Armoured Warfare Simulator 2D 2014

Harry Ward

November 8, 2013

Contents

1 Analysis

1.1 Introduction

In 2013, a study by The NPD group showed that 68% of all PC owners used their computer for online gaming [1], making it the largest user base in the PC market. The problem comes with the fact that high-powered gaming machines are costly, and most users cannot play triple-A, high-fidelity releases on their home PC. This fact means that popular multiplayer games often leave some players unable to join in.

My client is Slick Muffin Studios[2], a small game developer who specialise in creating 2D games. I have worked with the client in the past, making small scale games. They have approached me to produce a multiplayer game in their style that can run on all machines made in the last 10 years in order to answer a growing request from the userbase for a game they can play with friends. My point of contact to the company is the creative director Jessie Canfer who has been with the company since it's founding in 2012.

My user group includes those who cannot play the full release of a game due to the machine specifications required to run it at a reasonable frame per second count. These potential consumers vary greatly in technical knowledge, from expert to beginner, although all know how to use a basic program.

In short, my aim is to produce a well-optimised 2D multiplayer game able to run on most systems.

1.2 Investigation of User Needs and Acceptable Limitations

1.2.1 The Current System Analysis

The problem that I am solving is that there is no current system- the user cannot play multiplayer games on their PC at a reasonable FPS. As such there is no current system to analyse.

1.2.2 Questionnaire

I asked various PC owners about their computer usage and specifications of their machines. This will help me set my limitations. Due to my user group, some may have lower specification PCs than others; In order to provide to all users, I will need to establish to lowest specification.

Question	User 1	User 2	User 3	User 4	User 5
Do you own a PC?	Yes	Yes	Yes	Yes	Yes
Do you play online games?	No	Yes	Yes	Yes	Yes
Do you have a dedicated graphics card?	No	No	No	Yes	No
Can you play new releases?	N/A	Yes(barely)	No	Yes	No
Are graphics important to you?	No	No	No	No	No
What operating system do you use?	GNU/Linux	Win8	Win7	Win8	WinVista

From this I have found out that graphics are not an issue in the games my users play, I expected this as the users want to play the games and cannot normally, as shown by the fact that 3/4 users have trouble playing new releases. I have also found that my system will have to be compatable with windows versions Vista onwards, as everyone interviewed uses windows, as well as other major operating systems including GNU/Linux.

1.2.3 The Proposed New System Analysis

The client has asked me to interview a potential user of my solution, from it they hope to figure out what the average user enjoys and wants from a multiplayer game. The client also hopes to find out what style of game is enjoyable on low-end computers, to give direction designing the solution.

Interview with Fuego Gitano, low-end PC user
Q: What is most important to you in a multiplayer game?
A: I want fun mechanics with a small-ish team size, also a reliable connection - it's not fun having lots of lag. There should be end-game content to keep players interested as well.

Q: End-game content? What do you mean?
A: I rather like progression, working up to gradually better things, but once you reach the top there is nothing else to do. More content when you reach this end keeps me interested and playing.

Q: What graphics do you expect from a multiplayer game?
A: From my machine, I expect at least a detailed or stylised 2d environment, 3d isn't always reliable as my FPS can drop quite a bit. Other than that, a simple style works quite well.

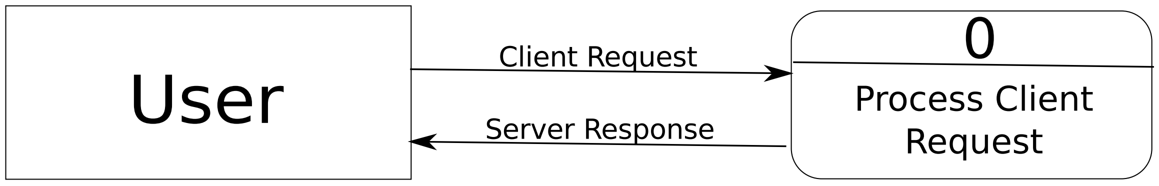
Q: What genres of game do you think work well for multiplayer?
A: I enjoy shooters and games involving individual skill. I also like fast-paced gameplay.

Q: Do you like luck being a mechanic of the game?
A: No, that ruins it for me. I like predictable gameplay .

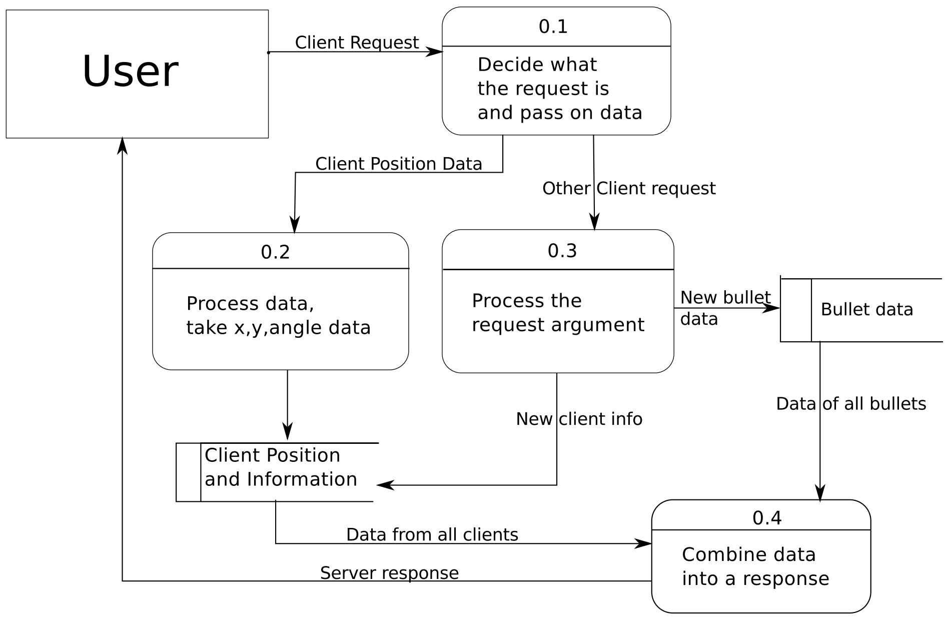
Q: Is there anything else you enjoy in multiplayer games?
A: I enjoy an element of customisation, visual mainly. Strategy is also fun, quick decisions leading to either victory or defeat can be really good. Also winning should be secondary to having fun, and the game should allow for this by not focusing too much on victory. That is everything, I think.

From this interview I have found that my typical user puts empasis on mechanics over graphics, which I expected as my user group does not have dedicated graphics cards, so have come to appreciate good mechanics. The user also wants there to be reason to carry on playing, to progress and to have something to work towards, this works nicely as the user enjoys shooters and progression is a mainstay in this genre. This combined with the user wanting a strategic game leads me to propose a top-down 2D armoured warfare game. It's will comprise of small-sized teams of perhaps 4 and an element of customisation as per the user's answers from the interview.

Level 0 Data Flow Diagram



Level 1 Data Flow Diagram



There is only one entity in my data flow diagram, as information is both taken from and given to the client. There are to be multiple clients entities connected to the server at one point, up to 8. The server will act as a data source, holding data for all connected clients and sending it to them on request.

Data will be stored on the server for user accounts, this will be a <10MiB database, and on the client side player sprites will be stored in PNG format for transparency, whilst the maps will likely be stored in JPG format along with a text file to store map collision data, i.e. where building are. These sprites could take up between 10 and 20 MiB of data, and the client/server collection will be under 30MiB. It will be able to be stored on any medium due to this size.

1.2.4 Data dictonaries

Below are the data dictonaries for my proposed system

Login database

Field name	Field purpose	Field Type	Example Data	Validation
Username	To store the users alias	String	FloatingGhost	Not null
Password	To enable secure login	String/Hashed	640ab2bae07bedc	Not null
XP	To store the user's progress	Int	7	Not null
Tanks	To store the user's unlocked tanks	String	PanzerIV,Maus	Not null
Wins	To store the users number of wins	int	7	Not null

Server variables

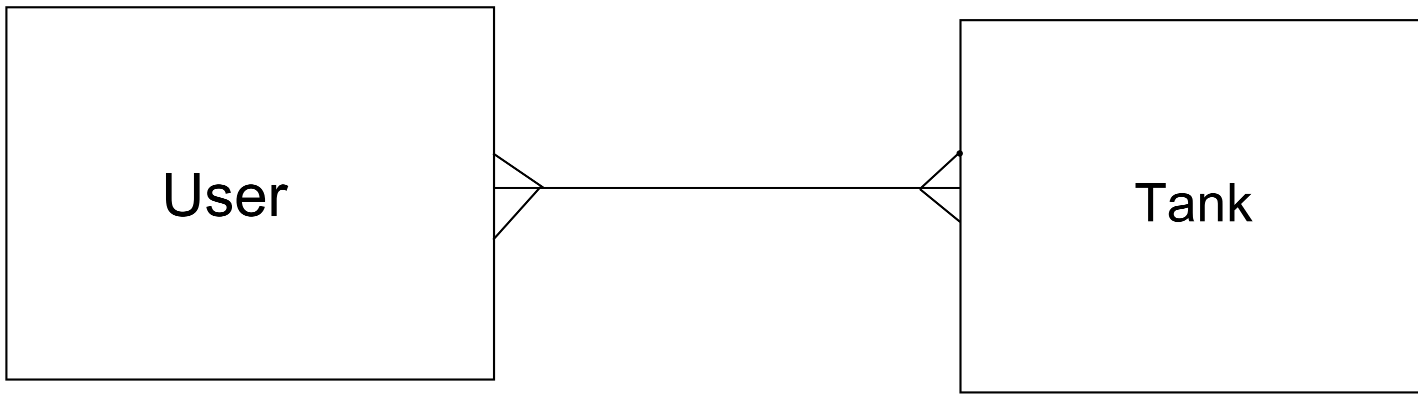
Field name	Field purpose	Field Type	Example Data	Validation
Id	To determine which client is connected	Int	7	Not null
x_pos	To store the clients x position	List of floats	[7.7,7.7]	Not null
y_pos	To store the clients y positions	List of floats	[7.7,7.7]	Not null
angles	To store the clients angles	List of floats	[77,77]	Not null
turret_angles	To store the clients turret angles	List of floats	[77,77]	Not null

Client variables

Field name	Field purpose	Field Type	Example Data	Validation
Id	To tell the server who you are	Int	7	None
x_pos	To draw every player on the screen	List of floats	[7.7,7.7]	None
y_pos	To draw every player on the screen	List of floats	[7,7,7.7]	None
angles	To draw every player on the screen	List of floats	[77,77]	None
turret_angles	To draw every player on the screen	List of floats	[77,77]	None
networkComms	To communicate with the server	Custom class	((“localhost”,9999))	None

1.2.5 Database design

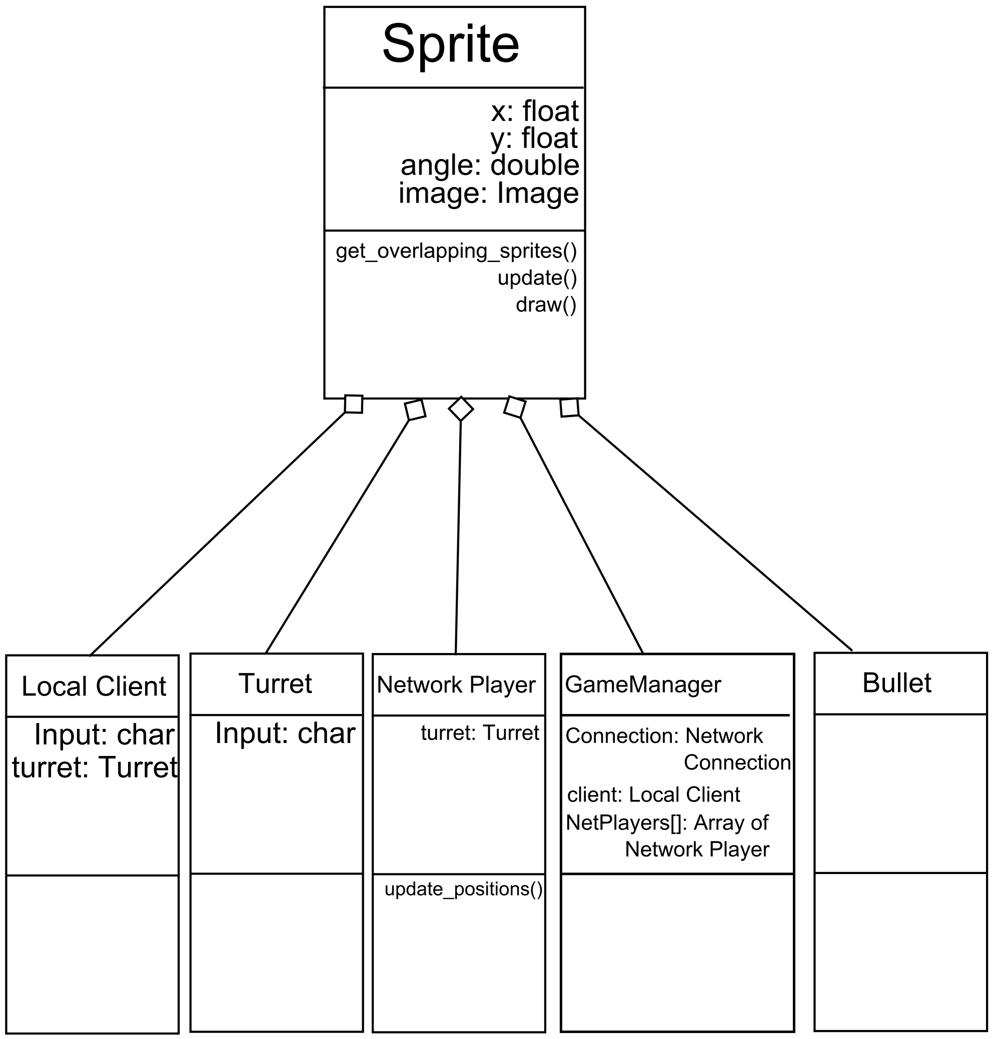
Main database design



User(UserID, username, hashedPassword, panzerIVProgress, churchillProgress, ... , [Tank]Progress, xp)

Tank(TankID, name, speed, hp, traverseSpeed, turretTraverseSpeed, damage, rateOfFire, armour, penetration)

1.2.6 Proposed objects



1.2.7 Data Sources and Destinations

Sources	Destinations
Player send client information	Server stored updates client information
Server send all player information	Player updates the network players, output in form of drawing player
Tank Database gives the tank stats	Client takes it's stats to correctly update it's own information, r
Player gives input to the client	Outputted to the server in the form of updates client inform

1.3 Constraints

1.3.1 Hardware

This is the main constraint of my proposed system as the problem the client has given me exists due to low-powered hardware. My solution will have to run on low-end integrated graphics systems at >30 FPS to fit with the client's request as expressed in the interview. It must also be able to host the server on both WiFi and Ethernet LAN to cover every possible machine.

Another possible hardware constraint is network interfaces - as the game will run over a network, it will need low latency to give a pleasant experience. Low-speed network interfaces are an issue due to the time taken for the client to send and recieve requests and responses respectivelty from the server.

1.3.2 Software

The proposed system is to be cross-platform, so it must use a programming language that is also cross-platform as well as any external libraries.

1.3.3 Time

The project must be finished by the deadline, easter 2014.

1.3.4 User's Knowledge Of Information Technology

As my user group mainly plays games online, they are famillier with how to use a program. But to ensure that everyone on all systems can use my solution, I will keep the program as jargon-free as possible and as simplistic as I can make it.

1.3.5 Who will be allowed to use the system?

The client can be used by anyone, whilst the running server can only be accessed by the owner of the PC running it. User accounts can only be accessed by their respective owners, so there will have to be some way to log in to ensure security.

1.4 Objectives

1.4.1 General Objectives

My client's overall objective is to create a fun, multiplayer armoured warfare game that has depth and end-game content. My client wants it to run on any system, regardless of computer specification, whilst making the game to the users satisfaction.

1.4.2 Specific Objectives

1. By deadline day, my game client must be able to communicate across the network with low latency.
2. By deadline day, it must also be able to run on a low-power PC at more than 30 Frames Per Second.
3. By deadline day, it must also have user accounts with progression.
4. By deadline day, it must also support up to 8 players.
5. By deadline day, it must also be supported on Windows and Linux with less than a 5 minute installation time.

1.5 Consideration of Alternative Solutions

- I could use an alternative language such as Java as it is in common use, installed on almost all PCs and is cross platform. This would fit my specification as the language is compiled, making it run more effectively on low-end computers. I am also familliar with the language and how to create games in it, although networking is one of the hardest features of Java.
- I could use a WYSIWYG game creator such as gamemaker as it is simple to use and speeds up production significantly. It is also easy to learn and has networking extensions to allow for ease of creation.

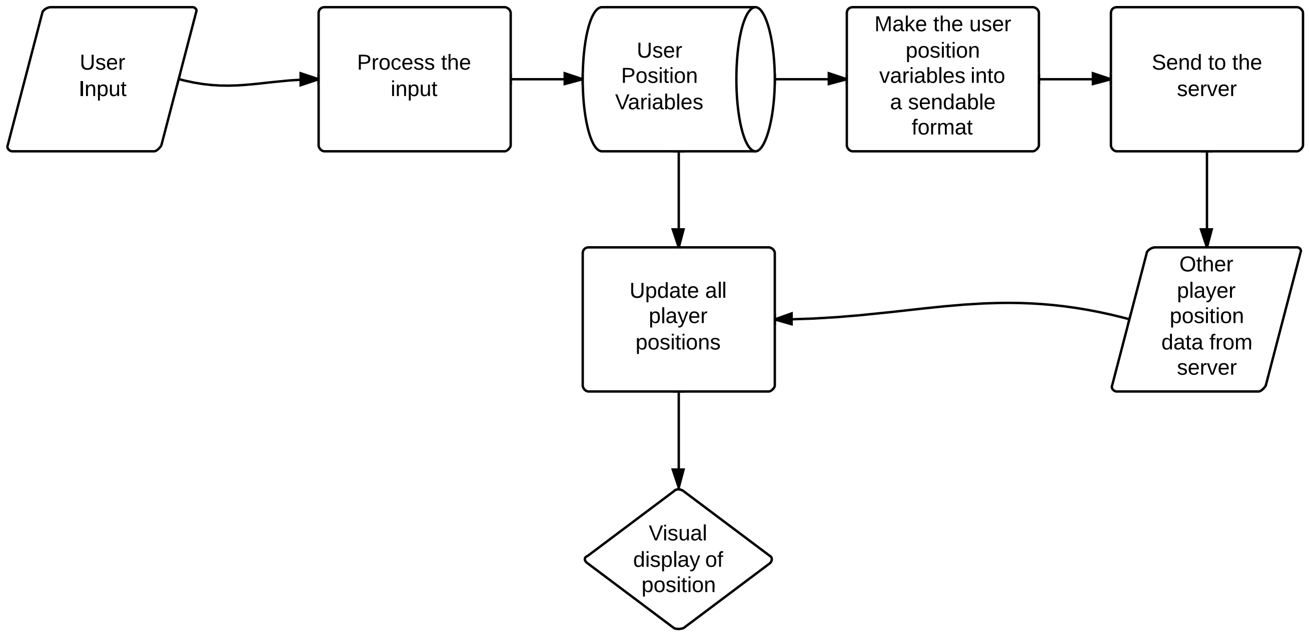
1.6 Justification of Chosen Solution

I have chosen to use Python with PyGame for the client and pure Python for the server, whilst using wxPython for the GUI forms, such as the server configuration form. This is due to my knowledge of the language and the ease of networks with inbuilt modules. PyGame also makes game design more intuitive and efficient, making it able to run on most any PC. wxPython is a cross-platform GUI toolkit that includes a WYSIWYG designer and python modules, making the creation of GUI forms simpler.

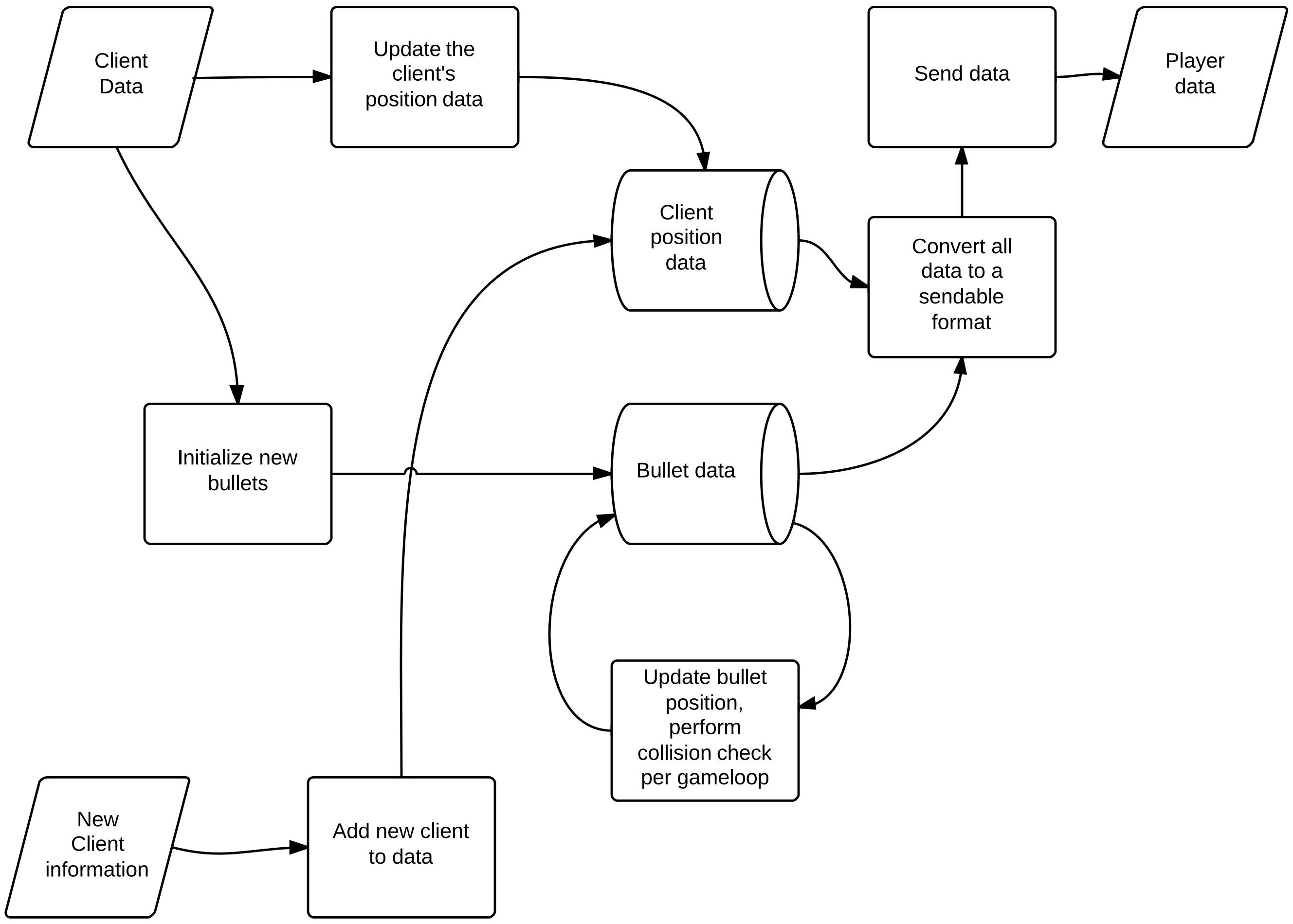
2 Design

2.1 Overall System Design

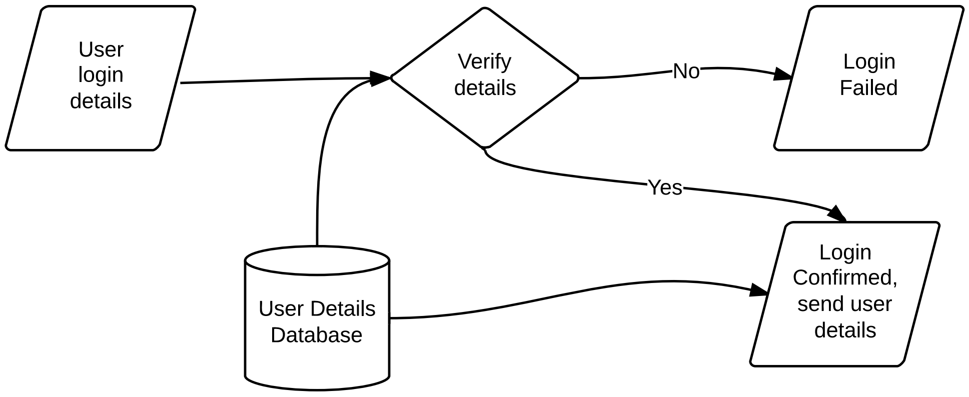
Client Side


User Input
Process the input
User Position Variables
Make the user position variables into a sendable format
Send to the server
Other player position data from server
Update all player positions
Visual display of position

Server Side

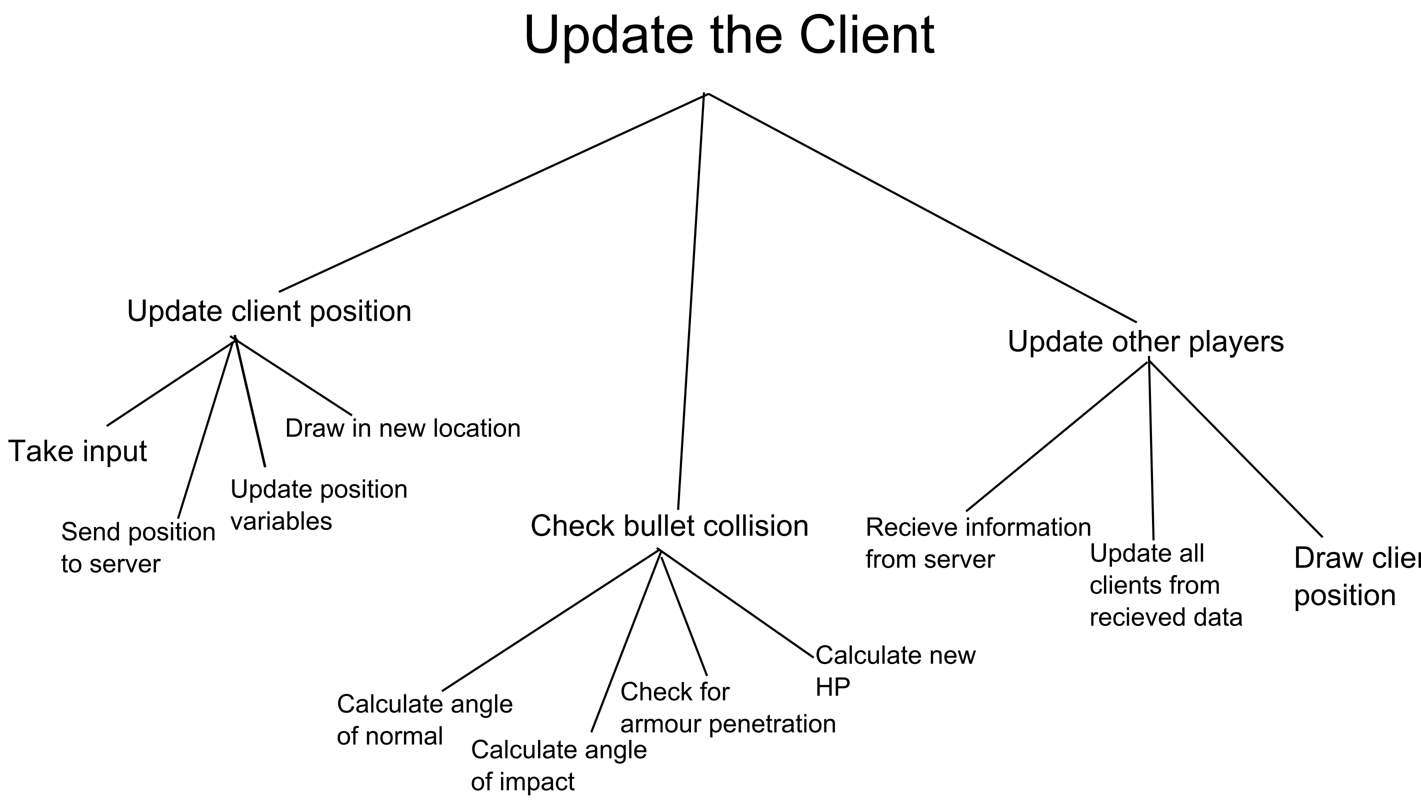

Client Data
Update the client's position data
Send data
Player data
Client position data
Convert all data to a sendable format
Initialize new bullets
Bullet data
Update bullet position, perform collision check per gameloop
New Client information
Add new client to data

Login Server



2.2 Description of Modular Structure of System

Top-down design for the system


Update the Client
Update client position
Take input
Send position to server
Update position variables
Draw in new location
Check bullet collision
Calculate angle of normal
Calculate angle of impact
Check for armour penetration
Calculate new HP
Update other players
Recieve information from server
Update all clients from recieved data
Draw clien position

IPSO Table for the proposed system, server-side:

Inputs	Processes
Username	Sign in player
Password	Search for player in database
Tank Choice	Process the client request
X position	Calculate bullet trajectories
Y position	Calculate if the bullet penetrates
Hull angle	Calculate the HP of hit tanks
Turret angle	Update all connected players
Bullet creation	Update bullet position
Client Request	Calculate angle of bullet impact
Client connection	

Outputs	Data Stores
Client positions	Client information
Bullet positions	Bullet information
Client HP	User information (stats)
Client XP and progress	
Network response	
Local server information	

2.3 Definition of data requirements

A table of a database is used to store the details of the users. It uses this structure:

UserID: Integer
Username: String[20]
HashedPassword: String[100]

Another table is used to store the user progress on every tank, with the following structure:

UserID: Integer
PanzerIVProgress: String[30]
.
.
.
ChurchillProgress: String[30]

The various tanks are stored in a third table with this structure:

TankID: Integer
Name: String[10]

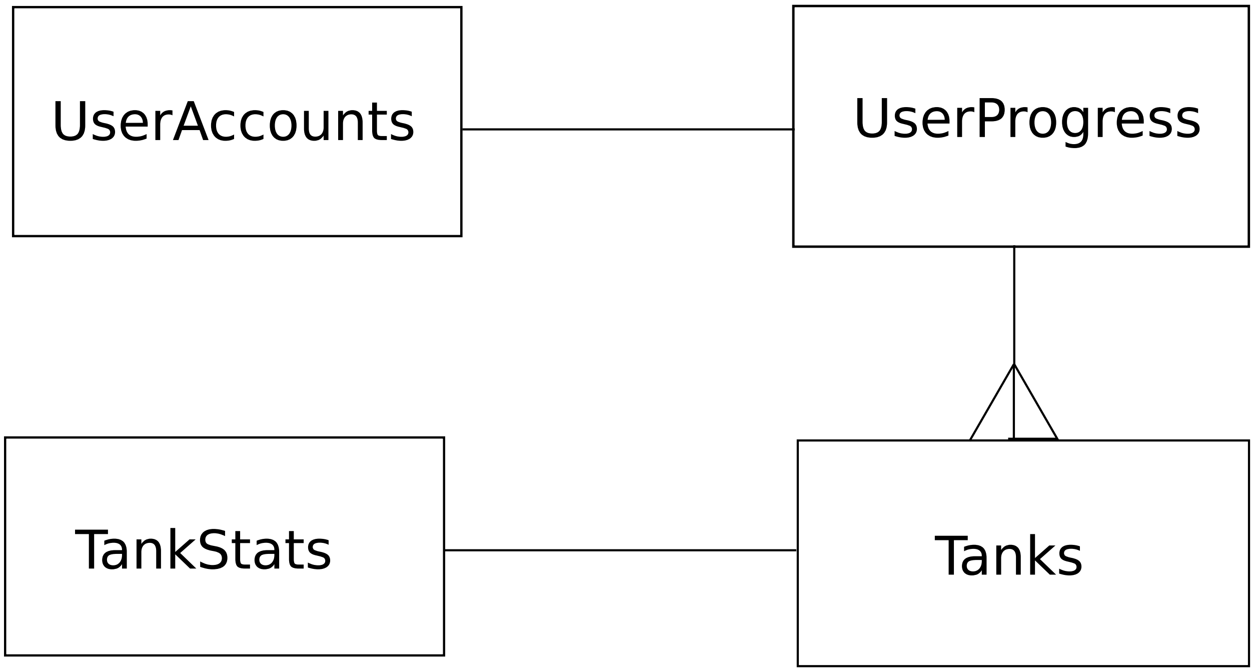
The stats for these tanks are stored in a final table with structure:

TankID: Integer

speed: Float
hp: Integer
traverseSpeed: Float
turretTraverseSpeed: Float
damage: Integer
reload: Integer
armour: Integer
penetration: Integer

The user tables will hold approximately 200 records based on the current user base of Slick Muffin, my client, and the tank tables will hold around 20 records.

2.4 Database Design



UserAccounts(UserID, Username, HashedPassword)
UserProgress(UserID, PanzerIVProgress, ChurchillProgress, (etc) [Tank]Progress)
Tanks(TankID, Name)
TankStats(TankID, speed, hp, traverseSpeed, turretTraverseSpeed, damage, reload, armour, penetration)

2.5 Storage

2.5.1 File Organisation and Processing

From my data dictionary, my database will likely require less than 5 MiB of storage space based on 200 users, although this could scale if the game is more popular than expected. As such I shall estimate the absolute maximum size to be 10 MiB. The game resources (pictures, sound etc.)

will take up quite a bit more room. Going on 20 tanks, I predict the total resources size to be approximately 30-50 MiB dependant on the quality of sound used.

An estimate for the final program file size would be around 60 MiB for absolutely everything.

2.5.2 Identification of Storage Media

My files will be stored on a hard drive, with data written and read in the working directory to avoid permission complications with Windows. I will use a complete backup on the server side, every day the server is online it will be copied to make a new file in the users home directory.

2.6 Identification of suitable algorithms

A large part of my game revolves around armour penetration mechanics, I.E if the bullet damages the vehicle or not. This is largely dependant on the bullet's angle of impact to the surface of the tank.

Definition 2.1. Henceforth, **A.B** shall be used to denote the dot product of vectors a A and B, where the dot product is defined as

$$A.B = A_1B_1 + A_2B_2 = |A||B|cos\theta$$

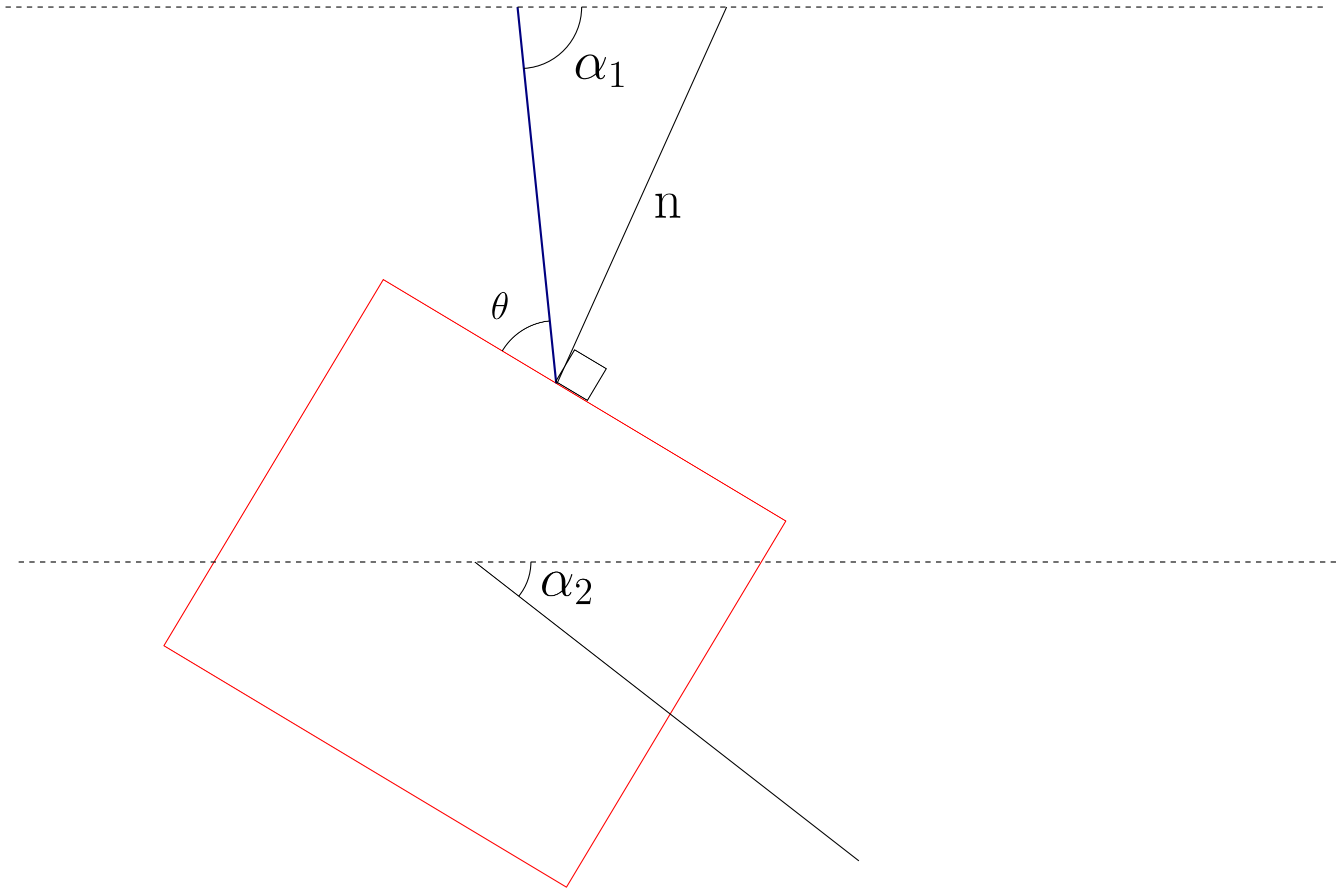
Where θ is the angle between the 2 vectors, and |A| is the modulus or length of the vector given by:

$$|A| = \sqrt{A_1^2 + A_2^2}$$

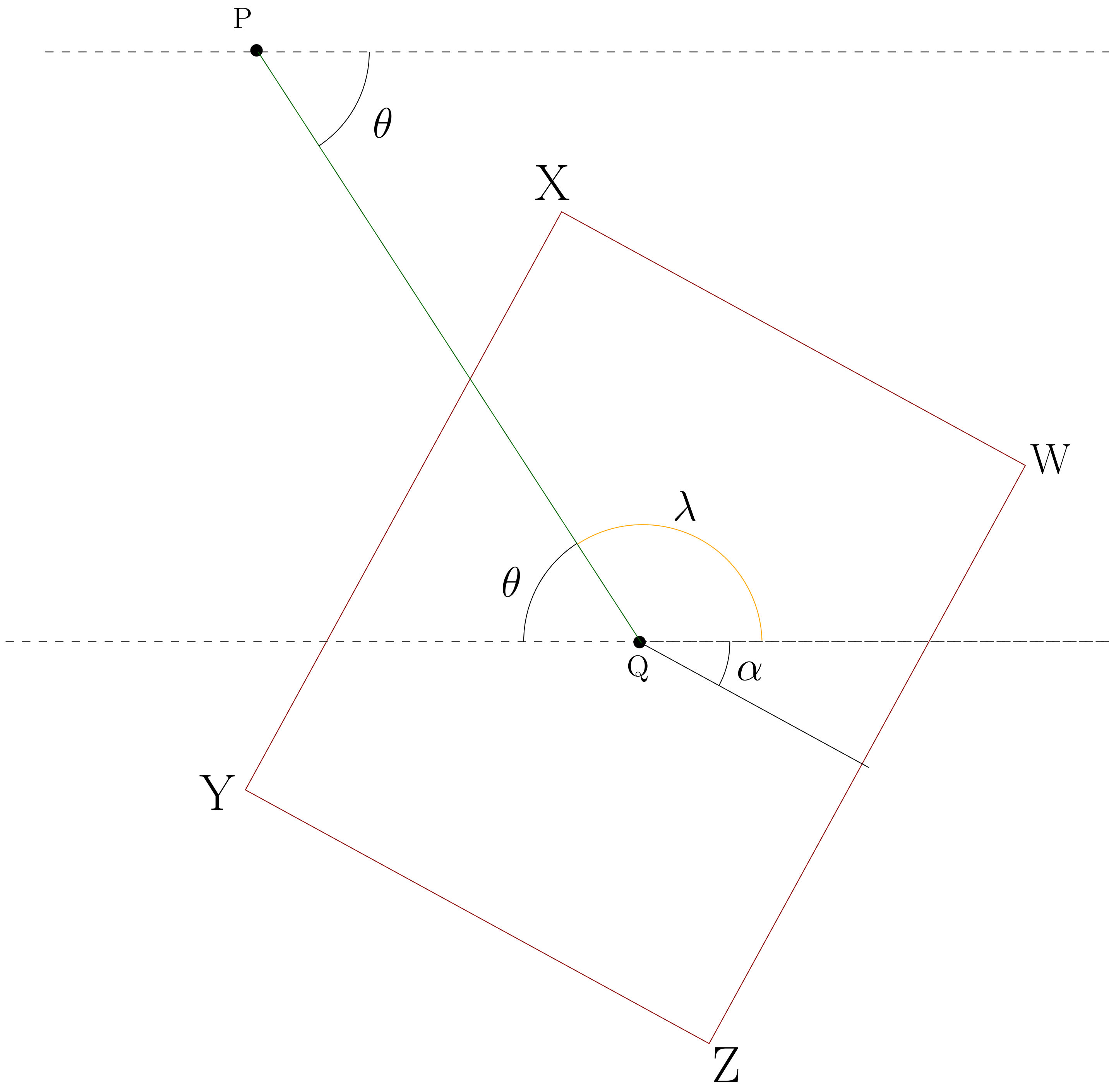
Hence:

```
DotProduct:
        VecA <- Input VectorA
        VecB <- Input VectorB
        xComponent <- VecA.x * VecB.x
        yComponent <- VecA.y * VecB.y
        output xComponent + yComponent
```

Suppose we have a tank at angle α_2 and the velocity vector of the bullet hitting it, with the angle of the velocity vector α_1, I.E:



n is the normal to the surface of impact, and it's angle is dependant on the bullet's location.



Consider the above diagram, from it we can create an algorithm that will return the angle of the normal to the surface. From the diagram, it is shown that the angle from the line PQ and the line running perpendicular through the tank is $180 - \theta + \alpha$ where θ is the angle of the bullet from the positive x-axis and α is the angle of the tank from the same axis.

From this we want to find what angle the normal is in reference to the bullet, as if it hits the top surface it is at a different angle to if it hits the side.

This angle will differ based on the size of the tank's hit box, so henceforth I will use TW to denote tank width and TH to represent tank height, with the horizontal on the above diagram running through the TW side.

We will need to find where the corners of the tank are in order to have any reference to the bullet, a will be used below to denote the angle the tank is facing.

- Corner Z is given by arctan$[(TW/2)/(TH/2)]+a$
- Corner Y is given by arctan$[(TH/2)/(TW/2)]+a$
- Corner X is equal to the angle to corner Z + 180
- Corner W is equal to the angle to corner Y + 180

So, finally, the angle of the normal is this:

```
Normal:
        angleOfTank <- Input Tank.Angle

        angleOfBullet <- Input Bullet.Angle

        angleBetween <- 180 - angleOfBullet + angleOfTank

        toCornerZ <- arctan[(TW/2) / (TH/2)]

        toCornerY <- arctan[(TH/2) / (TW/2)]

        if angleBetween < toCornerZ
                output angleOfTank

        else if angleBetween < toCornerY
                output 90 + angleOfTank

        else if angleBetween < toCornerZ + 180
                output 180+ angleOfTank

        else
                output 270 + angleOfTank
```

Definition 2.2. So if we consider finding the general angle of impact, taking into account the angle of the normal, we can use the following algorithm:

```
AngleOfImpact:
        Bullet <- Input bullet

        BulletXComp <- Bullet.Velocity * cos(Bullet.angle)

        BulletYComp <- Bullet.Velocity * sin(Bullet.angle)

        BulletVector <- Bullet.Velocity * Bullet.angle
```

```
NormalLine <- Normal(Input Tank)

NormalLength <- 2

NormalXComp <- NormalLength * cos(NormalLine)

NormalYComp <- NormalLength * sin(NormalLine)

Dot <- DotProduct(BulletVector, NormalLine)

ModA <- sqrt[(BulletXComp^2) + BulletYComp^2)]

ModB <- sqrt[(NormalXComp^2) + (NormalYComp^2)]

CosTheta <- Dot / (ModA * ModB)

Output arccos(90 - CosTheta)
```

2.7 Class definitions

I probably need these. What exactly are they?

2.8 User Interface Rationale

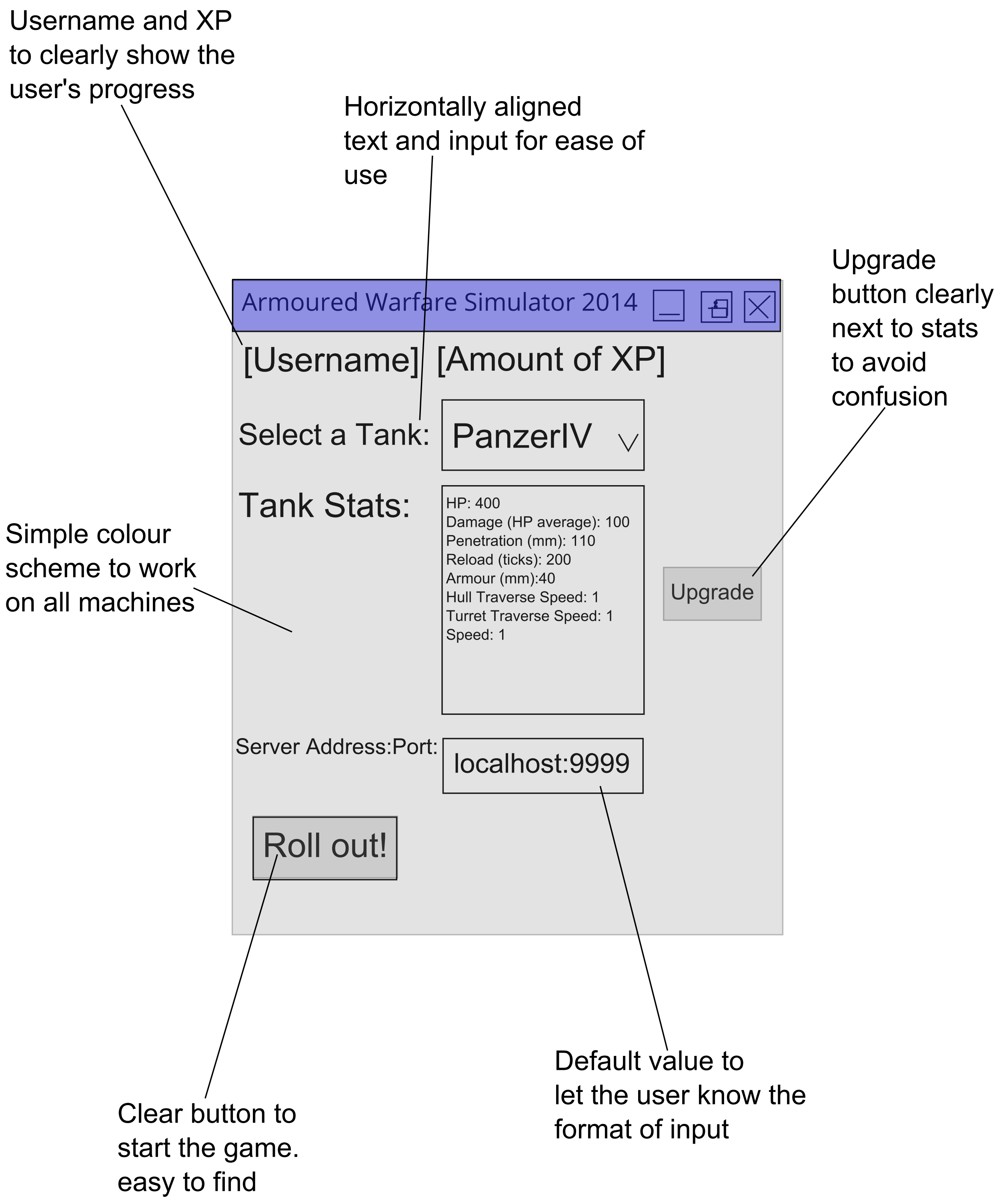
For my GUI, I shall be using as simplistic a graphics scheme as possible as the client wants the program to run on very low-powered computers and a complex GUI could possibly not run as well as intended. As such, I shall be using the default graphics scheme of the wx toolkit as it is the best optimised. I shall use a sans-serif font that can be rendered on all operating systems, the font will be Open Sans due to its clarity on smaller displays.

For multiple choice options I shall use drop-down lists as they save space, allowing for more to be on one form.

The main game shall be in 1024x768 resolution, it is universally supported which the client has said is a requirement. The other forms shall be no larger then 600x400 to avoid lack of screen space if multiple forms need to be displayed at once.

2.9 UI Sample of Planned Data Capture and Entry Designs

The main tank selection window:


Username and XP to clearly show the user's progress
Horizontally aligned text and input for ease of use
Upgrade button clearly next to stats to avoid confusion
Simple colour scheme to work on all machines
Clear button to start the game. easy to find
Default value to let the user know the format of input
Armoured Warfare Simulator 2014
Upgrade
Roll out!

[Username] [Amount of XP]

Select a Tank: PanzerIV

Tank Stats:

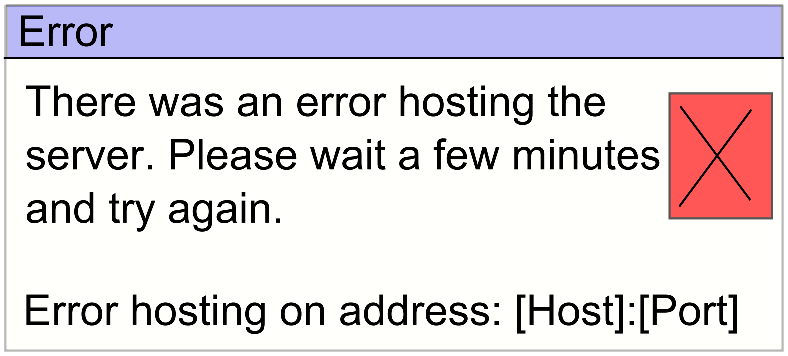
HP: 400
Damage (HP average): 100
Penetration (mm): 110
Reload (ticks): 200
Armour (mm):40
Hull Traverse Speed: 1
Turret Traverse Speed: 1
Speed: 1

Server Address:Port: localhost:9999

2.10 Description of measures planned for security and integrity of data

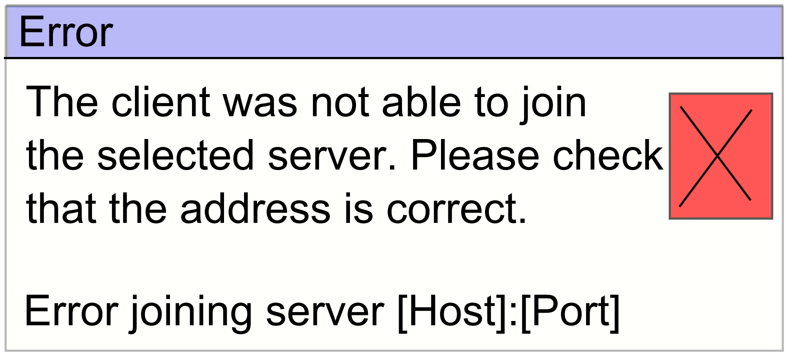
As this project is network-based, the majority of the errors come from the connection between client and server. As such user input much be validated to ensure the network connections can be carried out without error, and the "Server join" box will run a check on the input to make sure that there is a server address and that this address is reachable.

Error messages resulting from the connection will look like this:


Error

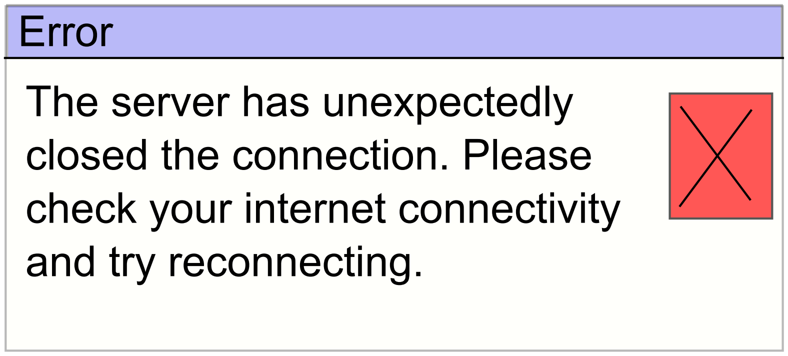
There was an error hosting the server. Please wait a few minutes and try again.

Error hosting on address: [Host]:[Port]

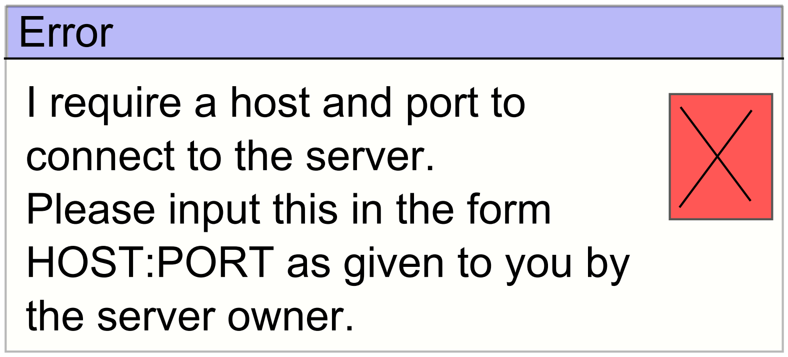

Error

The client was not able to join the selected server. Please check that the address is correct.

Error joining server [Host]:[Port]


Error

The server has unexpectedly closed the connection. Please check your internet connectivity and try reconnecting.


Error

I require a host and port to connect to the server.
Please input this in the form HOST:PORT as given to you by the server owner.

All of these errors are not recoverable directly by the program and will put the user back at the main GUI for either the server or the client for reconnection or the erronious data to be modified, in the case of the HOST:PORT error.

2.11 Description of measures planned for systems security

//Note: I'm assuming this is making the system only available to those who should have access to it. No idea what the textbook says.

The client has specified that they would like user accounts, and as such these must be secure to avoid tampering. The user database will be stored on a central server with passwords hashed using SHA-1 which is inaccessable to anyone except for the owner of the login server, the client. This ensures security of user data and progress whilst providing a central data storage location for the program to use.

Security of client-hacking is carried out through mainly server-side calculations, meaning that any modification of local code will not affect the game experience for anyone else.

2.12 Overall Test Strategy

Series	Purpose of Test Series
1	Population of the GUI - client data and network interfaces. (System Testing)
2	Network connection, do the client and server connect and stay connected? (Integration testing)
3	Database information transfer, does the client recieve the correct information? (System Testing)
4	Physics of bullets - do armour penetration calculations work? (White box testing)
5	Client draws the game correctly - do the graphics appear the same on all clients? (System testing)

References

[1] NPD Group study into PC Usage: URL: https://www.npd.com/wps/portal/npd/us/news/press-releases/the-npd-group-report-shows-increased-number-of-online-gamers-and-hours-spent-gaming/

[2] Slick Muffin Studios URL: http://slickmuffin.weebly.com/