OOAD Project 5

Status Summary

Project Name: **BattleCOD 3: Eastern Front 2**
Team Members:

- Austin Albert
- Ian Meadows
- Andrew Hack

Github Link:
https://github.com/aual1780/OOP_BigProject

Work Done

The networking side of the project is basically completely finished. The controllers and a test server that will show what command is being received have been completed and work as designed. The Unity side of the project is about halfway done. A majority of the game's basic logic has been implemented, although it needs some balance tweaks. The checklist below paints a more thorough picture of the work done and how it was split up.

Work Done: Project Checklist

- ❏ Game Complete: Players drive the tank, shoot enemies, and score points
- ✓ <Austin> 1-6 players
- ✓ <Austin> Local or remote play (allow input from host if player count = 1)
- ✓ <Austin> Support multiple games on 1 network without cross-talk
- ✓ <Austin> User-configurable control scheme
- ✓ <Austin> Cross platform controllers (Win10, Ubuntu, Mac, Mobile)
- ✓ <Austin> Gamepad support
- ✓ <Austin> No crosstalk between tank operator roles (ie driver cannot shoot)
- ✓ <Austin> Transparent controller reconnection (network failure recovery)
- ✓ <Austin> Some operator roles are collaborative or time based. The UI should show some elements of the tank operational state to the relevant operators.
- ✓ **GameHost** (Shared Library, TankSim.GameHost)
    - ✓ <Austin> Host ArdNet server
        - ✓ <Austin> Wait for client connections
        - ✓ <Austin> Manage client roles
        - ✓ <Austin> Listen for operator commands
    - ✓ <Austin> Validate user input
    - ✓ <Austin> Create friendly API hooks
- ✓ **GameHost CLI** (TankSim.GameHost.CLI)
    - ✓ <Austin> Testbed gamehost server
    - ✓ <Austin> Concise, clean implementation of TankSim.GameHost to serve as basis for Unity implementation
    - ✓ <Austin> Testing target for controller connections, role assignments, commands, failure recovery, operation binding, etc
- ✓ **Client** (Shared Library, TankSim.Client)
    - ✓ <Austin> Host ArdNet client
        - ✓ <Austin> Connect to server
        - ✓ <Austin> Push operator commands
        - ✓ <Austin> Listen for operator commands
    - ✓ <Austin> Load operator modules based on roles assigned from server
    - ✓ <Austin> Handle user input
        - ✓ <Austin> Capture input
        - ✓ <Austin> Map input to operator commands
        - ✓ <Austin> Generate tank command and broadcast via ArdNet
- ✓ **Client CLI** (TankSim.Client.CLI)
    - ✓ <Austin> Provide cross platform CLI implementation of TankSim.Client
    - ✓ <Austin> Connect via GameID
    - ✓ <Austin> Accept user name input
    - ✓ <Austin> Process tank operator commands
- ✓ **Client GUI** (TankSim.Client.GUI)
    - ✓ <Austin> Provide interactive GUI implementation of TankSim.Client
    - ✓ <Austin> Connect via GameID

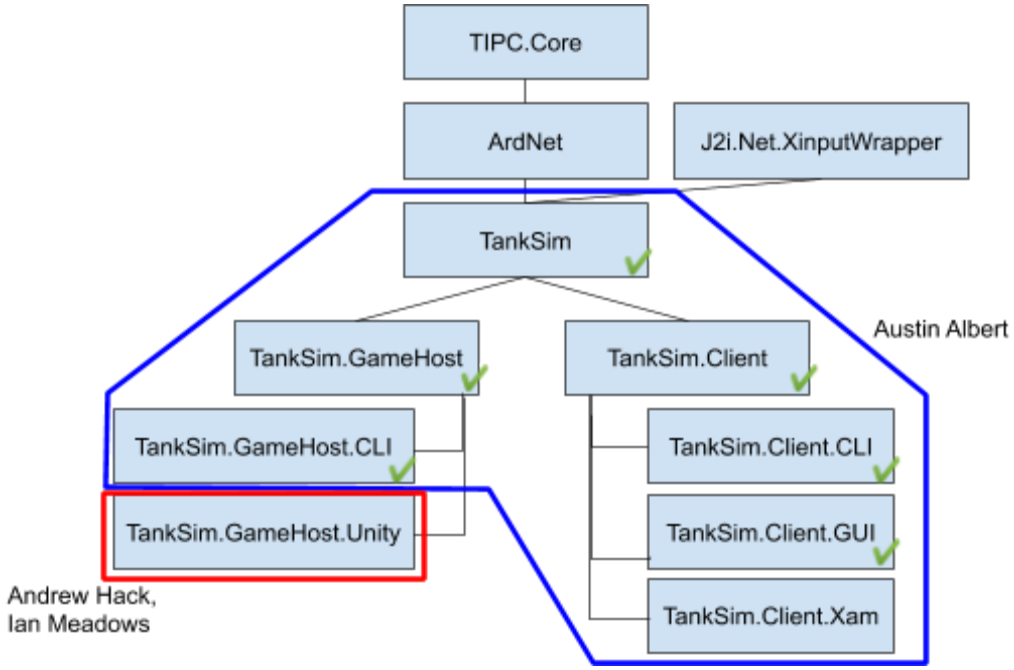
- ✓ <Austin> Accept user name input
- ✓ <Austin> Process tank operator commands
- ✓ <Austin> Process UI/HUD notifications from other collaborating controllers

❏ **GameHost** (Unity)

- ❏ Manage gameplay
  - ✓ <Ian> Create game lobby
    - ✓ <Ian> Get game name from user
    - ✓ <Ian> Get player count from user
  - ✓ <Ian> Wait for players to join
    - ✓ <Ian> Show players as “ready” after they set their name
    - ✓ <Ian> Show player names as they join the lobby
    - ✓ <Ian> Allow game to start after all players have joined
  - ✓ <Ian> Start Game
    - ❏ Autostart after all players are ready (3 second timer)
  - ✓ <Andrew/Ian> Manage game field
    - ✓ <Ian> Draw tank
    - ✓ <Ian>Draw field background
    - ✓ <Ian>Place field obstacles
    - ✓ <Andrew>Draw Enemies
    - ✓ <Ian> Draw HUD elements (score, health)
    - ✓ <Ian> Camera follows tank
    - ✓ <Ian> Draw Primary fire damage location
  - ✓ <Ian> Manage tank state
    - ✓ <Ian> Track tank health
    - ✓ <Ian> Track tank IsAlive status
  - ❏ Manage tank ammo
    - ❏ Determine available ammo types
    - ❏ Determine ammo stats (damage, radius, special effect, speed, max capacity, volley size)
    - ❏ Track current ammo capacity for each type
    - ❏ Get more ammo from world drops or point conversions
    - ✓ <Ian> Change fire retical based on ammo type
  - ❏ Manage tank operations
    - ✓ <Ian> Track movement
    - ✓ <Ian> Track gun aim
    - ✓ <Austin> Track gun loading (done on the server side)
    - ✓ <Ian> Track tank active ammo types
    - ✓ <Ian> Track gun shooting
      - ✓ <Ian> Primary
      - ✓ <Ian> Secondary
  - ❏ Projectile system
    - ❏ Show projectiles in flight
    - ❏ Show projectiles explode when they land
    - ✓ <Ian> Check nearby enemies to calculate damage

- ✓ <Ian> Fire secondary bullets
- ❏ Collision System
  - ❏ Tank collision damage
- ❏ Manage enemy generation
  - ❏ How many enemies on screen at once
  - ✓ <Andrew> Wave generation
  - ✓ <Andrew> Spawn rate
  - ✓ <Andrew> Spawn regions
  - ✓ <Andrew> Difficulty/health/speed/damage
  - ✓ <Andrew> Point tracking (how many points for hitting, killing)
  - ❏ Proper Texture for enemies
  - ✓ <Andrew> Enemies die when they are killed
  - ✓ <Andrew> Enemies leave splatter when they are killed
- ❏ Manage enemy AI gameplay
  - ✓ <Andrew> AI movement system
  - ✓ <Andrew> AI damage inflicting
- ❏ Manage scoring
  - ✓ <Andrew>Determine point system
  - ✓ <Andrew> Calculate points to add to score
  - ✓ <Ian> Track points during gameplay
  - ✓ Save points to disk after death
- ❏ Manage game stats
  - ❏ Driving distance
  - ❏ Total damage taken, damage dealt
  - ❏ Total ammo used, ammo type breakdown
  - ❏ Total enemies destroyed, enemy type breakdown

❏ Manage and display high scores
- ❏ Save/load scores from file
- ❏ Display to user in scoreboard scene
- ❏ Let user view full game stats


TIPC.Core
ArdNet
J2i.Net.XinputWrapper
TankSim
TankSim.GameHost
TankSim.Client
TankSim.GameHost.CLI
TankSim.GameHost.Unity
TankSim.Client.CLI
TankSim.Client.GUI
TankSim.Client.Xam
Austin Albert
Andrew Hack,
Ian Meadows

Changes or Issues Encountered

It turns out that Unity is not very pattern friendly, as it kind of has its own way of doing things, so most of the design patterns on the unity side function similarly without actually having the pattern itself implemented in our code.

Currently the clients can join the game, but they can't control the tank yet due to some kind of network issue.

Patterns

Command/Mediator: The ArdNet platform (along with the underlying TIPC library) expose a power command infrastructure that makes it easy for us to route commands from sources to listeners. This simplifies the process of developing cooperative modules while still keeping them isolated from each other.

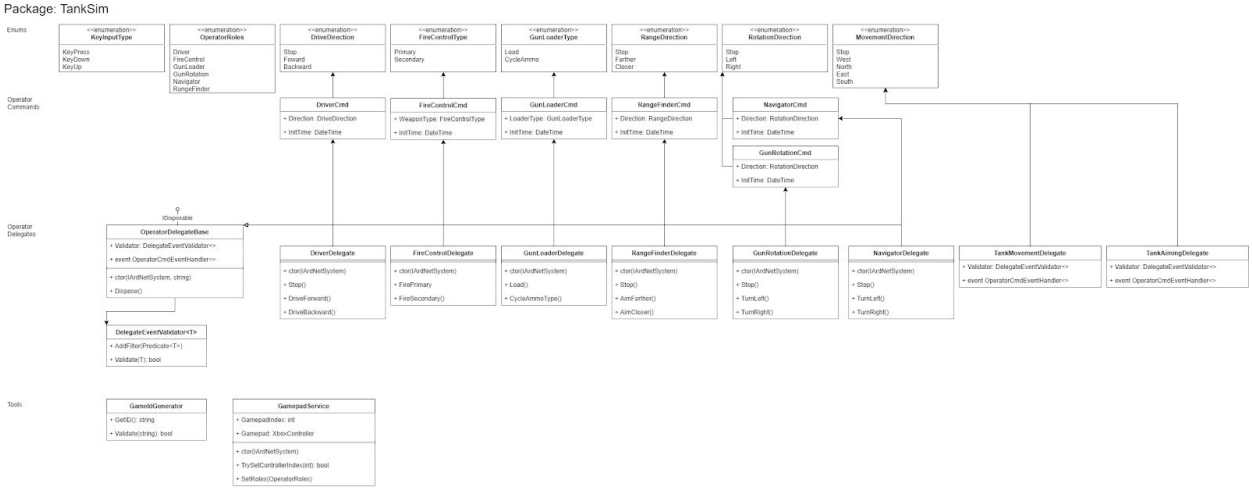
Dependency Injection: The tank client controllers are built with DI containers to simplify object lifetime management. Objects are injected into service constructors right when they are needed and are disposed when they are no longer used. This helps avoid lifetime bugs and memory leaks.

Argyle: My argyle socks are lovely and warm.

Facade: TankSim.GameHost exposes a facade over the ArdNet operator hooks. This facade makes it easy for both the testbed CLI and Unity to bind to operator command hooks without knowing any system internals.

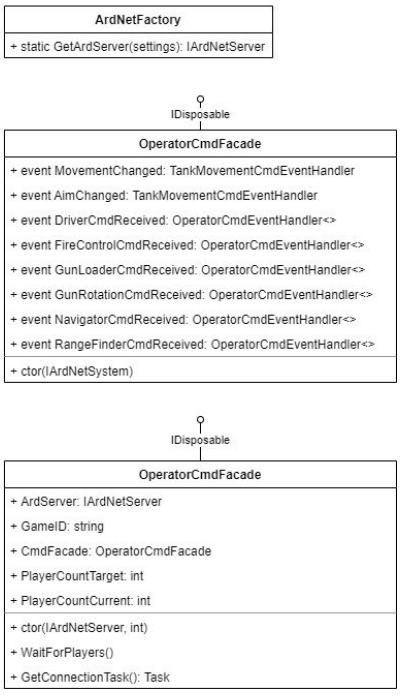
Class Diagram

Package: TankSim
Status: Complete
Contributors: Austin Albert



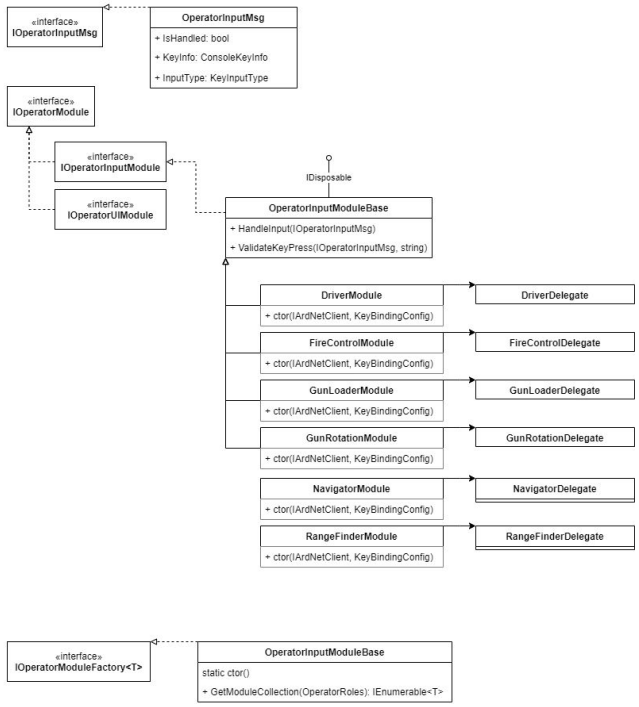
Package: TankSim.GameHost
Status: Complete (with modifications)
Contributors: Austin Albert

Package: TankSim.Gamehost



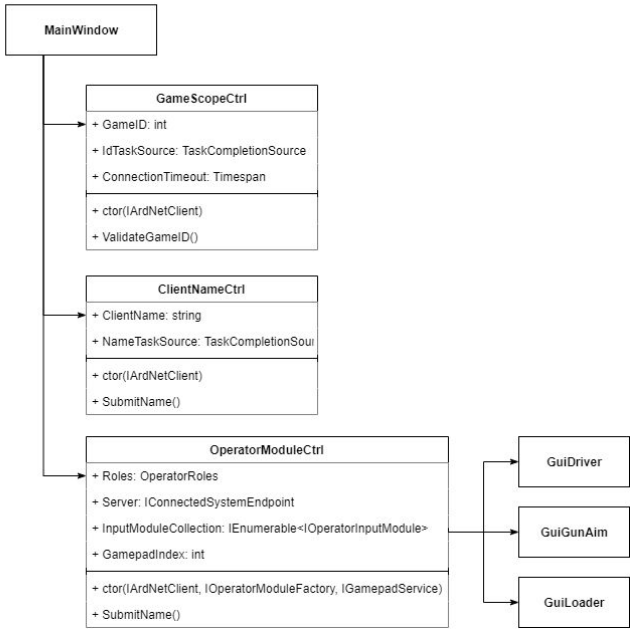
Package: TankSim.Client
Status: Complete (with modifications)
Contributors: Austin Albert

Package: TankSim.Client



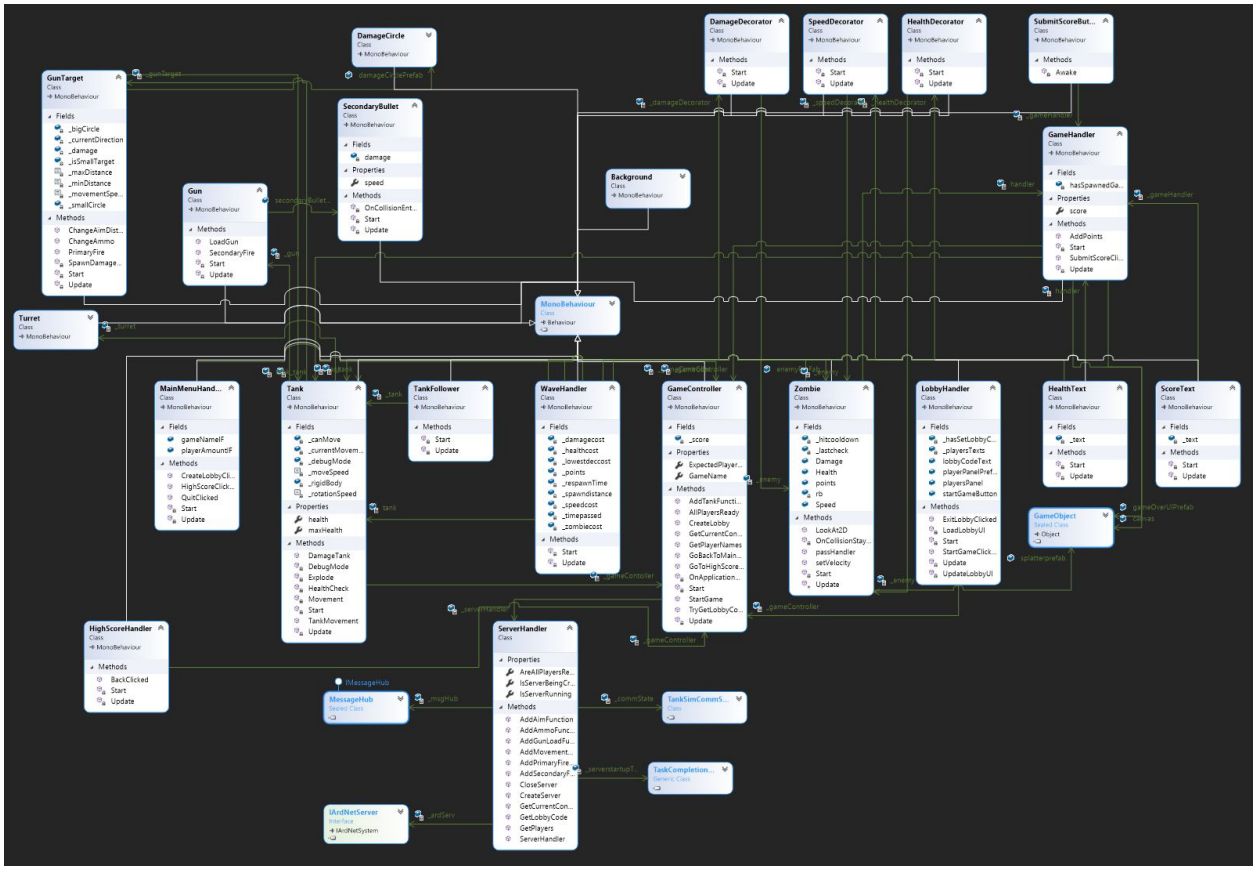
Package: TankSim.Client.GUI
Status: Complete
Contributos: Austin Albert

Package: TankSim.Client.GUI



Package: TankSim.GameHost.Unity
Status: In Progress
Contributors: Andrew Hack, Ian Meadows

Note: Unity does not translate well to a class diagram, because of its game object and component system. This is our best shot at representing our unity project as a class diagram.



Plan for Next Iteration

The networking side of the project is basically completely finished, and the game is about halfway done.

- Fix issue communicating with controllers
- Flesh out the game more
    - Finish implementing tank shooting mechanics
    - Add better images to the tank and enemies
    - Add a particle system to the tank
    - Give the primary gun a fire animation so the players have a better visual representation of the timing
    - Fix walls spawning on top of walls
    - Balance the game mechanics
- Implement the scoreboard system
    - Be able to save to and view a leaderboard
- Look into adding pickups
    - Ammo
    - Health packs
- Look into adding more enemy types
    - Possibly a tank enemy
- Look into adding a cool end game screen that shows neat statistics