

**POS-Tanks**

SEMESTRÁLNA PRÁCA

Vypracoval(a): **Patrik Vereš, Filip Mikula**

Študijná skupina: **5ZYI34**

Predmet: **Princípy operačných systémov**

Cvičiaci: *Ing. Patrik Rusnák, PhD.*

Obsah

[1. POS-Tanks 4](#_Toc123981436)

[2. Server 4](#_Toc123981437)

[2.1. Sockety 4](#_Toc123981438)

[2.2. Vlákna 4](#_Toc123981439)

[2.3. Synchronizačný problém 5](#_Toc123981440)

[2.4. Ďalšie problémy v rámci semestrálnej práce 5](#_Toc123981441)

[2.5. Štruktúra triedy Client 5](#_Toc123981442)

[2.5.1. Client(int id, std::string pName, float xPosition, float yPosition, DIRECTION direction, unsigned short port, sf::IpAddress ipAddress) 5](#_Toc123981443)

[2.5.2. ~Client() 5](#_Toc123981444)

[2.5.3. POSITION\* getPosition() 5](#_Toc123981445)

[2.5.4. CONNECTION\* getConnection() 5](#_Toc123981446)

[2.5.5. bool getFired() 5](#_Toc123981447)

[2.5.6. bool wasKilled() 5](#_Toc123981448)

[2.5.7. bool getLeft() 5](#_Toc123981449)

[2.5.8. bool wasScoreSent() 5](#_Toc123981450)

[2.5.9. int getClientId() 6](#_Toc123981451)

[2.5.10. int getKilledBy() 6](#_Toc123981452)

[2.5.11. int getScore() 6](#_Toc123981453)

[2.5.12. void updatePosition(float xPosition, float yPosition, DIRECTION direction) 6](#_Toc123981454)

[2.5.13. void setInitialPosition(float xPosition, float yPosition, DIRECTION direction) 6](#_Toc123981455)

[2.5.14. void setFired(bool fired) 6](#_Toc123981456)

[2.5.15. void resetPosition() 6](#_Toc123981457)

[2.5.16. void killed() 6](#_Toc123981458)

[2.5.17. void setLeft(bool left) 6](#_Toc123981459)

[2.5.18. void lockMutex() 6](#_Toc123981460)

[2.5.19. void unlockMutex() 6](#_Toc123981461)

[2.6. Štruktúra triedy Application 7](#_Toc123981462)

[2.6.1. Application() 7](#_Toc123981463)

[2.6.2. ~Application() 7](#_Toc123981464)

[2.6.3. void run() 7](#_Toc123981465)

[2.6.4. void initializeSocket() 7](#_Toc123981466)

[2.6.5. void waitForClients() 7](#_Toc123981467)

[2.6.6. void updateOfTanksPositions() 7](#_Toc123981468)

[2.6.7. void initializeGame() 7](#_Toc123981469)

[2.6.8. void sendData() 7](#_Toc123981470)

[2.6.9. void receiveData() 7](#_Toc123981471)

# POS-Tanks

Semestrálna práca POS-Tanks sa skladá z dvoch častí. Z klientskej časti (grafické rozhranie, ovládanie tanku, komunikácia so serverom,...) a zo serverovej časti (komunikácia s klientmi, riadenie hry,...).

# Server

Po tom ako sa spustí serverová časť si aplikácia vyžiada vstup na počet hráčov ktorý sa budú v danej hre vyskytovať. Následne sa čaká na pripojenie všetkých hráčov. Po tom ako sa pripojí hráč, hneď obratom dostane od servera jeho ID a plus jeho približné súradnice. Keď sa pripoja všetci hráči už len server rozpošle všetkým hráčom ich presné súradnice a dáta o ich súperoch.

Hra pokračuje až pokým sú v aréne pripojený hráči. Keď sa odpoja všetci hráči, server ukončuje svoju činnosť.

Po odpojení hráča mu server pošle aktuálnu štatistiku hry. To znamená že mu pošle jeho skóre a aktuálne skóre jeho protihráčov.

## Sockety

V serverovej časti aplikácie sú použité sockety na komunikáciu s klientom. Tato komunikácia prebieha oboma smermi. Komunikácia prebieha pomocou UDP protokolu.

UDP protokol sme zvolili z toho dôvodu že ma najmenšiu réžiu, čiže je najrýchlejší a pri tomto počte v akom server komunikuje s klientom tolerujeme určité straty.

Socket používame z knižnice SFML. Je to z dôvodu že ich sieťová knižnica je veľmi jednoduchá na správu a používanie. Zároveň z tejto knižnice používame aj Packet do ktorého vložíme dáta ktoré chceme poslať a následne keď druhá strana príjme tento paket tak jednoducho vie prečítať tieto dáta.

Server neustále počúva na zvolenom porte a keď príjme dáta od klienta tak následne pošle aktualizovane dáta ostatným hráčom.

## Vlákna

Pri spustení servera a po počiatočnej inicializácií sa fungovanie rozdelí na 2 vlákna. Prvé vlákno slúži na počúvanie (čaká na správy od klienta). Druhé vlákno slúži na odosielanie dát. Zvolili sme tento prístup aby mohol nezávisle čakať na správy od klienta a zároveň aj odosielať aktuálne správy ostatným klientom.

## Synchronizačný problém

Hlavný synchronizačný problém bol taký že dáta sa aktualizovali na jednom vlákne ale zároveň odosielali na druhom. Toto sme vyriešili pomocou Mutexu. Čiže po použití tohto prvku už nemohlo nastať že by sa dáta čítali a zároveň prepisovali v jeden časový okamih.

## Ďalšie problémy v rámci semestrálnej práce

Pri serverovej časti aplikácie bolo ďalším hlavným problémom prísť na to ako často posielať aktualizovane dáta klientom s tým aby hra bežala plynulo ale zároveň aby sa nevyťažovala kapacita siete na 100%. Pretože keď sme pustili komunikáciu medzi klientom na rozlične vlákna bez žiadnej prestávky/sleepu tak nastalo extrémne veľké vyťaženie siete.

Takže sme zvolili prístup že vlákno ktoré počúva na správy od klienta bude bežať neustále ale vlákno ktoré odosiela aktualizovane správy klientom bude fungovať iba potom ako príjme nejakú zmenu od klienta. Toto sme docielili pomocou mutexu a condition\_variable.

## Štruktúra triedy Client

* + 1. Client(int id, std::string pName, float xPosition, float yPosition, DIRECTION direction, unsigned short port, sf::IpAddress ipAddress)

Konštruktor triedy Client. Inštancia triedy Client predstavuje jedného hráča. Inicializuje počiatočne nastavenia hráča ako je jeho meno, ID, x a y pozícia, smer, IP adresu a port.

* + 1. ~Client()

Deštruktor triedy Client. Dealokuje všetky prostriedky a nastavý ich hodnotu na nullptr.

* + 1. POSITION\* getPosition()

Getter pre štruktúru pozícia. Týmto gettrom sprístupní klientove súradnice a jeho smer.

* + 1. CONNECTION\* getConnection()

Getter pre štruktúru pripojenia. Týmto gettrom sprístupní klientovu IP adresu a port.

* + 1. bool getFired()

Getter na informáciu či klient vystrelil. Vracia bool.

* + 1. bool wasKilled()

Getter na informáciu či bol klient zabitý. Vracia bool.

* + 1. bool getLeft()

Getter na informáciu či sa klient odpojil od hry. Vracia bool.

* + 1. bool wasScoreSent()

Getter na informáciu či bola hráčovi poslaná štatistika po odpojení. Vracia bool.

* + 1. int getClientId()

Getter na klientove ID. Vracia int.

* + 1. int getKilledBy()

Getter na ID hráča ktorý zabil tohto klienta. Vracia int.

* + 1. int getScore()

Getter na skóre hráča. Vracia int.

* + 1. void updatePosition(float xPosition, float yPosition, DIRECTION direction)

Funkcia na aktualizovanie aktuálnej polohy hráča. Aktualizuje štruktúru POSITION.

* + 1. void setInitialPosition(float xPosition, float yPosition, DIRECTION direction)

Funkcia na nastavenie počiatočnej polohy hráča ktorá sa neskôr používa na respawn po zabití.

* + 1. void setFired(bool fired)

Funkcia ktorá slúži na nastavenie či hráč vystrelil alebo nie.

* + 1. void resetPosition()

Funkcia ktorá slúži na resetovanie pozície hráča po zabití.

* + 1. void killed()

Funkcia ktorá slúži na nastavenie informácie že hráč bol zabitý.

* + 1. void setLeft(bool left)

Funkcia ktorá slúži na nastavenie či hráč opustil hru.

* + 1. void lockMutex()

Funkcia slúži na uzamknutie mutexu.

* + 1. void unlockMutex()

Funkcia slúži na odomknutie mutexu.

## Štruktúra triedy Application

* + 1. Application()

Konštruktor triedy Application. Inicializuje potrebné polia a parametre.

* + 1. ~Application()

Deštruktor triedy Application. Dealokuje všetky inicializované atribúty.

* + 1. void run()

Funkcia ktorá spustí hru. Na začiatku sa inicializujú všetky potrebné sieťové prvky, následne sa počká na pripojenie hráčov, pošlú sa im nastavenia hry a v poslednom rade sa už iba rozdelí prevádzka na vlákna. Jedno pre počúvanie na správ od klientov a druhé na odosielanie správ klientom.

* + 1. void initializeSocket()

Funkcia ktorá slúži na inicializovanie socketu. To znamená že sa alokuje port na ktorom bude server počúvať.

* + 1. void waitForClients()

Funkcia ktorá si vyžiada vstup na koľko klientov má čakať (koľko bude hráčov) a následne čaká na ich pripojenie kde im hneď aj pošle ich ID a približné súradnice kde sa majú postaviť.

* + 1. void updateOfTanksPositions()

Funkcia ktorá slúži na presne nastavenie počiatočnej pozície hráčov. Klient pošle serveru presne súradnice ktoré sú prepočítané na základe od veľkosti okna hráča.

* + 1. void initializeGame()

Funkcia ktorá slúži na poslednú časť inicializácie hry. Server rozpošle všetkým klientom informácie o ich súperoch a to aké je ich ID, meno, pozícia a smer.

* + 1. void sendData()

Funkcia vlákna ktorá slúži na odosielanie dát klientom.

* + 1. void receiveData()

Funkcia vlákna ktorá slúži na prijímanie dát od klienta.