TankWars

Analysis and Design Document

Version	1.5
Peter Leung	petle728@student.liu.se
Simon Hellbe	simhe966@student.liu.se
Ludwig Krokstedt	ludkr175@student.liu.se
Date:	2013-05-06

Summary

The purpose of this document is to get the main idea about the game context and how to test the program as we move further in the project. In this document we will elaborate the requirement specification and present somewhat more detail on how we will go about when creating the game and game contents.

The game creation will consist of three major phases; two of which can be started parallel. These phases will be the game creation, network creation and the implementation of them both. More details about the different test milestones can be found at the bottom of this document.

Table of Contents

Document conventions

None.

Class diagrams

The class diagram is given below.

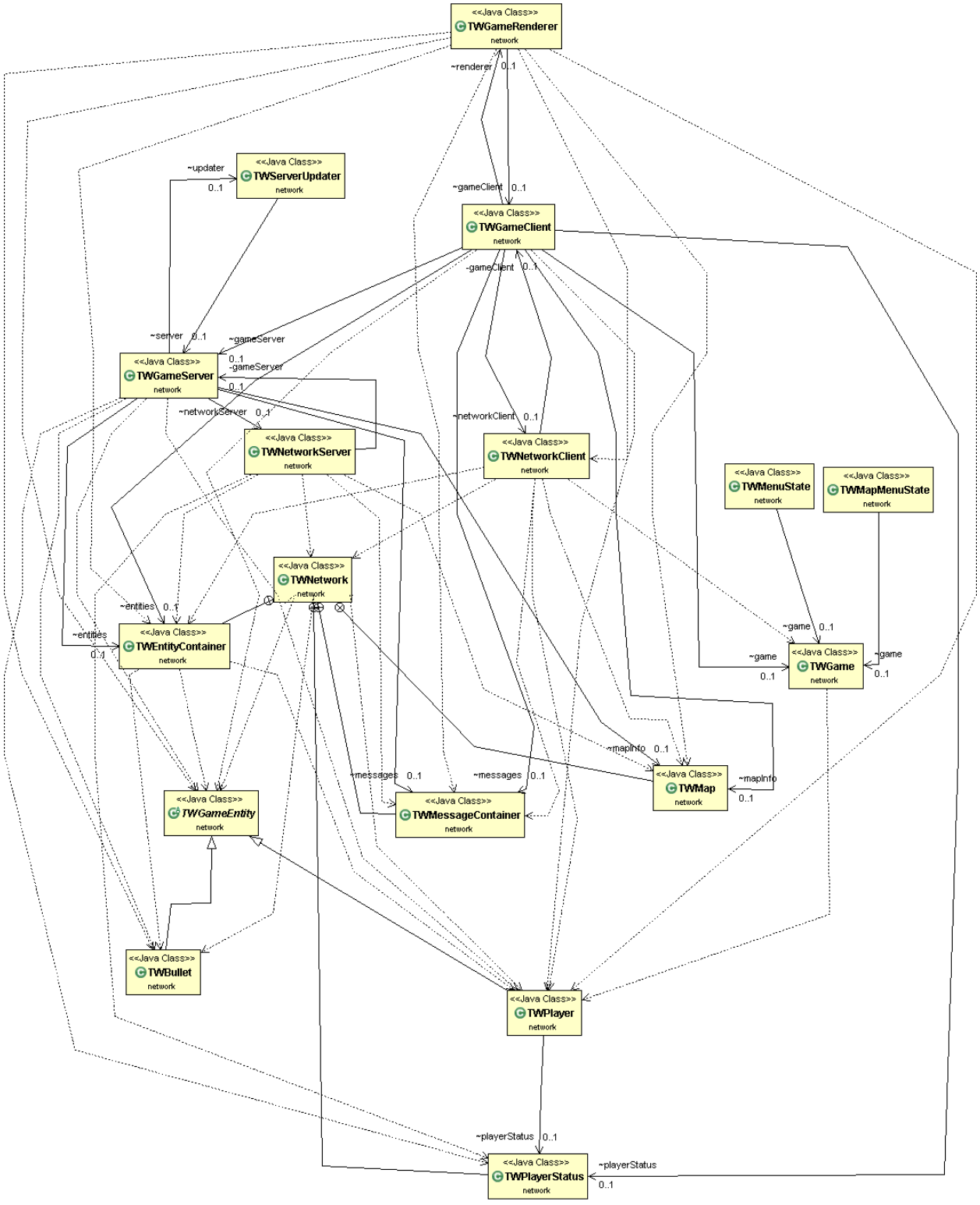

Figure 1. Class diagram

Class descriptions

TWGame
Superclass None
Subclasses None
Responsibilities The main program where all the different states initiate and enter TWMenuState. Creates the window for the program.
Collaborations Enters TWMenuState on startup.

TWMenuState
Superclass None
Subclasses None
Responsibilities The mainmenu where you can choose to Host, Join or Exit a game
Collaborations *Host*: Starts a TWGameServer, TWGameClient *Join*: Starts a TWGameClient

TWGameClient
Superclass None
Subclasses None
Responsibilities Coordinates the client side, sends key inputs to server and receives TWPlayerStatus. Prints the map etc for the user.
Collaborations Start a TWNetworkClient, if host then it also starts an TWNetworkServer

TWGameServer
Superclass None
Subclasses None
Responsibilities Holds the map, responsible for all the calculations and determines all the rules such as possible moves etc.
Collaborations Started by a TWGameClient host. TWGameClients non-host can connect to the server.

TWGameRenderer
Superclass None
Subclasses None
Responsibilities Renders all the graphics for the TWGameClient
Collaborations Sends the updated renders to TWGameClient

TWNetwork
Superclass None
Subclasses None
Responsibilities Contains and registers all the classes that are sent between TWNetworkClient and TWNetworkServer.
Collaborations The communication between TWNetworkClient and TWNetworkServer

TWNetworkClient
Superclass None
Subclasses None
Responsibilities Exchanges information between TWNetworkServer and TWGameClient
Collaborations Connect to a TWNetworkServer & TWGameClient

TWNetworkServer
Superclass None
Subclasses None
Responsibilities Exchanges information between TWNetworkClient and TWGameServer
Collaborations Connect to TWNetworkClient and TWGameServer

TWMap
Superclass None
Subclasses None
Responsibilities Contains information about the map to be loaded on the clients.
Collaborations TWNetwork have a TWMap

TWEntityContainer
Superclass None
Subclasses None
Responsibilities Contains all the entities in the game
Collaborations TWNetwork has a TWEntityContainer

TWPlayerStatus
Superclass None
Subclasses None
Responsibilities Holds information about a TWGameClient's key inputs and it's status, e.g if the player want to move, shoot etc.
Collaborations TWNetwork has TWPlayerStatus

TWGameEntity
Superclass None
Subclasses TWPlayer, TWBullet
Responsibilities A abstract class that is a superclass for all the entities in the game
Collaborations Subclassed by TWPlayer or TWBullet

TWPlayer
Superclass TWGameEntity
Subclasses None
Responsibilities Can move and shot, have health property.
Collaborations Implements the interface TWGameEntity

TWBullet
Superclass TWGameEntity
Subclasses None
Responsibilities A projectile with direction, speed and position
Collaborations Implements the interface TWGameEntity

TWServerUpdater
Superclass None
Subclasses None
Responsibilities Updates the server content on a given interval (here 40ms)
Collaborations Implements Runnable

TWMapMenustate
Superclass BasicGameState
Subclasses None
Responsibilities A menu where you can pick a map to host
Collaborations Initiates from TWMenuState and causes the client to enter Gamestate

TWMessageContainer
Superclass CopyOnWriteArrayList<E>
Subclasses None
Responsibilities Contains the chat messages in the network
Collaborations Defined in TWNetwork and is sent between NetworkServer and NetworkClient

Use case diagrams

- **GameClient**
    - Host a game
        - The GameClient starts a NetworkServer and a NetworkClient instance and connects to the NetworkServer. (they both extend network functionality from the Kryonet library)
    - Join a game
        - The GameClient starts a NetworkClient and connects to a NetworkServer class that is already running.
    - Prints world to user
        - NetworkServer sends information about the players to all the NetworkClients continously.
- **User**
    - The user starts the GameClient
    - Chooses to host a game, join a game or exit the game
    - Starts playing
    - Moves the tank with keyboard
    - Shoots a projectile
    - When the game is over the user chooses to go to either the main menu or exit the game.

- **Optional use case options**
    - User can choose between three tanks
    - Pick up power-ups
    - Shoot powerup projectile

Interaction diagrams

Menus are sequential and during gameplay the user watches the screen, sends input, the screen is updated and the process is repeated.

We chose to use sequence diagram because it is easier to overview.

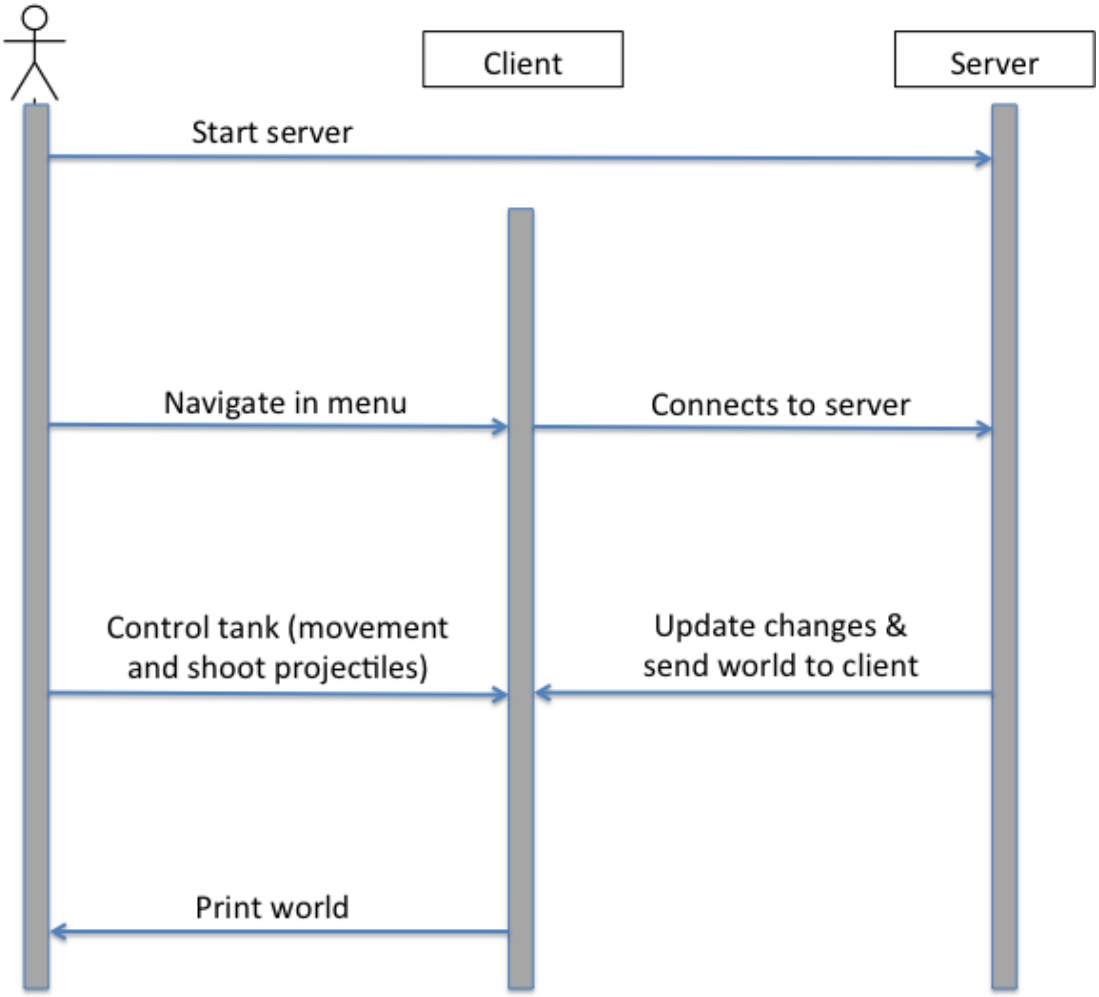

Figure 2. Interaction diagram

State and activity diagrams

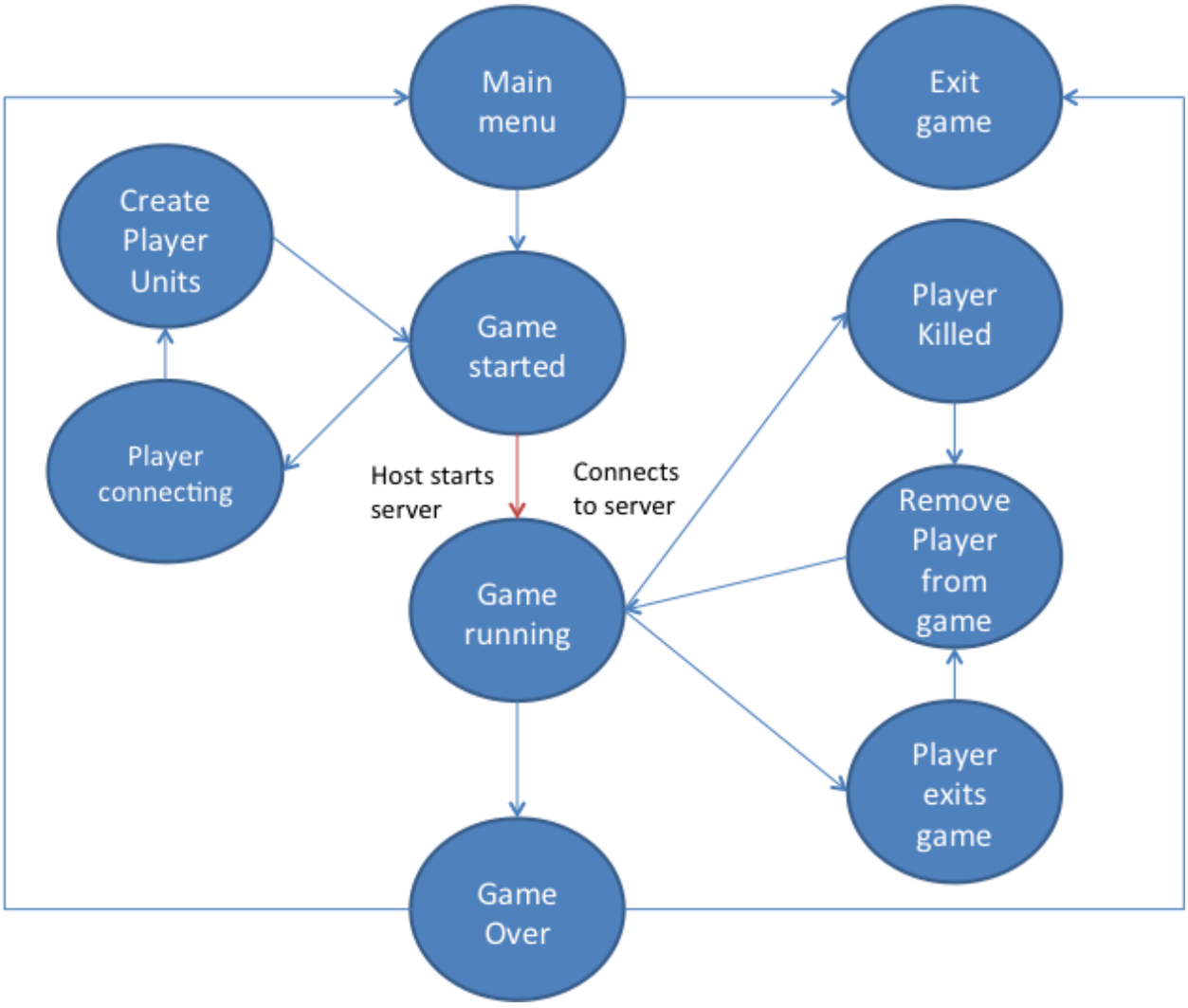

Figure 3. State diagram - a diagram of the different states in game

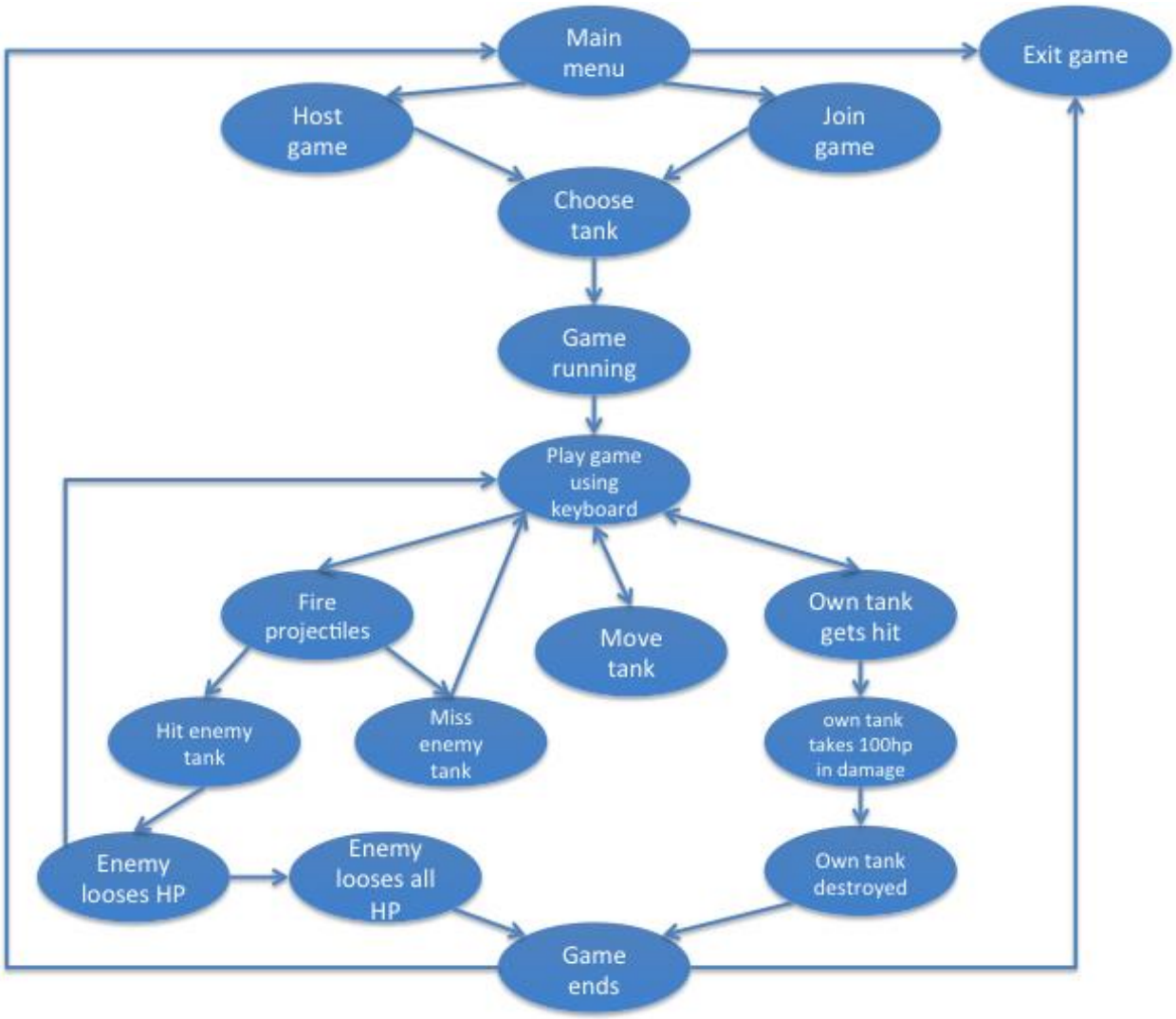

Figure 4. Activity diagram – a diagram of the activity in game

Test planning

Since our program has a clear division between game and network we will test these two parts separately until the two parts have been so implemented that they can be joint together.

Game test:

gTest 1: Game map and tank entity rendered, including walls
gTest 2: Tank moving and shooting correctly
gTest 3: Tank and bullets colliding with other entities properly
gTest 4: Health and powerup implemented
gTest 5: Game rules fully implemented
gTest 6: Game map rendered from generic TMX file

Server/Client test:

scTest 1: Client and Server classes created through Kryonet
scTest 2: Client successfully connecting to Server
scTest 3: Many Clients can successfully connect to one Server
scTest 4: Multiple clients communicating through Server
scTest 5: Send messages between Client and Server
scTest 6: Send messages between multiple Clients and one Server
scTest 7: Send keyinput from Client to Server
scTest 8: Send keyinput from multiple Clients to one Server
scTest 9: Clients receives package containing player statuses from Server

Implementation

iTest 1: Server running a game locally with clients connected to it
iTest 2: Server Host controls implemented
iTest 3: Clients controlling their tanks in game
iTest 4: Client can see enemy tanks and update positions as the other players controls it’s tank
iTest 5: Collision with game objects
iTest 6: Clients running on different computers connected to the same Server
iTest 7: Frames per second (fps) rate, not lower than 60 fps
iTest : Game fully implemented with network