OmniTanks Design Notes

Concept

A cooperative multiplayer Player Vs Environment (PvE) class based vehicle shooter with mission based game play.

Players control an omnidirectional tank based off one of three chassis types (light, medium, or heavy) with fixed equipment loadouts. Players work as a team to complete a mission objective for each session.

Player death does not remove the player from the session. Dead players will be able to respawn in specific safe-ish areas based on the mission type after a respawn cooldown. The cooldown time is based on chassis type with the lighter (and more fraile) tanks being able to respawn faster, since they will probably die more often.

Tanks

Tanks can move in any direction at the same speed due to having omnidirectional wheels. Tanks have a turret that mounts two weapons that can be aimed up and down along a limited arc (larger upward arc than downward). 4 direct axes of controls are available to the player.

- Forward / Backwards translation (moves along the view vector)
- Sideways translation (moves perpendicular to the view vector, or sidestep)
- Rotation (rotates the tank and the turret together)
- Vertical Aiming (only rotates the weapon turret up and down).

This is standard FPS shooter mechanics.

Tank Chassis Type Overview

Light

The light chassis has the fastest base speed in the game, but carries the least amount of equipment and has the weakest armor.

Medium

The meidum chassis is a well rounded platform providing decent speed and equipment. This is considered the “standard” chassis.

Heavy

The heavy chassis has the slowest base speed but can carry the most equipment in the game and is the most survivable.

Components

All tanks have a component loadout that is selected outside the game world. The choice of loadout and chassis type allows players to build combinations that fit their most desired play-style.

All tanks have a capacitor. Weapons and active accessories use power from the capacitor. Power regenerates over time. Capacitor size and recharge rate is determined by class. Some passive accessories can increase capacitor size or charge rate.

All tanks have an armor hit point value. This is the tank's health. Hit point values are determined by chassis class. Some passive accessories can increase the base hit points for a tank.
Damage to the tank will lower the armor value. When it reaches 0 the tank will explode and start a respawn timer. Respawn time is determined by chassis class. A tank's armor value only increase in the following situations.

- The tank runs over an armor pickup in the field.
    - Armor pickups are dropped randomly from killed enemies
    - Armor pickups restore a fixed number of armor hit points.
- The tank is repaired by a teammate that has a repair accessory
    - Repair is hit points over time, the longer the repair is active the more hit points will be restored.

All tanks have at least one main weapon and one alternate weapon attached to a turret. The turret can aim up and down. There is no need for the turret to rotate since the tank can move in any direction. The types of weapons that can fit in each slot is determined by chassis class. Weapons use capacitor charge when fired. Weapons can not be fired if their is not enough power in the capacitor. Each weapon has a minimum cycle time that is independent of capacitor charge. Some heavy weapons use special ammunition in addition to charge. Special ammunition is picked up in game by killing enemies. The amount of special ammunition is limited, and varies by weapon type. A tank can only carry one weapon that uses special ammunition. Passive accessories can increase the amount of special ammunition a tank can carry.

All tanks have at least one slot for an accessory. Accessories are either passive or active. Light chassis tanks can mount a single accessory that is passive or active. Larger tanks can mount multiple accessories. No tank can have more than one active accessory.

Passive accessories provide some bonus all the time, usually in the form of an enhanced ability or stat (health or capacitor).
Active accessories must be triggered by the player in order to be used. Active accessories use capacitor charge when active. Active accessory effects can not be used if there is not enough power in the capacitor.

All tanks pick a chassis and an equipment loadout before the game starts. Loadouts may be changed during respawn at the cost of double respawn time. Chassis type may not be changed during the game.

Chassis Types

Class	Speed	Capacitor	Hit Points	Accessories	Respawn
Light	25m/s	50j @ 10j/s	50	1 (P or A)	2s
Medium	15m/s	100j @ 15j/s	100	1P 1A	5s
Heavy	8m/s	200j @ 20j/s	150	2P 1A	10s

Weapons

Name	L	M	H	Charge	Special
Blaster	M	M	A	10	
Cannon	A	M	M	20	
Laser		M	M	30	
Machine Gun	M	M	A	2	
Chain Gun		M	M	5	
Target Designator	A	A	A	1	
Shotgun	M	A	A	8	
Mortar			M	2	10
Rockets		A	A	1	25
Repair Beam	A	A	A	25	
Repair Field		A	A	30	

Grenade	A	A	A	5	
Heavy Grenade			M	10	5
Missiles		M	M	1	10
Directional Shield		A	A	25	

Accessories

Name	Effect	Charge	Type
Improved Motors	Increase Speed		P
Larger Capacitor	More charge		P
Improved Generator	Faster charge		P
Improved Armor	More Hit Points		P
Shield Projector	Prevent damage around tank	25/s	A
Ammo Rack	Increased Special Ammunition		P
Turbo Boost	Increased Speed when active	10/s	A
Jump Jets	Limited Jumping	15/s	A
Radar	See through objects	5/s	A
Scavenger	Increases pickup radius		P
Zoom	View zoom and increased damage	2/s	A
Jamming	Disrupts enemy sensors in radius	5/s	A
Drone Turret	Drops a single AI turret (w/ cooldown)	50	A

Equipment Details

Blaster

Low grade plasma bolt. Fires in a straight line. Does small splash damage. Relatively high rate of fire. Moderate Range.

Cannon

Larger Plasma bolt. Fires in an arc. Does larger splash damage. Lower rate of fire. Medium-long range.

Laser

Focused energy beam. Fires in a straight line. Does damage over time. Beam must be held on target. Does zero splash damage. Long range.

Machine Gun

Rapid fire blaster. Fires in a straight line. High rate of fire. Small splash damage. No Spin up time. Moderate Range

Chain Gun

Highest rate of fire for any weapon. Higher damage per round but less splash damage. Shorter range. Has spin up time. Medium-Short Range.

Target Designator

Does no direct damage to target. Marks target as destination for mortar and missiles fired by teammates. Target does not know it’s being designated. Long Range.

Shotgun

Short range, wide cone of fire. High splash damage. Moderate rate of fire.

Mortar

Uses special ammunition. Fires a very heavy cannon shell in a large arc. Will home in on targets that are flagged with designator by teammate (can not self designate). Extreme splash damage, and burn damage over time. Slow rate of fire. Long range.

Rockets

Uses special ammunition. Fires a volley of high velocity unguided rockets along an arc. High splash damage. High rate of fire. Medium long range.

Repair Beam

Fires a repair beam to damaged target nearest to weapon crosshair, within a cone. Medium/short range. Can not repair through objects or world geometry. Must have line of sight on target. Undamaged teammates are ignored. Repaired hit points are at a fixed rate over time and not percentage based. Heavy chassis take longer to repair.

Repair Field

Repair all damaged teammates in radius of tank. Will repair through objects and world geometry. Smaller range than repair beam. Will self repair tank.

Grenade

Fires an explosive plasma ball on light and medium chassis installations. Medium range.Explosion detonates on proximity to enemy or contact with ground. Does medium splash damage. Moderate rate of fire. Large chassis installation uses special ammunition and fires a cluster grenade that will bounce off objects with a larger spread and more splash damage.

Missiles

Uses special ammunition. Fires a single guided missile at the nearest target or targets marked with laser designator. Missiles are slower than rockets but more maneuverable than mortars. Can be self designated or have targets changed after launch. High splash damage. Medium rate of fire. Long range.

Directional Shield

Projects a tall shield wall in front of the tank, preventing all damage from that direction to all teammates behind the wall. Wall moves with tank. Teammates can fire back through the wall without issue. Hits to the shield will drain power faster. Enemy tanks can not pass through a shield. Player tanks can. Enemy tanks that are on a shield boundary when the shield is erected will be pushed away from the shield generator. Enemy tanks that are fully inside the projected shield are not affected. The shield can be used to push tanks away. Tanks that can not be pushed away due to world or object geometry will take damage over time.

Improved Motors

Increases tank speed in all directions by some fixed percentage. Can be only be picked once on heavy chassis.

Larger Capacitor

Increases the power capacity of the tank by some fixed percentage. Can selected multiple times on heavy chassis.

Improved Generator

Increases the power recharge rate of the tank by some fixed percentage. Can only be selected once on any chassis.

Improved Armor

Increases tank hit points some fixed percentage but lowers speed. Can be picked multiple times on heavy chassis for double speed penalty.

Shield Projector

When active, projects a shield bubble around the tank to block damage. Teammates in radius are protected as well. Hits to the shield drain power faster. All rules from Shield wall are also applied to the projector just in a 360 degree bubble.

Ammo Rack

Doubles special ammunition capacity.

Turbo Boost

When active doubles speed of tank, including bonuses from improved motors.

Jump Jets

Allows the tank to jump into the air and fly for a short period. Flight time uses power. When jets are disabled in the air tank will fall uncontrollably.

Radar

Shows outline of enemy tanks behind objects in short to medium range when active.

Scavenger

Increases the radius that tank will pickup armor and special ammo drops.

Zoom

When active view zooms 3x and all shots are 1.5x damage.

Jamming

Blocks enemy sensors in short range when active. Will cause targeting errors in all enemy types.

Drone Turret

Drops a drone turret behind the tank. Drone turret can not move and has its own hit points and can not be repaired. Drone will attack all enemies in medium range with a standard blaster for 60 seconds or until destroyed, or another drone is doped. Drones can only be dropped every 30 seconds. Dones can be protected by shield walls and domes. A tank can only have one drone active at a time.

Mission Types

Several mission types will be available. Mission type will determine the maps that can be used and are always selected before the game starts. Missions can have configurable minimum player counts, including single player.

Destroy Target

Team starts outside of target area on the map. Must infiltrate and destroy one or more objects in the target area. Target objects can be static objectives, enemy boss tanks, or both. Mission fails if all team tanks are destroyed. Respawns can happen only as long as one team member is alive.

Capture Point

Team starts outside a target area and must infiltrate. Team must get to a specific part of the map and defend it against waves of enemy tanks, including boss tanks, for a specific number of waves or a specific time. Defense point may be an area to deny enemy or a destructible object that must be protected. Mission fails if target is destroyed or occupied by enemy for too long.

Defend Point

Identical to Capture Point but team starts in the area at mission start.

Retrieve Object

Team must infiltrate a target area, collect an object and bring it to some extraction site. Team member carrying objects can not fire and may be slowed. Mission fails if enemy tanks acquire or destroy the object before it is taken to the designated site.

Escort

Team must defend/protect a mobile object as it makes its way to some designated area on the map. Enemy tanks will be waiting in ambush and/or attack in waves. Mission fails if escorted object is destroyed.

Loot Modes

Missions can be in one of two loot modes.

Cooperative (default)

Armor and Ammo drops from killed enemies are duplicated for each player in an area. The values of the drops will be lower but each teammate will be able to pick up the drop. This spreads the drops around to all team members in a fair, but slower building way.

Free for All

Drops are singular but have higher values. They are picked up by the first player to grab them and are not shared. This gives larger individual drop values but can allow players to hoard drops if not managed well.

Matchmaking ,Team Gathering, and Communications

Players can choose to start a mission and wait for teammates or join a mission that is looking for teammates. Players that create a mission are given a mission code that they can give to friends so they can all join the same mission. Missions can also allow random players to be matched into the mission if desired. Players are shown all pending missions that are open to them when they are at the join mission screen.

Players are encouraged to register with the system but all new players will be given a temporary ID when they first start the game. Temporary IDs will be limited in what public missions they can join, and must complete a single player tutorial mission before they can join any open public mission that allows unregistered users. Temporary IDs are bound to a key that is stored on the user's system to attempt to give them keep the ID the same until they choose to register. Server side infrastructure may also use IP addresses and hostmasks to force temporary IDs to stick with a player across game sessions. Unregistered users will always be able to join a mission they have a join code for. Unregistered users can not select their player names or customise cosmetics/colors, all visuals are pre-selected by the matchmaking system. Registration allows players to customize cosmetic parts of tanks and select specific colors (Primary, secondary, accent colors). When a user registers they keep all data that is associated with the temporary ID (games played, tutorial status, etc..)

In game communications will be based around preset lists of signals and emotes that are shown as icons in the hud and are thus translatable to any target language. Text chat will be available only to registered users. Unregistered users can see chat from registered users they just can not respond with text, only preset responses. The rate an unregistered user can issue response is limited.

Voice chat will be provided by external third party applications (discord, skype, teamspeak, etc…)

Registered users can create friends list. Friend requests must be approved by both sides. Friends and their statuses are shown in a list on the main setup menu and when a user sets up a mission it has the option to invite online friends. Friends may chat with each other in the lobby menu. Groups of friends may register a group chat channel for use in game (lobby and and during missions). The chat system will not store offline history for friend or group chat.

When teammates have all joined a mission it will move to the loadout phase. All users are asked to select a chassis and loadout. Registered users may store loadout presets. The currently selected loadout for all teammates is shown to everyone in the mission during loadout selection to allow for team balance. Mission starts when all team members have finalized

loadout selections, or a loadout time limit has been reached. The player that started the mission (mission leader) may kick any player from the mission. Players may vote to kick any player from the team during loadout by a ⅔ vote of registered users. This includes the player that started the mission. If the mission leader is kicked, then the first person to join the mission will become leader.

The same kick system is in place during the game as well , ⅔ vote or leader may remove any user for any reason. If the player count drops below the minimum requirement then the mission will be scrubbed and terminated. Scrubbed missions do not count as a failure for the users who were not kicked. If a user is kicked offen, kicks may be weighed into the mission success ratio. All users, even unregistered users have a mission success ratio that is visible by all other users. If a player joins a mission that is lead by a player that has kicked them recently, they will be asked to confirm that they wish to join the mission.

Default Loadouts

Default loadouts are designed to be easy to understand for a new player. Weapon selections do not require complex mechanics and active accessory is always just a boost (like sprint in a FPS).

Chassis	Main	Alternate	Passive	Active	Passive 2
Light	Blaster	Grenade		Turbo Boost	
Medium	Machine Gun	Shotgun	Scavenger	Turbo Boost	
Heavy	Chain Gun	Rockets	Scavenger	Turbo Boost	Ammo Rack

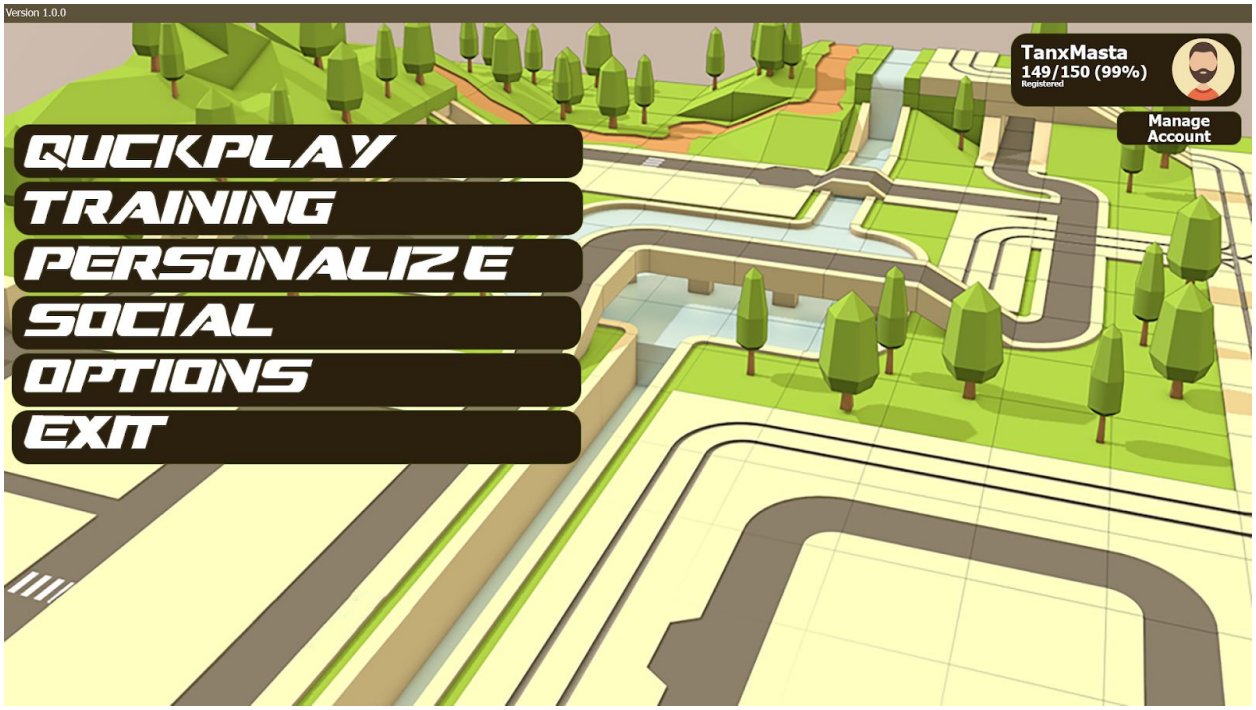
Role Presets

Role presets are premade chassis and loadout selections that are meant to fill a specific role in a team.

Name	Chassis	Main	Alternate	Passive	Active	Passive 2
Scout	Light	Shotgun	Target Designator		Improved Motors	
Support	Medium	Blaster	Repair Beam	Larger Capacitor	Shield Projector	
Sniper	Medium	Laser	Rockets	Improved Generator	Zoom	
Tank	Heavy	Canon	Machine Gun	Improved Armor	Shield Projector	Improved Generator

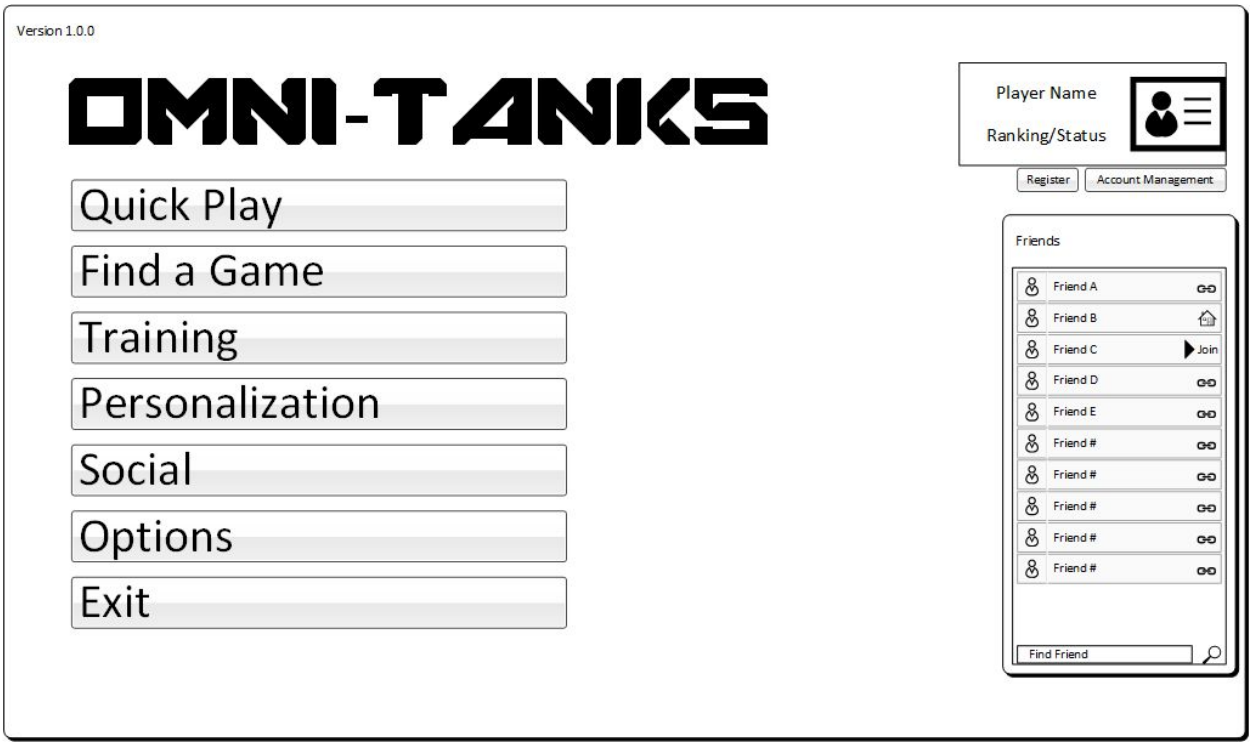
Menus

The game will have several menus in a tree that allow the user to get into a game easily. The menus will feature a background image or 3d scene that is updated from the master server when needed, or based on seasons.



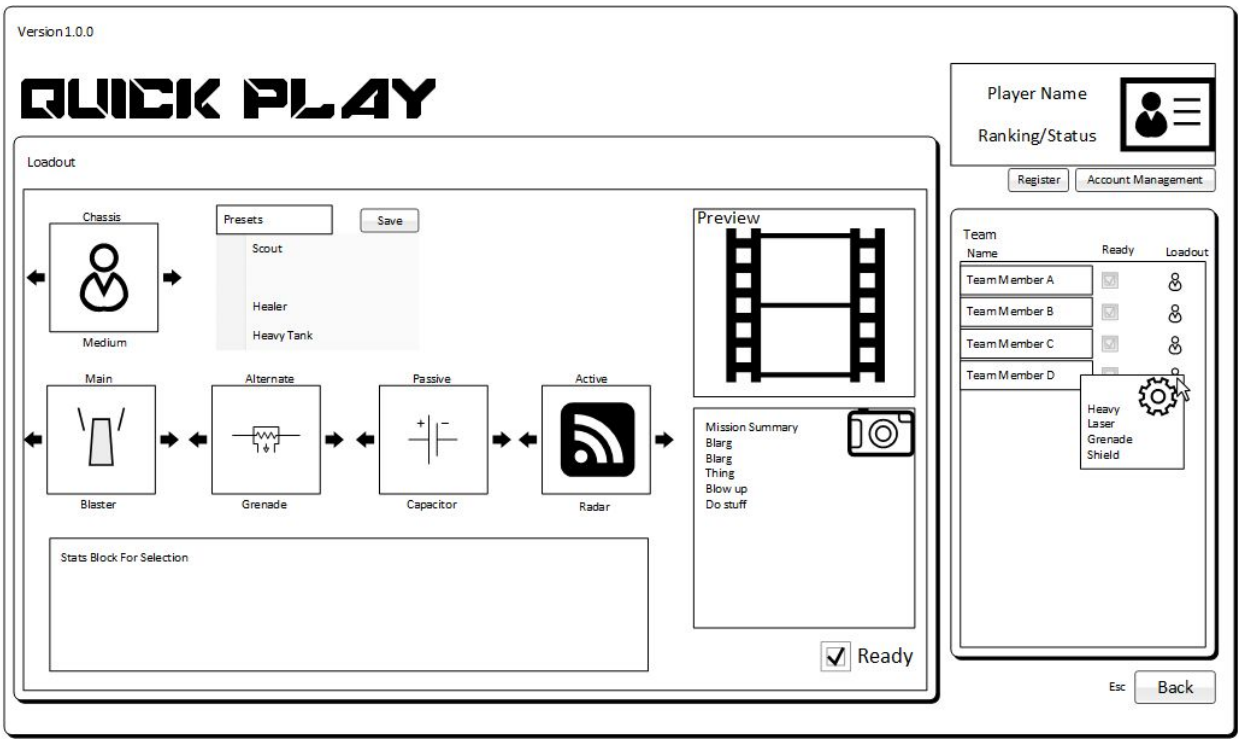
Possible Style Layout

Main Menu



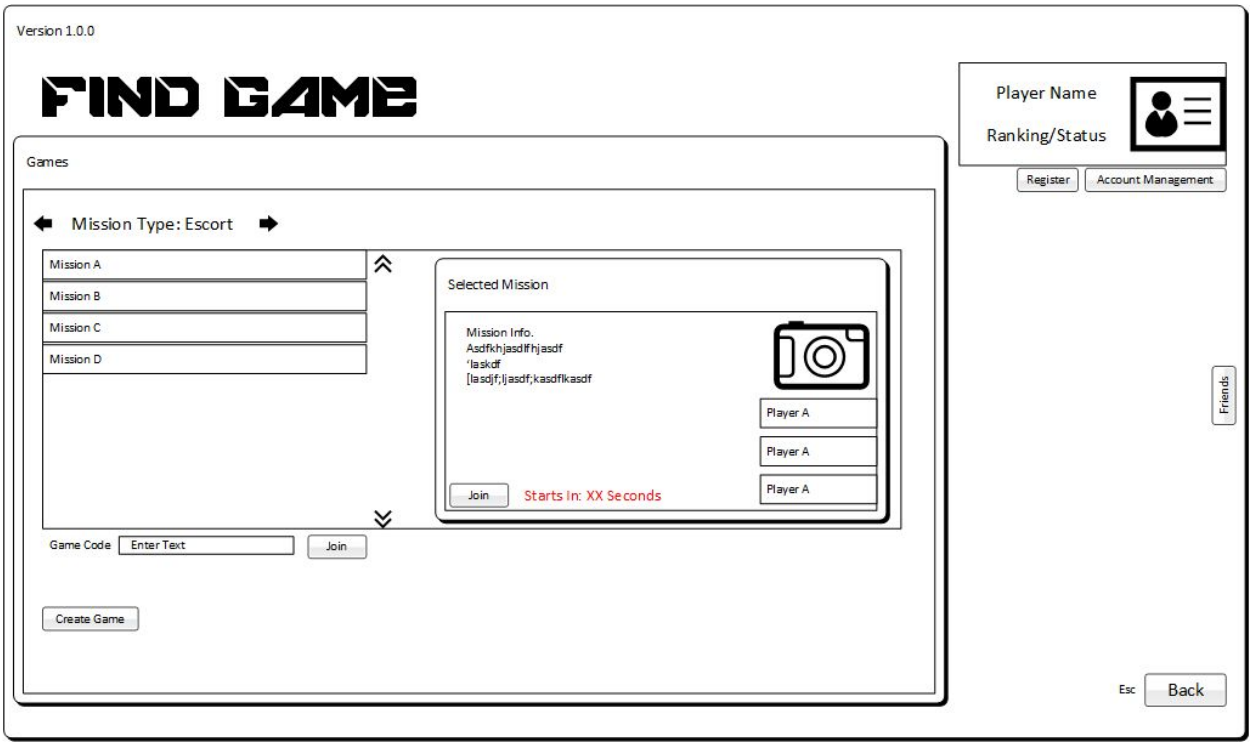
The main menu contains all the primary controls and sub menus to get into a game. A friends list is shown on the right hand side of the screen and can be folded away as needed. When folded a friends tab will be available on all menu screens except load-out.
The upper left of all menus contains the current user's avatar information and account management options. Accounts can be registered and changed from these sub dialogs. Registration is instant, but will require email verification. Some services may not be available until email is verified.

Quickplay



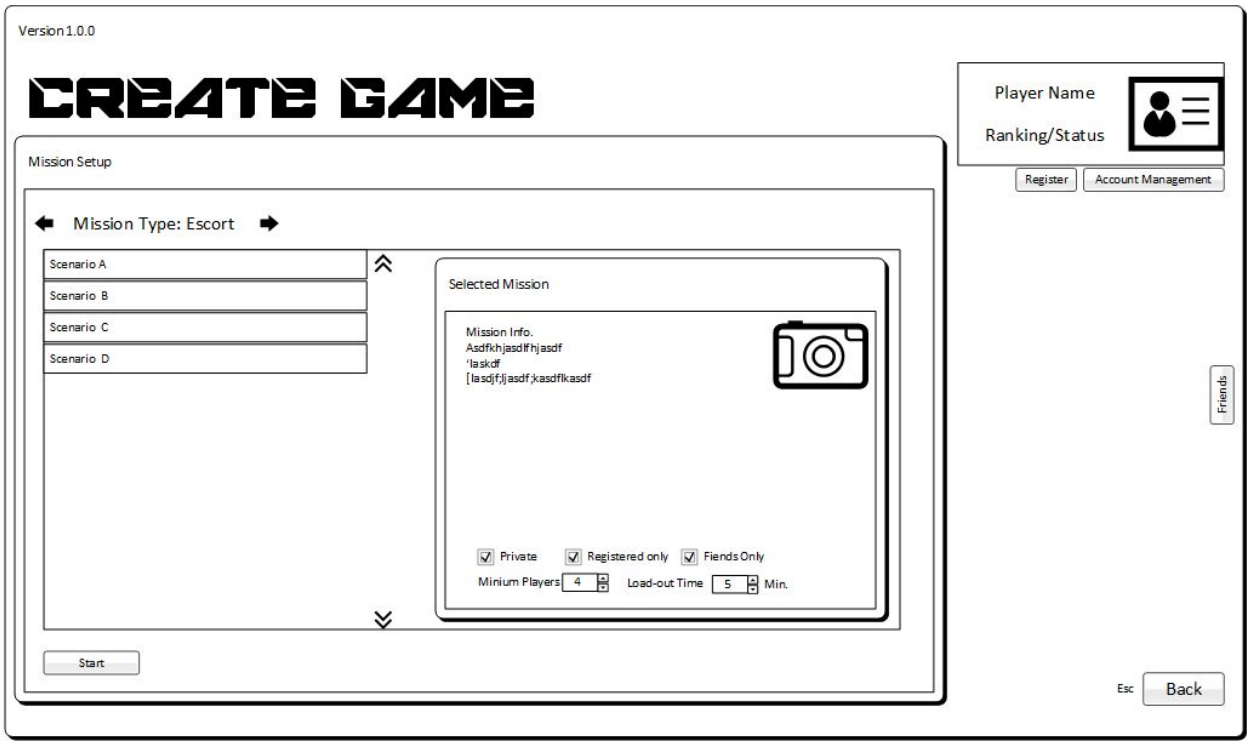
Quick Play is a shortcut that takes the player directly to a random game that is waiting to be filled that can accept the user (based on registration/friends status). Central services may force this quickplay to be a single player instance if player counts are low or no games are waiting. The user will be given the option to wait some time, or go back to the more advanced find game. See the load-out section for more info on rest of this menu.

Find Game



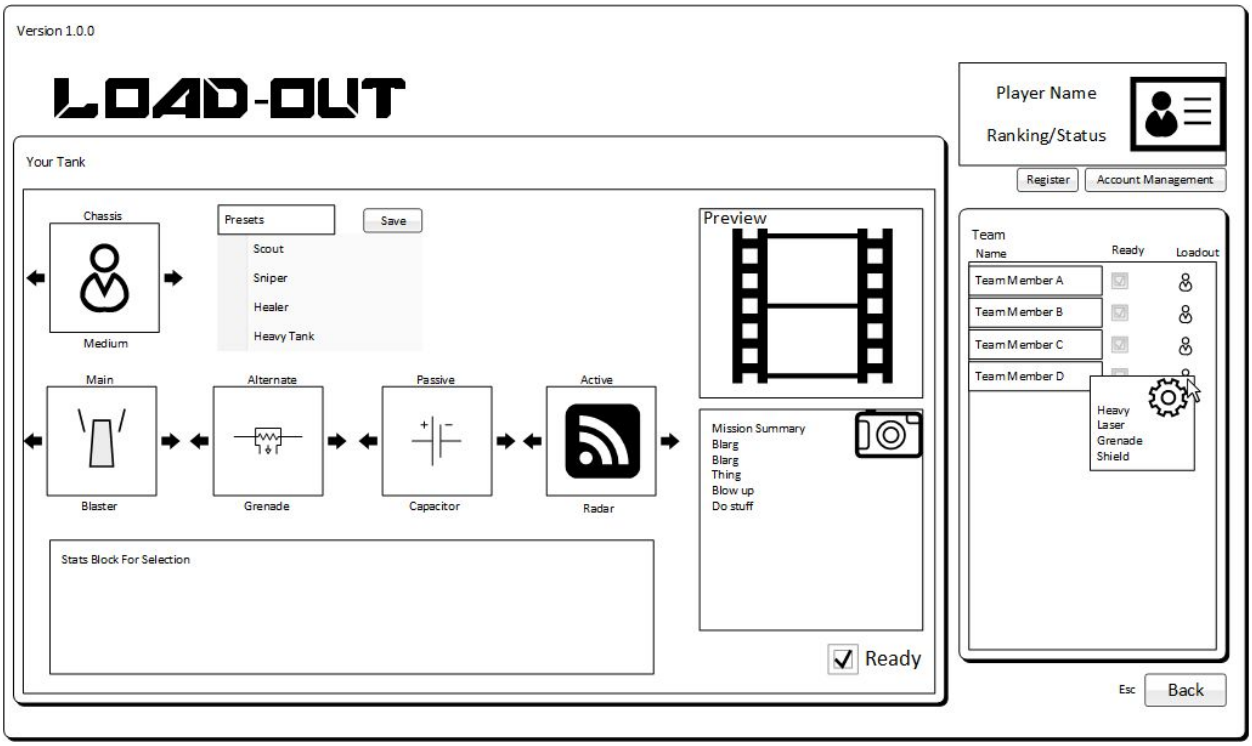
Find Game allows a user to locate a game that is waiting for players. All available games are shown. Information about the mission and its settings can be viewed. If a user has a game code they can use it to directly find a game. Joining a game takes the player to the load-out screen. Optionally from this screen a user can invoke the create game menu.

Create Game



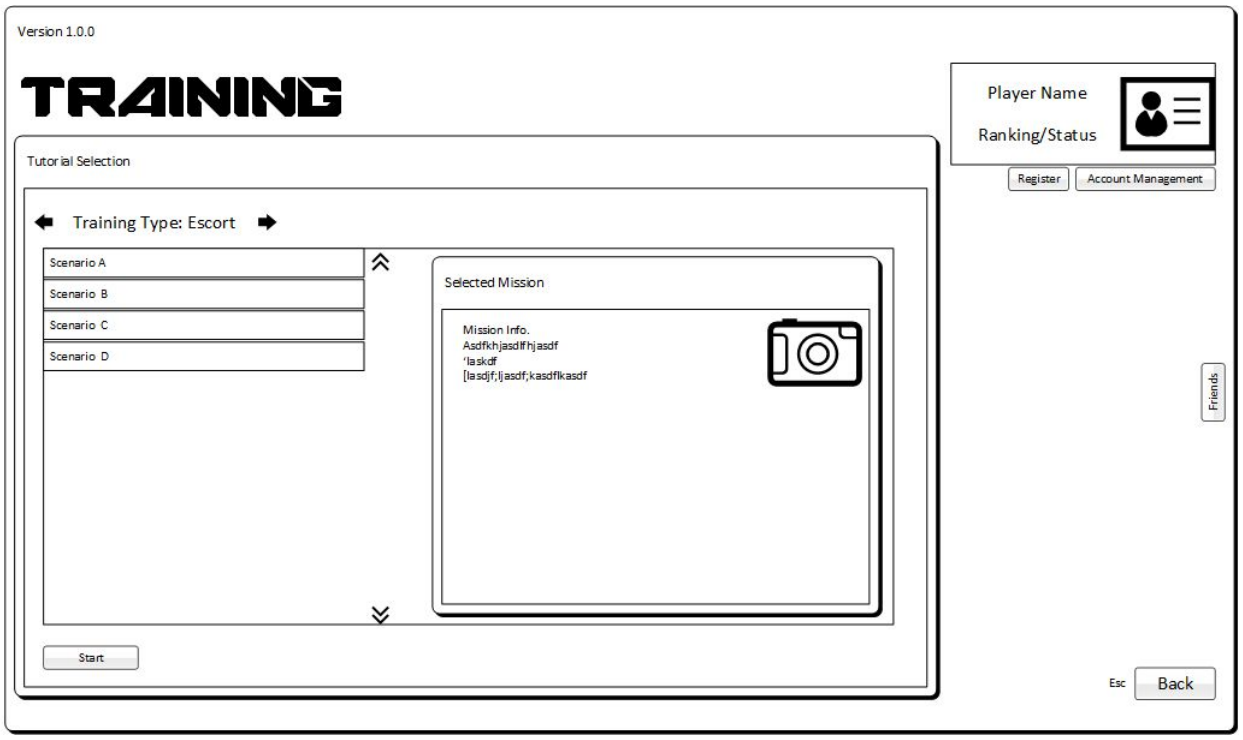
Create Game allows a user to create a new game using preset mission scenarios. A scenario includes a mission type, objectives and map. Games can be made private, for registered players only, or for friends only. Scenarios may have other specific settings that are shown here. Creating a game takes the player to the load-out screen.

Load-out



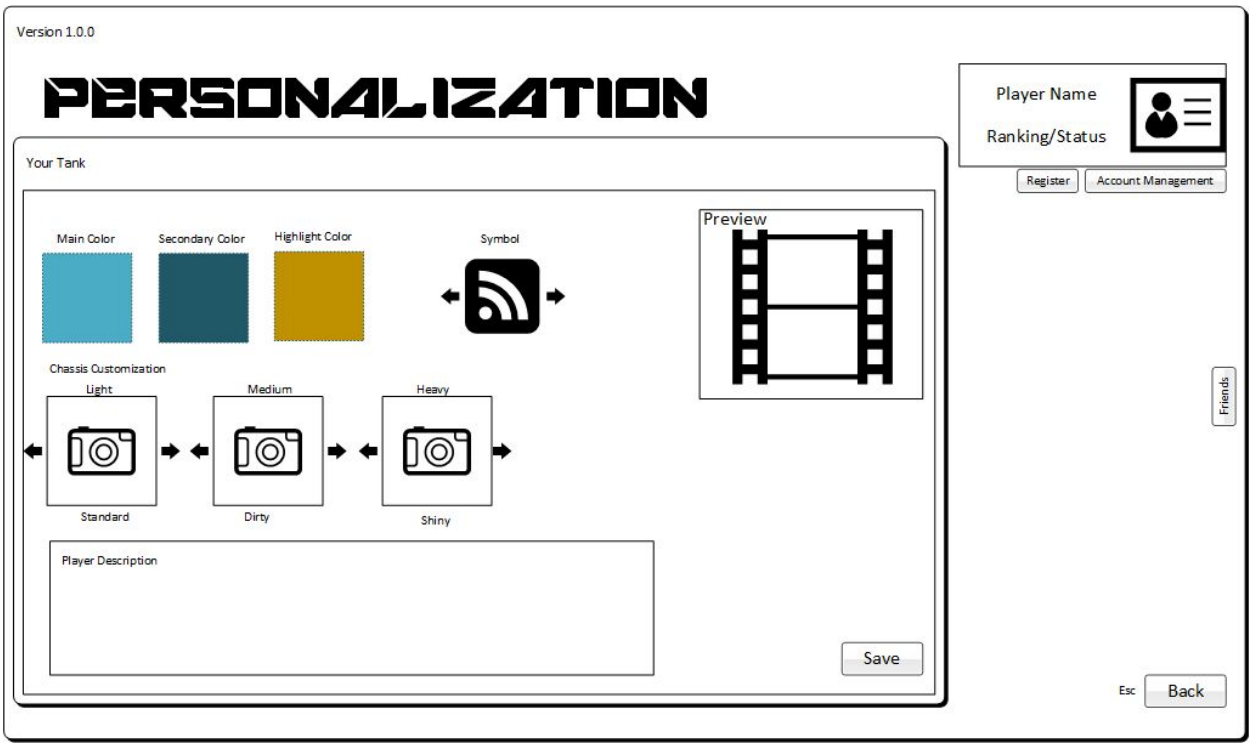
Load-out is where all players in a game select the chassis type and equipment they will use once the game starts. Once a load-out is selected the user must check the ready box to flag themselves as ready to start the game. The ready status and loadout summary for all players is shown on the right hand side of the screen, with details shown in a hover text area. A game will auto start after some fixed time limit even if their are unready players. A short countdown will be shown when all players are ready.

Training



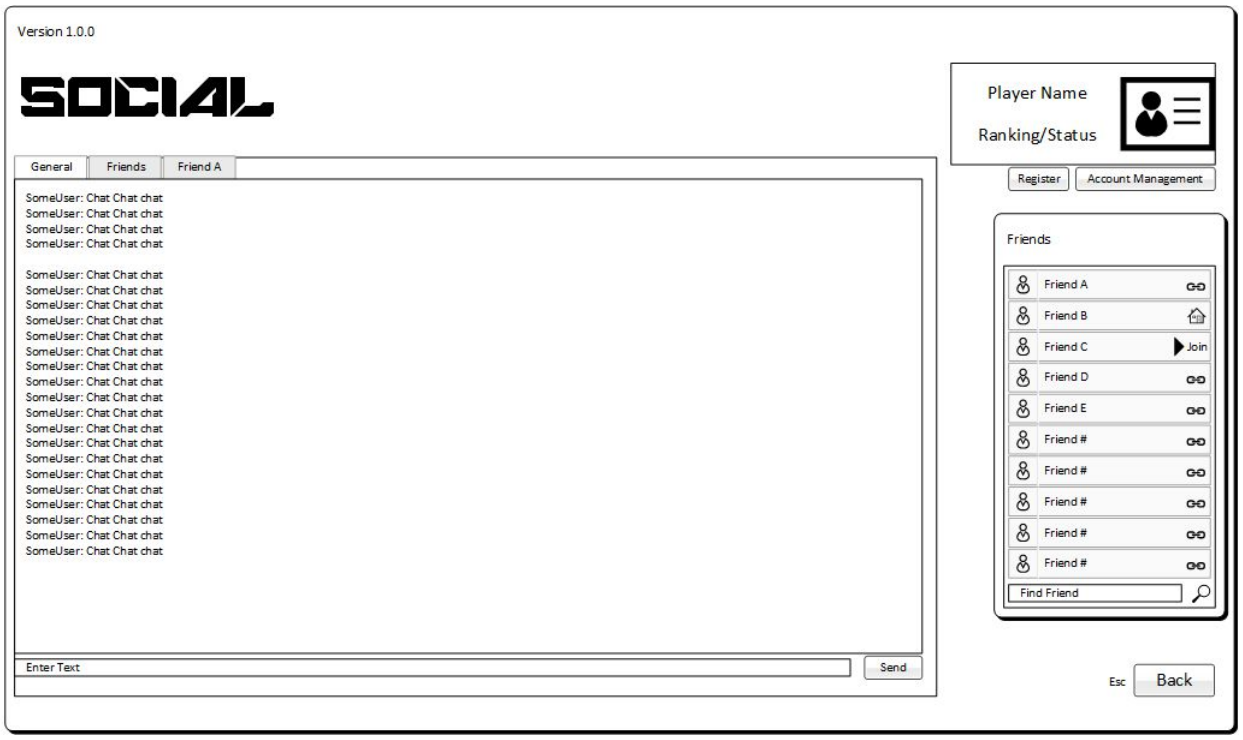
The training menu is a subset of the start game menu that will only start single player missions. New unregistered players will be taken to a specific tutorial when they pick quick play for the first time.

Personalization



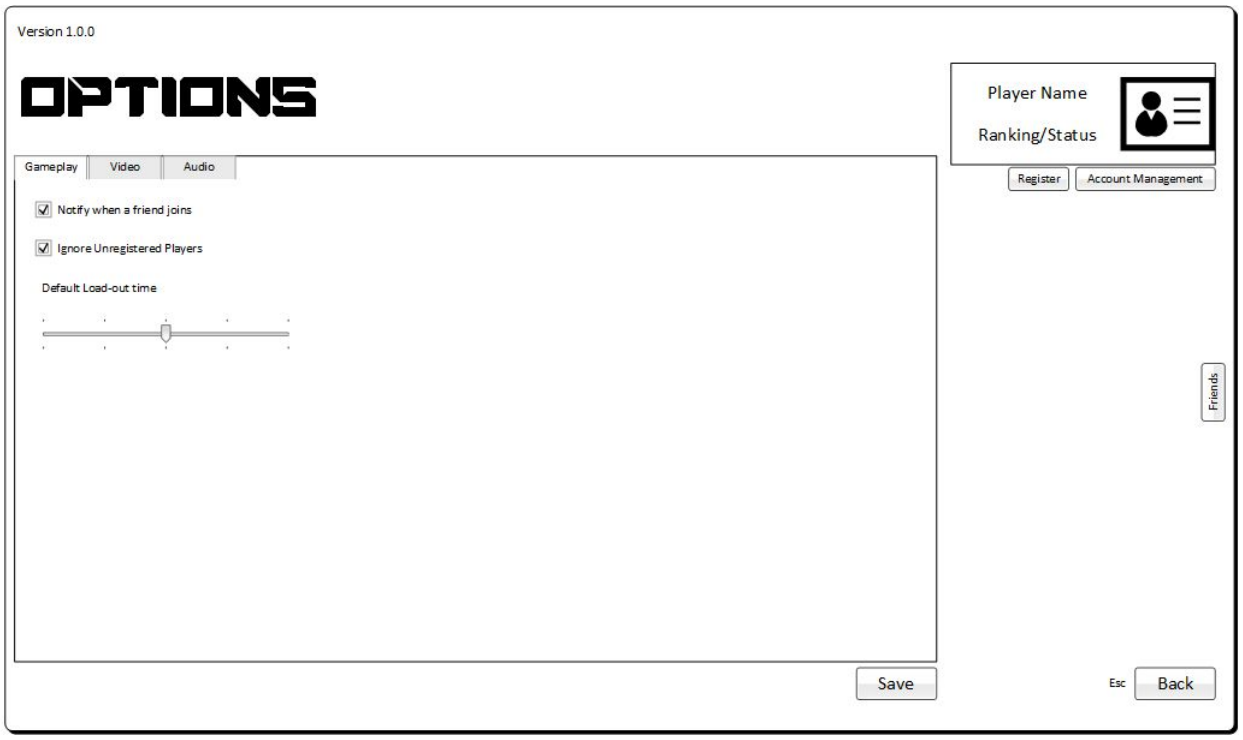
The personalization menu is where registered users can set visual customization options. Players can pick 3 colors that will be applied to all tanks they select in all configurations. Chassis and tank components will use the main and secondary colors for the bulk of all geometry. The highlight color will be used for specific small nodes or symbols on geometry. Multiple variants of geometry are available for each chassis type and the player can select which one they want to use. Chassis variants are purely cosmetic and do not affect gameplay in any way. The player can also select a symbol that represents themselves in the game HUD. A live preview of color and chassis customizations will be shown in the upper right oof the screen.

Social



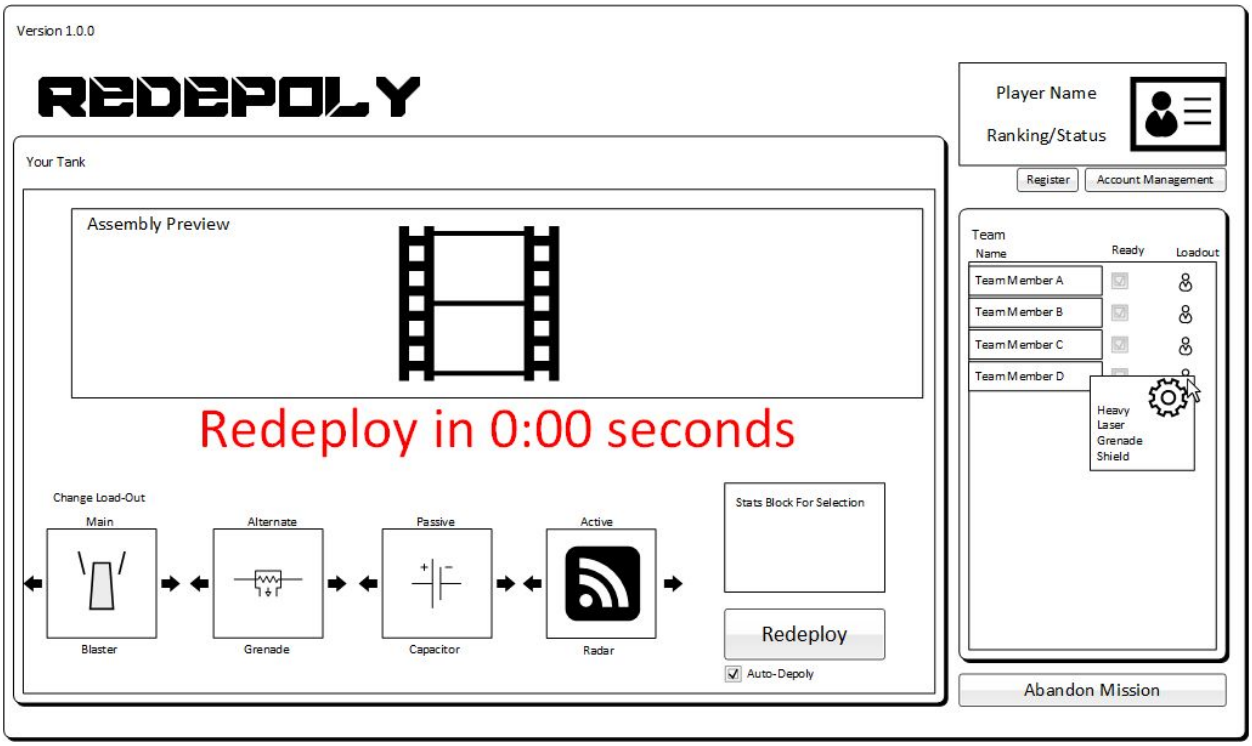
The social menu is a basic tab based chat screen. Their is a general lobby chat tab where registered users can chat. If a group of friends has been registered, it will receive a tab for chat as well. Finally chat with specific friends will be each given their own tabs. A client will remember what chat tab's it had open and will restore them on each launch.

Options



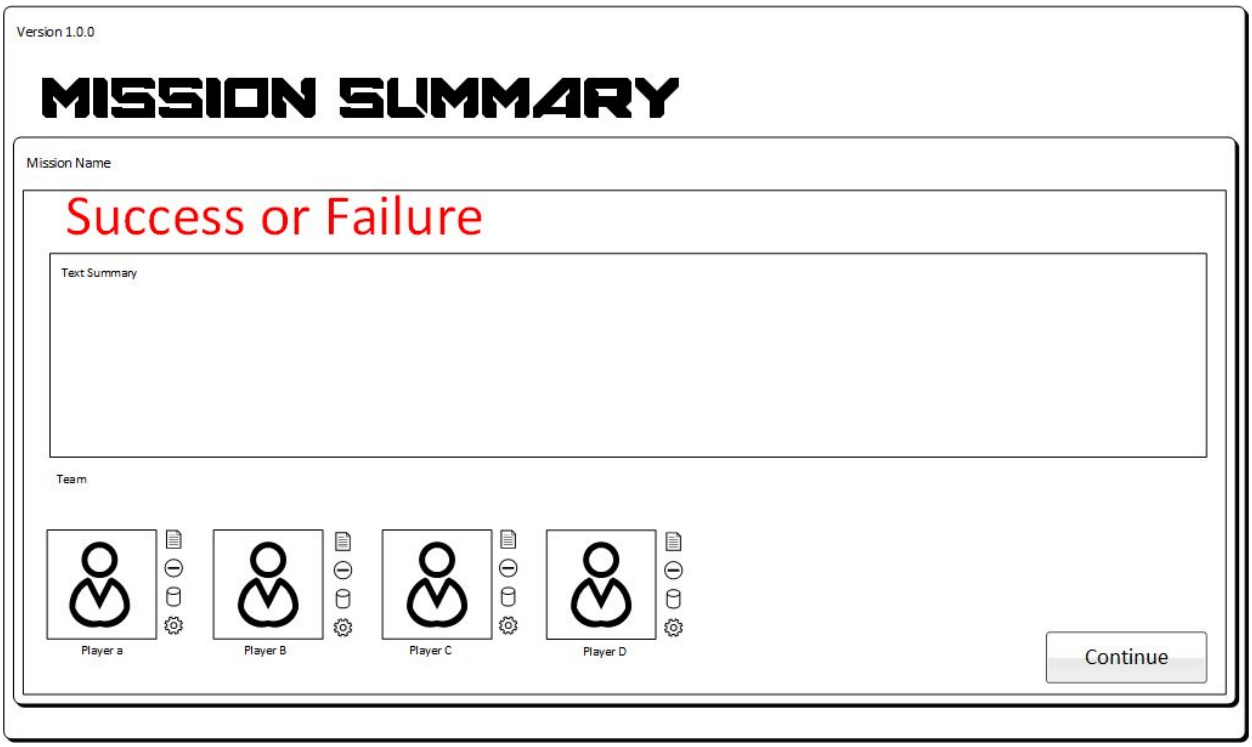
The options menu will show options (TBD).

Re-deploy



The re-deploy or respawn screen is shown when a player has died during a mission. The player will need to wait for the respawn timer to countdown before they can redeploy into the game. The user will click the redeploy button to re-enter the game. An option to auto deploy when the timer reaches 0 is available and will be keep it's setting across respawns when changed (local client setting). Each chassis class has its own respawn timer length, heavy tanks take longer to respawn. During the respawn wait the screen will display a rebuild animation showing the new tank being assembled with the current loadout. The user can change the loadout if they wish before re-deploying, but doing so will restart the countdown and the rebuild animation will restart as a new tank is constructed. The chassis class can not be changed while a mission is in progress.

MIssion Summary



The mission summary screen is shown when a game mission finishes, either in success, failure, or abandonment. A summary of the mission objectives that were completed will be shown as well as any failures. The lower portion of the screen contains a list of all team members who participated in the mission and any relevant stats they accumulated, such as hit ratio, total damage, total healing, etc… The player will press continue when they are ready to go back to the find game or main menu (if they ran a quick play mission).

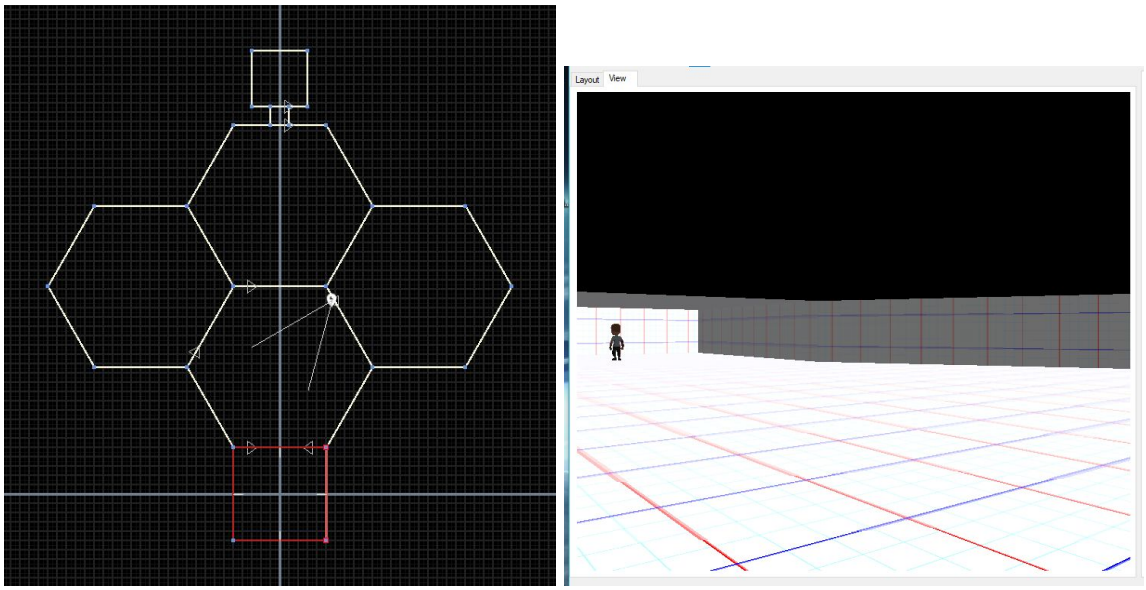
Gameplay

World

There are two main possible systems for describing the world. Each implies a different world graphical style but both meet the requirements of a world system that is easy to edit, highly compressible and allows both inside and outside areas. World definitions are only for geometry and texture descriptions, they do not contain actual image data.

Portal-Cell Worlds

The world map will be comprised of closed irregular convex polygon “cells”. Cell’s define an open space that can contain players, enemies, objects, or other game items. Cells can be linked at edges to form larger spaces. Edges can be marked as closed or open. All of a cell’s closed edges are extruded up to form walls. Each cell’s floor and ceiling can be at any height or in any plane, allowing for sloped floors or ceilings. Open Edges, Floors, or Ceilings will show the world skybox through the opening.
This system allows for simple 2.5D collisions and navigation and well as providing a simplified tree structure for world building.



Simple cell map, and 3D view.

Voxel Worlds

THis is another possibility, it uses more data but is easier to build for most people since it’s lego-like.

Physics

Objects in the world will use simplified physics, keeping to the 2.5 nature of the gameplay. All collisions will be done between cell walls using a simple bounding sphere and the colliding edge. The height collisions will be done with a simple height range, keeping the object above the floor and below the ceiling. This is effectively a bounding cylinder check. Objects for the purposes of simulation will never tilt off of the plane of the cell they are in. Graphical objects may tilt to be normal to the plane they are on, such as tanks rotating to “tilt up” and align with the ramped floor of a cell, but it will still be doing a 2.5D cylinder collision check.

Objects that are not resting on the ground and are not flagged as unaffected, will fall using a constant gravity acceleration. Tanks and weapon shells will fall down when in the air. Contact with a wall during a fall will cause the object to slide down the wall to the floor of the cell the wall is attached to.

Tank motion will have small amounts of acceleration and tank collisions will generally be non-elastic.

Shot Physics

Shots interact with tanks by using the bounding cylinder for the tank just like the world.
Shots fall into one of the following categories as far as how they interact with the world.

- Bolt
- Beam
- Shell
- Field
- Hitscan

Bolt

Bolts have no mass and are unaffected by gravity. They fly in a straight line at some fixed speed and detonate on impact with anything. They use one ammo/charge per shot.

Beam

Beams have infinite speed and are effect over time. They are not affected by gravity. They use ammo/charge rate per second.

Shell

Shells have mass and are affected by gravity. They fly in an arc that is determined by their launch angle and flight speed. They use one ammo/charge per shot. Shells show a hud element that shows the curve and the drop based on distance (project firing arc into world)

Field

Fields affect all items within a spherical radius from the point of origin. They are attached to the shooter and move with it. They are effect over time and use ammo/charge rate per second.

Hitscan

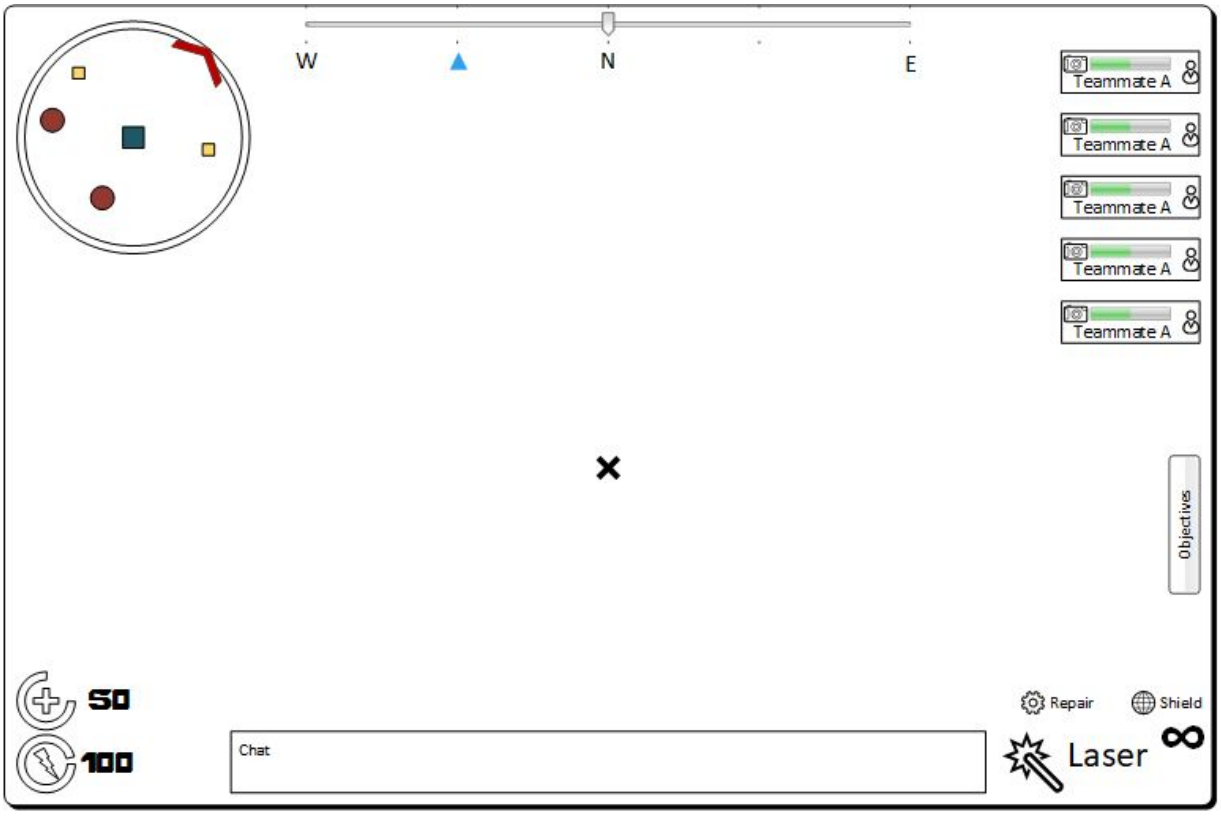
Hitscan shots are are infinitely fast moving shots that are not affected by gravity, but are not effect over time. They are point based and use ammo/charge per shot

General Gameplay Loop

The overall gameplay loop will proceed as follows.

1. Player Spawns into the world at a location determined appropriate for mission progress.
2. Player and teammates proceed toward mission objective, fighting enemy tanks as they are encountered.
3. As enemy tanks are destroyed they drop health and special ammunition refill pickups that players can use to restore expended resources. See loot mode section for more info.
4. If a player takes damage to the point where armor is 0 or less, they will be destroyed.
    a. A short death animation is shown
    b. The player is taken to the redeploy screen.
    c. When ready the player may respawn into the world.
        i. Player respawns at a location in the nearest spawn group to the largest number of team members but is furthest away from an enemy (Weighted search)
5. As the team completes mission objectives they are marked as completed on the mission list.
6. If the teammate completes the objectives required for success the mission ends in success.
7. If the mission failure conditions are met before success, the mission ends in failure.
8. The players are sent to the mission summary screen.
9. Players may select a new mission and start again.

HUD



The in game hud has information required by the player during the game. It is projected on top of the 3DView. The hud has several main sections

Mini Map

The upper left corner of the screen contains a mini map of the area around the player. Detectable objects and pickups in the area will be shown relative to the player who is always centered on the map. The map view is rotated so that forward for the tank is always up on the map. Mission objective markers will be shown around the edge of the mini map when active, to help the player know the direction they need to go.

Compass

The upper center of the screen shows a compass that scrolls as the player turns. The compass shows markers for mission objectives as well as nearby teammates and enemies.

Team List

The team list is in the upper right corner of the screen and shows the health status of all teammates as well as a summary of their loadout. When a teammate sends a communication packet the item in the team list will highlight or jiggle in some way.

Objective Flyout

Below the team list on the right side of the screen is the mission objective flyout. When activated this will bring up a mission objective summary and with the status of each item.

Loadout Display

The lower right corner of the screen is filled with the player's current loadout status.
Each weapon and active accessory is shown along with the number of special ammunition loaded in each item (if required).

Communications Area

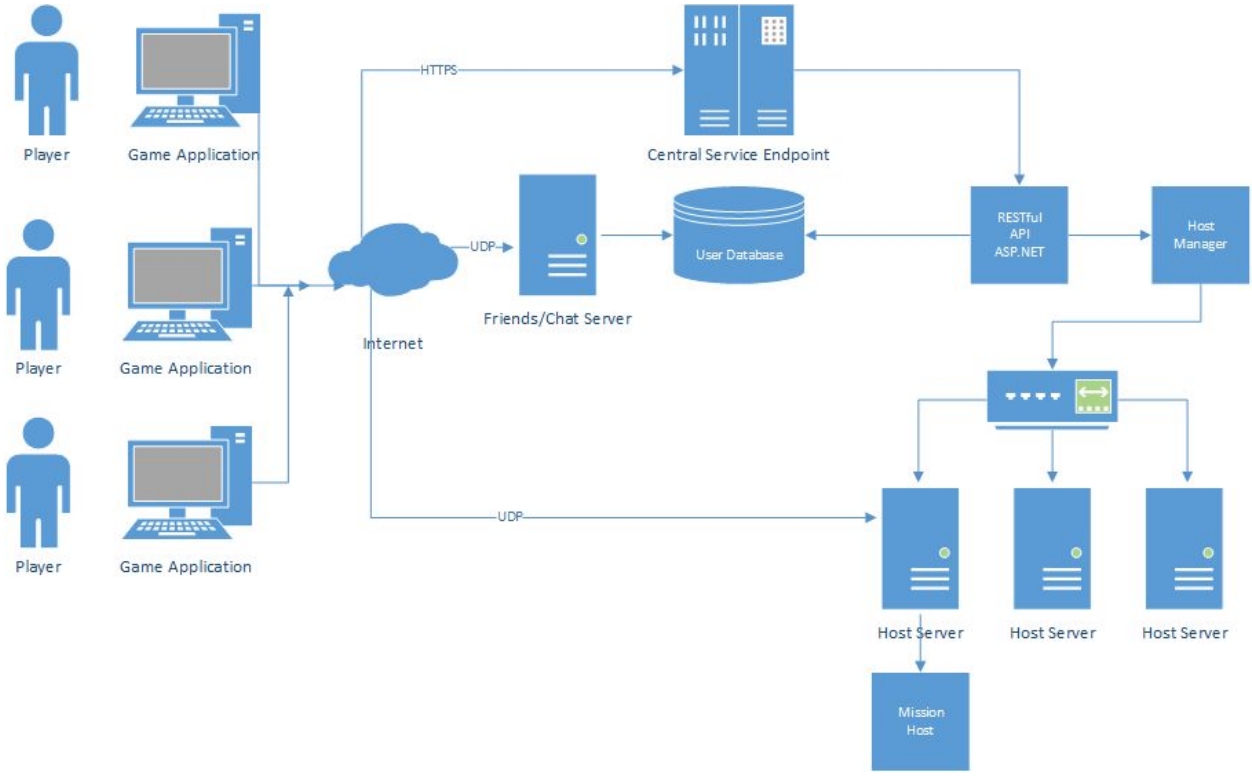
The lower centers of the screen contains the communications area. This is a chat-like text box that will show the most recent communications from teammates. If available a text entry box will be shown below the chat box for registered users to enter text after going into chat mode. Otherwise communications messages will be triggered via a radial menu after going into communications mode.

Status Display

The lower left corner of the screen is filled with the player's current status.
Capacitor and Armor values are displayed along with a circular fill ring to show the current percentage of maximum for each resource.

System Design

Overall System Structure



The system will be designed with a client/server model with three major systems that player clients will interact with.

Central Service Endpoint

This is the main server that all clients first connect to. It handles user authentication and registration as well as active game management. It is presented as a restful API exposed over HTTPS with JSON messages. It will perform the following services;

- Registration
- Authentication
- Account Management
- Game Listing
- Host Management (allocating new games)
- Address Management
- Software Updates

Social Server (Friends and Chat)

The social server will handle all real time communication and notifications dealing with inter-player communications and connections. Clients will connect to it via UDP after authenticating with central services and getting an access token. The server will implement the following services;

- Friends List
- Friend Status Notification
- Global, Group, and Game Chat
- Inter player private messages.

This service can be scaled into a cluster as needed.

Host Management

The Host manager and host server components will handle the spawning of mission server instances. This level of service can be run at a global server level, or at a local mission leader level if desired.

Global Services API

The global services API will be based on HTTPs and use JSON messages via a restful API. The POST verb will be used whenever possible. Below are the resources provided by the API.

GenerateAnonUser

Generates an anonymous user and provides a temporary game ID. This game ID will be stored and kept available for reuse by the user as long as the same access key is used. Temporary IDs that are not used for 4 weeks will be recycled.

Temporary IDs are flagged with special restricted permissions.

1. Must complete a tutorial before joining a public game
2. Are limited to public games that allow anon users
3. Can join a game they have a game code for (even without doing the tutorial).

Request

- Client MAC Address

Response

- Temporary User ID
- Temporary User Name (selected by the system)

- Access Key
- Authentication Key
- Current Cosmetics (color and user selections)

The API will use the request IP and hostmask, and provided MAC address to help minimize temporary ID abuse for honest people. Dedicated malcontents will of course be able to easily subvert this, so the system will do what it can to track uses and auto-ban IP ranges if they have too many requests. Temporary IDs are also IP locked, so they can only be joined from the same IP they authenticated on.

RegisterUser

Registers a user account and links it to an email. Requested may be for a completely new user, or to convert a Temporary ID to a registered user (with a name change if desired)

Request

- Client MAC Address
- Desired Name
- Email Address
- Desired Password
- Temporary ID
- Temporary Access Key

Response

- Result (OK, or Error code)
- User ID
- User Name
- Authentication Key
- Current Cosmetics (color and user selections)

The API will return an error if the desired name is in use, if the email is invalid, or the password does not meet basic requirements.
TODO, look into OATH and that kind of stuff.

The new user account will be flagged as “unverified registered user” and may gain access to more services. A verification email will be sent out with a verification access code. This code must be entered into the game to finalize verification.

AuthenticateUser

Requests an authentication key for a use from credentials

Request

- Client MAC Address
- Email address or temporary user ID
- Password, Access Key or temp password

Response

- Result (OK, or Error code)
- User ID
- User Name
- Authentication Key
- Current Cosmetics (color and user selections)
- Must Change Password (bool)

The API will return an error if the credentials are invalid or the account is disabled.

VerifyUser

Requests that a verification code be processed for the user

Request

- AuthenticationKey
- UserID
- Verification Code

Response

- Result (OK, or Error code)

The API will apply a verification code to a user and if valid, upgrade the user to “Verified registered user” and open up full services to them.

RecoverUser

Requests a temporary password reset to be emailed to the user. Temp code is stored in a resets database and is only valid for some time frame

Request

- Email Address

Response

- Result (OK, or Error code)

The API will apply a verification code to a user and if valid, upgrade the user to “Verified registered user” and open up full services to them.

UpdateUser

Updates user data that is not empty.

Request

- AuthenticationKey
- New User Name
- New User Email
- New User Password
- New User Avatar
- New User Cosmetic Selections (tank bodies and colors)
- Remove User Data (deactivates account, removes email address)

Response

- Result (OK, or Error code)
- User ID
- User Name

GetAvailableCosmetics

Requests an authentication key for a use from credentials

Request

- AuthenticationKey

Response

- Result (OK, or Error code)
- Cosmetics list
    - Component Size
    - Component ID
    - Component Display Name

GetServiceHosts

Gets an updated list of service hosts

Request

- Authentication Key

Response

- Service Host List
    - Central Service Endpoint hosts
- Chat Host List
    - Chat/Friends server host list

The API will return the current list of service endpoints for all user facing systems. This allows the client to be notified of additional service hosts without a direct software update.

CheckNewsUpdate

Checks for an updated news blob

Request

- Authentication Key
- Current App Version
- Current App Platform
- Current Locale (language the user has)
- Last News Version

Response

- Result (Data, or None)
- News Version
- News Data

The API will check to see if there is an updated news data block.

CheckAppUpdate

Checks for an updated application.

Request

- Authentication Key
- Current App Version
- Current App Platform
- Current Locale (language the user has)

Response

- Result (Update, or None)
- Download URL
- Update Version Number
- Update Version Name (display name for update)
- Update Version Descriptor (localized display data for update)

The API will check to see if there is an updated version of the application. If there is then the response contains a download link for the requested platform, a machine readable version number, a localized human readable update name, and an optional localized update description (in HTML) about the update.

CheckContentUpdate

Requests an authentication key for a use from credentials

Request

- Authentication Key
- Current App Version
- Current App Platform
- Current Locale (language the user has)
- Current Content Version

Response

- Result (Update, or None)
- Download URL
- Update Content Version Number

The API will check to see if there is an updated version of the game content for a specific version. If their is it will return a download URL for a content update package

GetGameList

Requests a list of games that the user can join

Request

- Authentication Key

Response

- Game Descritor List
    - Game Code
    - Game Name
    - Mission Type
    - Mission Name
    - Owner User Name
    - Public
    - Friends Only
    - Allows Anon
    - Start Time (UTC)
    - Player Count

The API will generate a game list that the user can access. Data will be used by the client to show the available games.

RequestJoinGame

Requests a list of games that the user can join

Request

- Authentication Key
- Game Code

Response

- Result (OK or Error)
- Game Host Name
- Game Host Port
- Connection Key

The API will process a request to join a game that is waiting. If successful the result returns the connection information needed to connect to the game host, as well as a connection token.

RequestCreateGame

Requests that a new game instance be started and listed

Request

- Authentication Key
- Mission Type
- Mission Name
- Game Name
- Public
- Friends Only
- Allow Anon
- Wait Time
- Minimum Players
- Self Hosted
- Self Host Info (for NAT puchthrough)

Response

- Result (OK or Error)
- Game Code
- Game Host Name
- Game Host Port
- Connection Key
- Management Key

The API will process a request that a game be created with the specified options.

StartGame

Used by game hosts to indicate that a game instance has started and is no longer joinable.

Request

- Management Key

Response

- Result (OK or Error)

The API will flag the game as in progress and remove it as joinable

EndGame

Used by game hosts to indicate that a game instance has ended.

Request

- Management Key
- List Players
    - Player ID
    - Result

Response

- Result (OK or Error)

The API will flag the game as in over and the players as free.

UpdateGame

Used by game hosts to update player counts when people abandon or rejoin games in progress

Request

- Management Key
- Added Players
    - Player ID
- Removed Players
    - Player ID
- Player List
    - Player ID

Response

- Result (OK or Error)

VerifyGamePlayer

Used by game hosts to verify that a connecting player is who they say they are and if they are supposed to connect to the host.

Request

- Management Key

- Player Connection Key
- Player IP
- Player Hostmask

Response

- Result (OK or Error)
- User ID
- User Name
- Player access information
- Player avatar information
- Current Cosmetics info (colors and selected cosmetic for each size)
- Mission stats, totals/ratio

Global Services Databases and Settings

The global services system will maintain a database for non-user related items. These will only be items that need to be searched for, such as ban lists. Configuration items such as URLs and file locations will be stored in configuration files on disk.

Ban List Record

- Index
- User ID
- Hostname submask
- IP Submask
- Ban Author UserID
- Ban Reason
- Ban Comment
- Ban Date
- Active

Social Services API

The social services API will be based on UDP (Game message library) and use packed serialized class messages. Every client will connect to one endpoint / port on some server on the Social Services cluster. The addresses of all machines in the cluster will be provided by the Central Services API after a call to GetServiceHosts. The social services API can only be accessed with an authentication key that is provided by one of the authentication/registration methods. This key is stored in the userdabase and used to authenticate the client with the system. No usernames or passwords are sent to the social services server.

JoinRequest (client to server)

Sent on connection to request access to the services provided by the social system.

Payload

- Authentication Key

ErrorResponse (server to client)

Sent on any error

Payload

- RequestName
- Error Code
- Error String
- Fatal

JoinResponse (server to client)

Sent back on valid join

Payload

- UserID
- User Display Name

GetFriendsList(client to server)

Requests the list of current friends

Payload

- Empty

UpdateFriendsList (server to client)

Provides a full list of friends, overriding existing lists on the client

Payload

- Friends List
    - Friend ID
    - Friend Name
    - Friend Status (Pending Response, Pending Request, Online, Offline, In-Game, Away)
    - Friend Avatar Info
    - Current Game Code (if available)

AddFriendRequest (client to server)

Client requests to add a new friend. If valid server will send UpdateFriendsList with new friend set to Pending Response status back to the requester and UpdateFriendsList with requeter as Pending Request status to the requested friend (if online)

Payload

- Friend Name

VerifyFriendRequest (client to server)

Client requests to verify a pending friend request. Success will Update friends list with new status for both parties.

Payload

- Friend ID

RemoveFriendRequest (client to server)

Client requests to verify a pending friend request.Success will Update friends list with new status for both parties.

Payload

- Friend ID

UpdateChatGroup(server to client)

Provides a full list of users in a specific chat group, overriding existing lists on the client with the same name

Payload

- Group ID
- Group Name
- UserList
    - User ID
    - User Name
    - User Avatar Info

JoinChatGroup (client to server)

Requests that a user join the specified chat group ID or name

Payload

- Group ID
- Group Name

LeaveChatGroup (client to server)

Requests that a user join the specified chat group ID

Payload

- Group ID

CreateChatGroup (client to server)

Requests that a user join the specified chat group ID

Payload

- Group Name
- Invited Users List
    - User IDs

ReceiveChat(server to client)

Server sends to all clients in a chat room when chat is added. Message data may be emote code.

Payload

- Group ID
- Sender ID
- Sender Name
- Sender Message Data

SendChat(client to server)

Client sends chat to a specific group.

Payload

- Group ID
- Sender Message Data

User Database

The user database is used by the central service and chat clusters. It contains all user and token data.

User Record

- Index
- User ID
- User Name
- Email
- Credential Hash
- User Type (Temporary, Registered)
- Verification State (Temporary, Unverified, Verified)
- Permission List (User, Admin, etc..)
- Active
- Creation Date
- Last Use Date

User Access Record

- Index
- User ID
- Date
- Action Type

User Reset Record

- Index
- User ID
- Temp Pass Hash
- Requested Date
- Used Date
- Active

Friends Record

- Index
- ***FriendA ID CompoundKey***
- ***FriendB ID CompoundKey***
- Status (Requested, Verified, Inactive)
- Active
- Creation Date

Authentication Tokens Record

- Index
- Token
- User ID
- Creation Date
- Last Used Date

Game Host Record

- Index
- Game ID
- Game Code
- Owner ID
- Creation Date
- Last UpdateDate
- Status (Pending, Active, Completed)
- Cluster Hosted
- Host Info
- Host Public Key
- Management Key
- Mission ID
- Minimum Players
- Timeout
- Public
- Allow Randos
- Allow Anon
- Chat ID

Game Connection Key Record

- Index
- Game ID
- Connection Key

- User ID
- Creation Date
- Active

Chat Group Record

- Index
- Group ID
- Group Name
- Group Creator
- Creation Date
- Group Type (Game, Personal)
- Active

Chat Group Membership Record

- Index
- User ID
- Group ID
- Active

Game Messages

The game messages will be based on UDP (Game message library) and use packed serialized class messages. Every client will connect a game host that is provided by the join game or create game response. These responses will give the client a connection key. This key is a token that is used for that game only.

JoinGameRequest(client to server)

Client sends on connect to get basic info.

Payload

- User ID
- Connection Key (from services, or a reconnect key)

ErrorResponse (server to client)

Sent on any error

Payload

- RequestName
- Error Code
- Error String
- Fatal (when true client will be disconnected)

JoinGameResponce(client to server)

Client sends on connect to get basic info.

Payload

- Player ID
- UserID
- User Name
- Re-Connection Key (Client should shave this info as part of the “last played game”, to allow reconnects in the case of crashes)

AddPlayer(Server to Client)

Server sends this to add a human player to the client’s game state

Payload

- Player ID
- User ID
- User Name
- Avatar Info

RemovePlayer(Server to Client)

Server sends this to remove a human player to the client’s game state

Payload

- Player ID

SetPlayerInfo(Server to Client)

Server sends this to update a player’s tank data

Payload

- Player ID
- Tank Data (class, loadout, cosmetics)

- Tank Health

GetMissionInfo(client to server)

Client sends to server to get info about the mission

Payload

- Mission Type
- Mission Name
- Mission Description
- Map Name
- Map Hash
- Mission Objective List
    - Objective ID
    - Objective Name
    - Objective State (shown, hidden)

GetWorldData(client to server)

Client sends to server to request map data during loadout phase.

Payload

- Chunk Index (-1 for all chunks)

SetWorldData(Server to Client)

Server sends this to client with compressed map data.

Payload

- Chunk Index
- Total Chunks
- Map Data

RequestLoadout (client to server)

Client sends to server to change loadouts

Payload

- Tank Size ID
- Main Weapon ID
- Alternate Weapon ID

- Passive Accessory ID
- Passive 2 Accessory ID
- Active Accessory ID

SetLoadout (server to client)

Server sends to client when loadout of any player changes

Payload

- Player ID
- Tank Size ID
- Main Weapon ID
- Alternate Weapon ID
- Passive Accessory ID
- Passive 2 Accessory ID
- Active Accessory ID

LoadoutComplete(client to server)

Client sends to server when it is done with a loadout, or ready to respawn

Payload

- Empty

LoadoutCountdownUpdate(server to client)

Server sends to client when a player has spawned, including self

Payload

- CountdownTimeLeft
- CountdownTimeEnd (GMT)

SpawnPlayer(server to client)

Server sends to client when a player has spawned, including self

Payload

- Player ID
- Position Data
- Rotation Data
- Effects State data

DespawnPlayer(server to client)

Server sends to client when a player has despawned, including self

Payload

- Player ID
- Redeploy time

SpawnObject(server to client)

Server sends to client when a non player object is spawned

Payload

- Object ID
- Object Class
- Timestamp
- Position Data
- Rotation Data
- Graphics data
- Effects State data
- Interpolation Data
- Owner Data (so client can handle shot prediction of own shots)

UpdateObject(server to client)

Server sends to client when any object is updated, this includes players.

Payload

- Object ID
- Timestamp
- Position Data
- Rotation Data
- Interpolation Data
- Effects State Data

RemoveObject(server to client)

Server sends to the client when an object is removed or destroyed

Payload

- Object ID
- Timestamp
- Position Data
- Rotation Data
- Destroy
- Destroy Effects Data

UpdatePlayerInput(client to server)

Client sends to server when input changes (on fixed input clock)

Payload

- Timestamp
- Input Values

SetPlayerStatus(server to client)

Server sends to clients to update various player stats (health, charge, ammo). Ammo and charge values are left empty for non self players. Heath is sent to entire team

Payload

- Player ID
- Timestamp
- Health
- Charge
- Ammo

ObjectiveUpdate(server to client)

Server sends to clients when mission objectives are updated.

Payload

- Objective ID
- State (show, hide, success, fail)
- Display Message

EndGame(server to client)

Server sends to clients when mission is completed.

Payload

- Mission Status (Success, Failure, Abandon, Error)

SetMissionSummary(server to client)

Server sends to clients when mission is over

Payload

- Mission Status (Success, Failure, Abandon, Error)
- Summary Text
- Player Stats List
    - Player ID
    - Player Stats

PartGame(client to server)

Client sends to server when it disconnects from the mission host.

Payload

- Empty

Application Structure

Main client and server apps are done in C#, using the open source UrhoSharp 3d Engine. This is a wrapper around the mature Urho3d engine that uses an Entity/Component system. Networking will be done with LineNetLib over UDP, it fully supports IPv4 and IPv6 and has excellent class serialization functions.

Data Structures

Used by server,client, or both.

Tank Instance : Node, Client & Server

- Entity ID
- Name
- Group List
- Chassis Class
- Chassis Graphics

- Color Set
- Weapon List
- Accessory List
- Charge
- Armor
- Ammunition
- Reload Timers
- State
- Last Input
- Last Update Timestamp
- Last Update Data

Player Instance : TankInstance, Client & Server

- User ID
- Name
- Avatar Info
- Respawn Time

Player Network Input: Node Component, Server

Attached to server side player instances to apply network input to server simulation.

- Network User
- Last Input
- Last Update Timestamp
- Last Update Data

Local Player: Node Component, Client

Attached to client side player local instances to get user input and send to server

- Network Host
- Last Input
- Last Update Timestamp
- Last Update Data

Enemy Instance : TankInstance, Server

- Spawn Group
- AI Processor
- Enemy Class
- Spawn Buddies List

Enemy Classes : Server

- Name

- Colors
- Cosmetics
- Difficulty Options
    - Min Players
    - Max Players
    - Difficulty Ordering
    - Weapon List
    - Accessory List
    - Charge Modifier
    - Armor Modifier
- AI List

Weapon Class : Server

- Name
- Display Name
- Acceptable classes (map by size, with main or aux or both)
- Graphic List (by size class)
- Charge Per Shot
- Min Reload
- Ammo Per Shot
- Max Ammo
- Shot Class

Accessory Class : Server

- Name
- Display Name
- Passive or Active
- Acceptable Classes
- Charge Per Use
- Effect
    - Speed Modifier
    - Charge Capacity
    - Charge Rate Modifier
    - Armor Capacity
    - Bubble Shield
    - Ammo Capacity
    - Boost
    - Jump
    - Radar
    - Pickup Range Modifier
    - Jamming
    - Drone

- Effect Value

Shot Class : Server

- Name
- Graphic
- Shot Type
    - Bolt
    - Beam
    - Shell
    - Hitscan
    - BoltBurst
    - Repair Beam
    - Repair Field
    - Rocket
    - Designator
- Shot Guidance
    - Dumb
    - Auto Lock
    - Full Lock
- Proximity Distance
- Max Time
- Full Damage Radius
- Max Splash Radius
- Base Damage
- Damage Rand Add
- Flight Speed
- Spawn Effect
- Despawn Effect
- Hit Effect
- Flight Effect

Shot Instance : Node, Client and Server

- Shot Class
- Lifetime
- Lock Target
- Owner

Spawn Group : Server

- Group Name
- Group Tank Instances

AI Processor : Node Component, Server

Base class that can be attached to TankInstnaces, overrides input

- Type
- Team
- Targets
- Allies
- Spawn Method
- Update Method
- Target Destroyed Method
- Derived Types
    - Hunter
        - Goes after targets, will fight until dead, will focus on primary weapon unless secondary weapon will affect more people or do more damage. Will use active accessory whenever possible. Always runs out of charge.
    - Defender
        - Protects attackers, support, and hybrid. Will swap to general if alone
    - Support
        - Attempts to repair allies of any type, will hide if alone.
    - Hybrid / General
        - Will go after targets of opportunity but will also back off when damaged. Will favor longer range attacks over closer range ones.
    - Attacker
        - Will go after a singular target, use for defend actions
    - Escort
        - WIll follow a path, may stop if attacked
        - Destination
        - Path
        - Quiet time before continue (time in seconds between getting damaged and continuing path)
        - Self Defend (true/false)
        - Attack Opportunity (true/false)

Mission Objective Data

Mission Objectives are part of the mission and seperate from the map file. Mission objectives may reference map structures and locations but multiple missions may reuse the same map.

- Mission Type (Destroy, Capture, Defend, Retrieve, Escort)
- Mission Text

- Minimum Players
- Recommended Players
- Objective List
- Root Step

Objectives

Objective Items are the steps required to succeed at a mission. Their are visible objectives and invisible objectives as well as required and optional objectives. Objectives can be added at runtime by other actions. A mission is complete when all required objectives are complete or a mission action ends the mission. If all required objectives are successful the mission is successful, otherwise it is a failure. Objectives that are visible are shown to the team with a completion status.

Steps

Missions are made of steps. A step contains a list of actions that are processed while the step is active, and a list of condition groups. When a condition group is completed it's completion steps will be pushed to the active steps for the mission.

Condition Groups

A step has a list of condition groups. All active conditions are checked each game loop. A condition group contains a list of conditions. When all the conditions in a group are satisfied, then the completion steps for that group are pushed to the active stack.

Several Conditions are defined;

GetToCellCondition

Checks to see if players or an object are inside a specific cell

- Map Location data (cell number)
- Min Team Count (0 for entire team)
- Required Entity Types (used for escort)

GlobalKillCountCondition

Counts a number of global entity kills.

- Count
- Type Filter

GroupKilledConditons

Continues until a named spawn group is empty

- Group Name

TimerUnderValueCondtion

Continues until a named timer is under the specified value

- Timer Name
- Timer Value

WaitCondition

Uses a specific time value then continues

- Wait Time

Actions

Actions are things that happen when a step is started. Actions affect the game or mission state in some way. Actions may be instantaneous or take time. All actions have a list of sub actions that are processed when an action is completed in order to allow for action dependency.

Several actions are defined;

AddObjectiveAction

Adds an Objective to the objectives list

- Objective Name
- Display Name
- Description
- Visible
- Required
- Insert After

CompletMissionAction

Forces a mission to complete with success

- Message

DelayAction

Waits N seconds before processing Sub Actions

- Delay

DisplayMarkerAction

Shows a mission marker to the specified cell

- Cell
- Name
- Description

EnableSpawnGroupAction

Enables a new spawn group in the map

- Spawn Group Name

FailMissionAction

Forces mission failure

- Message

IncrementTimerAction

Adds a delta to an existing timer

- Timer Name
- Increment

PlaySoundAction

Plays a sound

- Sound Resource ID

RemoveMarkerAction

Removes a named maker

- Marker Name

RemoveTimerAction

Removes a named timer

- Timer Name

SetObjectiveStatusAction

Sets the status of a named objective

- Objective Name
- State

ShowMessageAction

Shows a mission message to all players

- Message

SpawnObjectsAction

Spawns one or more waves of enemies in a named group

- Group Name
- Wave list
    - Cell

    - Spread
    - Category
    - Count
    - Delay

StartTimerAction

Starts a named timer

- Timer Name
- Timer length
- String for display empty if hidden timer

Component Design

Server

Main Application

Has the program startup logic, accepts config file and starts Urho in Headless mode. Initializes all other components. Note: Server does not handle chat directly, only notification messages.

Game State

Manages the game play state, holds the spawn groups and references to active players and enemies. Runs the server side API. Processes network message commands.

Mission Manager

Manages the mission objectives and processes the mission state as the game goes on.

Network Manager

Handles inbound connections, message distribution and parsing. Handles resource management (sending map, etc..)

Services Manager

Handles server side events and interaction with central services.

Configuration

Passed in from whatever started the server.

- Game Code
- Management Code
- Mission Type
- Mission Name
- Game Name

- Public
- Friends Only
- Allow Anon
- Wait Time
- Minimum Players

Client

Main Application

Has the program startup logic, client side config, and starts the Urho application window. Initializes all other components.

Menu Manager

Handles a menu stack, and going into and out of menu mode. Service related menus interact with services and chat manager.

Update Manager

Works with services manager to download, and install application and content updates.

Services Manager

Handles all interactions with central services. Input comes from menu stacks.

Game State

Manages the game play state, holds references to active players and enemies. Processes network message commands and sends out messages from local player input. Directly handles in game display via Urho Nodes.

Network Interface

Maintains game server network connection. Handles sending messages from Game State and parsing inbound messages from server for use by game state (keep in a thread safe list)

Resource Manager

Thin wrapper around Urho Resource manager that takes server graphic / sound resource IDs and loads them in as real Urho Nodes from local resources.

Chat Manager

Handles all chat interactions with chat servers. Works on it’s own thread in the background.

Local Config

Saved client side with local settings

- Window Size and Position
- Graphic settings
- Input Mappings
- Saved Login Info
    - Last credentials
    - Temp ID
- Chat Logs

TODO

Cloud Host Management API

Future

Crowd control options?
Accessory balance