SandBox Wars

Game Draft

Table of Contents

Development Tasks

Tasks to be completed by the end of August 2015.
Nested task may be optional for now.

- ☑ Unity + Git development environment setup (2015-08-05)
- ☑ Networking (2015-08-06)
- ☑ Walking in the map (2015-08-07)
- ☑ Picking up items (2015-08-08)
- ☑ Shooting/Attacking (2015-08-19)
    - ☑ Bullet entities (2015-08-18)
    - ☐ Client side player position extrapolation
    - ☑ Server side player/bullet interaction (2015-08-18)
    - ☐ Retro-checking of player positions
- ☐ Killing other players/respawning
    - ☐ Creatures health
    - ☐ Despawning

Brainstorming Ideas

Handling Player Constructions

One planet with quests, other planets for players to colonize.

Government, managed by the computer, with quests, players cannot build here.

Possibility to ask the government for a plot to build a house (cheap, but only one per-player).

Possibility to ask for a bigger space (higher cost) to get resources and build bigger structures, players get interest in building something cool.

Unlocking Blueprints Ideas

Different ideas to unlock blueprints:

1. Every player finds it's own blueprints. There is no wiki as there is an extremely big amount of equivalent items.
   Experience required to unlock blueprints, experience improves as the player interacts with the environment; when the player reaches an high enough familiarity level with an item, it's blueprint is unlocked. Simple items unlock faster than rare items. The player doesn't know what his experience level is.
2. Quests unlock new items.
   Example: hidden items in secret quests.
   Example: NPCs reveal the blueprints (depending on player's behaviors, like the type of weapons used).
   Example: examine items in acceptable conditions to create blueprints.

Item behaviors

Different ideas about item behaviors:

1. Player doesn't know anything about items but the name and eventually the category.
2. The player doesn't know anything about items until used for the first time, stats are then revealed.
3. Every item has a known and a secret set of stats.
4. Players don't know anything about items, but stats are progressively revealed and improved as the item is used more and more.

Lag compensation

Reflections on lag compensation systems.

Terms:
Client = The machine connecting to the server
Peer = Another machine connected to the server
Server = The machine serving all peers (including the client)

Latency values in example:
Client: 100ms
Peer1: 75ms
Peer2: 250ms

Client sends movement to the server; the server instantly updates the simulation and saves the position of the player in it's time buffer with time SERVER_TIME - CLIENT_PING (so that in the future it will know where the player was at this time) and informs peers of the new situation; peer1 tries to display the player ((100 + 75) / 2)ms ahead of where it should have been because that's where is probably is on the client's screen; peer2 displays it ((100 + 250) / 2)ms ahead of where the server reported.

To not penalize low-ping players in favor of high-ping players, the simulation is actually run back in time, not actually at SERVER_TIME, but at SERVER-TIME - RMS(CLIENT_PINGs).
So if a low-ping player receives a packet ahead-of-time it can hold it until it becomes valid (and thus play with vritually zero-lag; the only difference is that every command is actually executed RMS(CLIENT_PINGs) time later of when it's issued.

When a player fires a bullet, the bullet is simulated back-in-time too (up to a reasonable amount; a 500ms ping player will not fire a bullet 500ms back in time, a reasonable amount could be RMS(CLIENT_PINGs)*2), the server looks back where the characters were in the timeframe of the bullet, and if any of the characters is hit, the effect are forward-propagated to current server time simulation.

(NOTE: this chapter is very confused, I'm not very sure about the things I've written, it will probably have to be reviewed intensively.

RMS = Root mean square = sqrt(sum(CLIENT_PING ^ 2) / CLIENTS_COUNT)

References:
https://developer.valvesoftware.com/wiki/Latency_Compensating_Methods_in_Client/Server_In-game_Protocol_Design_and_Optimization