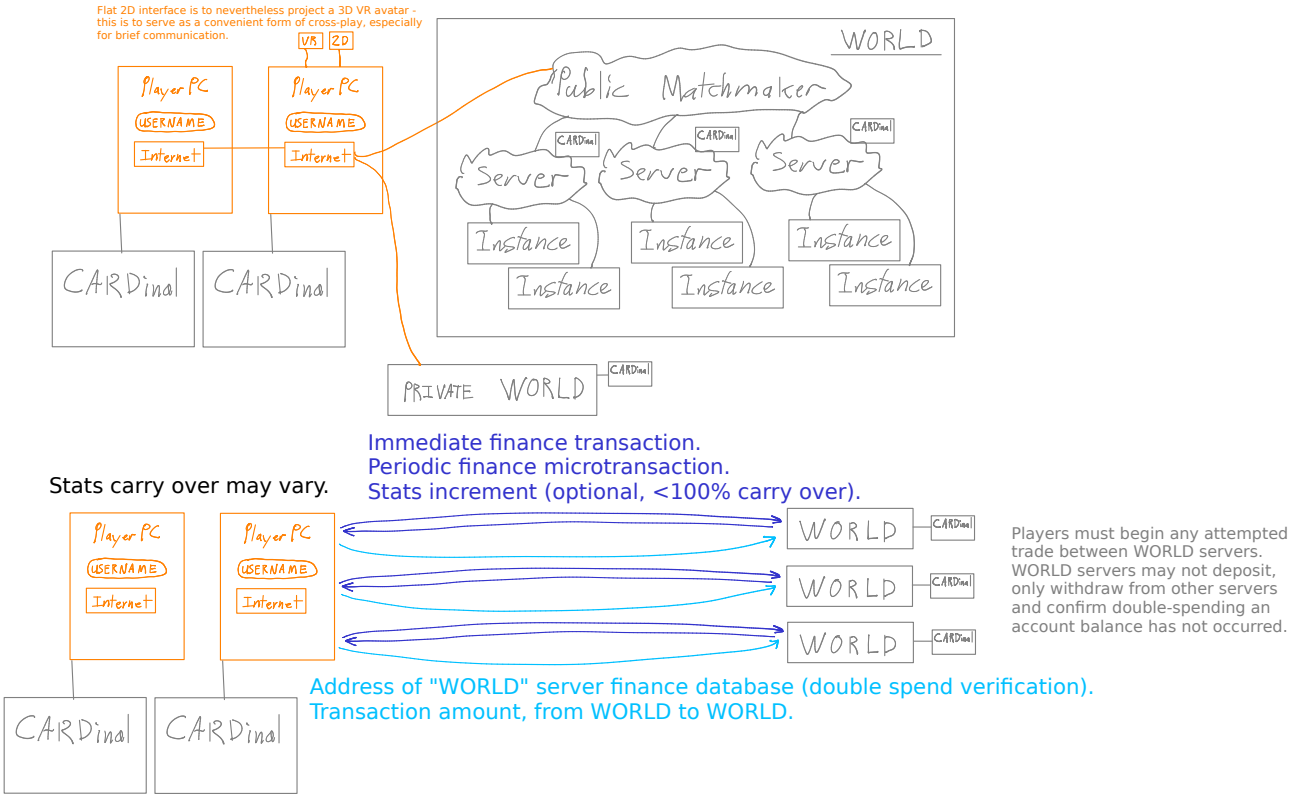
Bone position alone is enough for other players to impart force-feedback and tactile sense to other players. Composite structure from movable solid bone, through shaped elastic materials typical of muscle and any other tissue, well approximates physical contact between players, in multiplayer, without incurring the bandwidth and complexity of sharing and convolving surface deformation. Fewer than 80 bone positions are expected sufficient.



Effects (eg. visual/tactile presentation of fluids, fish, insects, grass, detailed shrubs, weather, clouds, decorative stars/terrain, etc) may be not only delegated to but also synchronized through client-specific game engine netcode. Avatars, vehicles, tools, etc, requiring balance, player maneuvering coordinates, player 3D models, coarse planet/star terrain, etc, must still rely on Cardinal exclusively and must endow multi-engine cross-play.

Trade between WORLDs

Trade between WORLD servers is expected to allow players of mostly one profession (eg. galactic space pilots) to purchase relevant services (eg. vehicle maintance) from players who mostly carry on their profession at another WORLD (eg. Swordland blacksmith). Stats matching between servers allows players 'proof-of-cognition' at one WORLD to establish their ability to participate in similar activities at other WORLDs.



Trade between CARDinal accredited and compliant WORLD servers, including both finance and stats progression, is equivalent to a 'cheque' banking system. Player issues a request to a WORLD server, to withdraw funds or match character stats from another WORLD server.

WORLD servers periodically publish a list of recent requests, confirm double spending of an entire account balance has not occurred, and publish a list of recent confirmed requests. When a WORLD server sees the transaction confirmed by the second list, the transaction is complete. All accredited WORLD servers publish all other WORLD servers recent lists to every other WORLD server. Such transactions are relatively rare, low bandwidth, and easily compressed files, swapped at most once every few tens of minutes.

Rate limits, <80% exchange rates, absence of deposit capability, and optionally any WORLD server imposed taxes, prevent economic volatility at one WORLD server from spreading to other WORLDs. Possibility of severe damage to player's account balances is also prevented. Attempting to deposit from multiple WORLD servers at a rate above limits of one WORLD server is precluded.

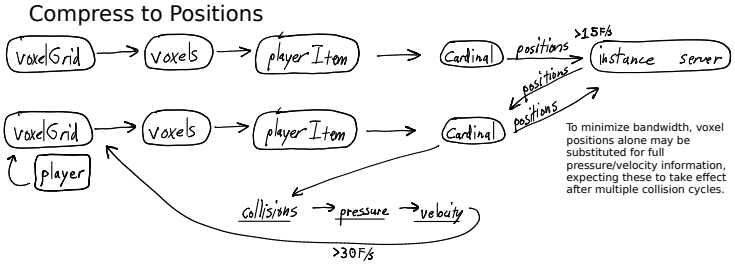
Compromised or malicious accredited WORLD server cannot deposit funds without a withdrawl request initiated by a player at other WORLD server, and only a small fraction of a player's financial balance or monthly earnings can be withdrawn from any WORLD server per month.

WORLD servers festering bigotry against player play styles (ie. severe inequality and lack of legitimate PvP activities resulting in PvE oriented players having no options for protection from unwanted random PvP hostility), otherwise inhibiting diversity and quality of life, or offering unintended brokering services to trade valuables beyond expected limits, will be noticed and lose accreditation.

Algorithms

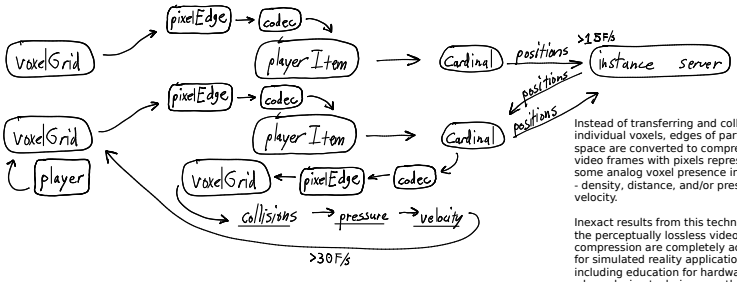
Simulated Reality (Voxels)

Compress to Positions



To minimize bandwidth, voxel positions alone may be substituted for full pressure/velocity information, expecting these to take effect after multiple collision cycles.

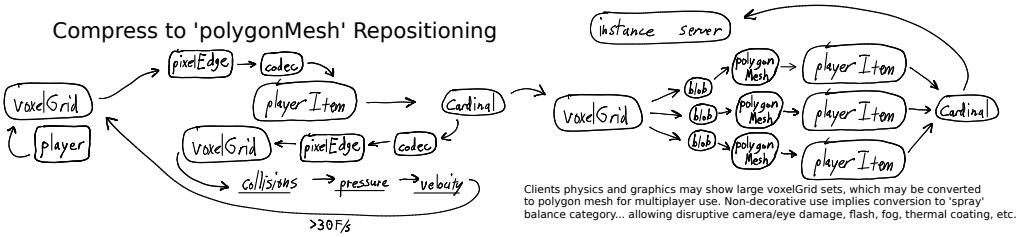
Compress to 'pixelEdge' Video Frames



Instead of transferring and colliding individual voxels, edges of partitioned space are converted to compressed 2D video frames with pixels representing some analog voxel presence indication - density, distance, and/or pressure, velocity.

Inexact results from this technique or the perceptually lossless video compression are completely acceptable for simulated reality applications, including education for hardware design, where design techniques, rather than exact designs, are most relevant.

Compress to 'polygonMesh' Repositioning

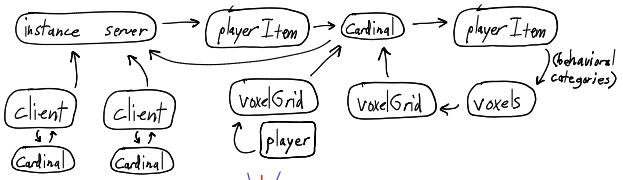


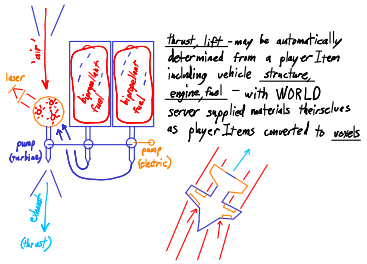
Clients physics and graphics may show large voxelGrid sets, which may be converted to polygon mesh for multiplayer use. Non-decorative use implies conversion to 'spray' balance category... allowing disruptive camera/eye damage, flash, fog, thermal coating, etc.

Sharing voxels brings 'simulated reality' capabilities, rather than 'virtual reality' at a cost in bandwidth, hardware computation, and software development, comparable to the computations performed by cognition that would use such 'simulated' or 'virtual' reality. Approximately 100Mbit/Second per player bandwidth expected from voxel grids as cubes with sides as video frames, rather than the ~100kbit/Second per player from repositioning a polygon mesh.

'Simulated Reality' using voxels is worthwhile, given the substantial benefits.
*) CAD modeling by sculpture, manufacturing methods, or iterative sense of correctness from neural interface.
*) Fluid physics simulation as deterministic large numbers of consequential events instead of decorative visual effect converted to few consequential events.
*) Vehicle thrust/lift analysis.

'Virtual Reality' also benefits from some fluid physics simulation, with at least local force feedback, and some consequential events (eg. flooding a previously habitable area with liquid).





thrust, lift - may be automatically determined from a playerItem including vehicle structure, engine, fuel - with WORLD server supplied materials themselves as playerItems converted to voxels