

Armoured Warfare Simulator 2D 2014

Harry Ward

March 18, 2014

Contents

| | | |
|----------|--|----------|
| 1 | Analysis | 3 |
| 1.1 | Introduction | 3 |
| 1.2 | Investigation of User Needs and Acceptable Limitations | 3 |
| 1.2.1 | The Current System Analysis | 3 |
| 1.2.2 | Questionnaire | 3 |
| 1.2.3 | The Proposed New System Analysis | 4 |
| 1.2.4 | Data dictionaries | 5 |
| 1.2.5 | Database design | 6 |
| 1.2.6 | Proposed objects | 7 |
| 1.2.7 | Data Sources and Destinations | 7 |
| 1.3 | Constraints | 7 |
| 1.3.1 | Hardware | 7 |
| 1.3.2 | Software | 8 |
| 1.3.3 | Time | 8 |
| 1.3.4 | User's Knowledge Of Information Technology | 8 |
| 1.3.5 | Who will be allowed to use the system? | 8 |
| 1.4 | Objectives | 8 |
| 1.4.1 | General Objectives | 8 |
| 1.4.2 | Specific Objectives | 8 |
| 1.5 | Consideration of Alternative Solutions | 9 |
| 1.6 | Justification of Chosen Solution | 9 |
| 2 | Design | 9 |
| 2.1 | Overall System Design | 9 |
| 2.2 | Description of Modular Structure of System | 12 |
| 2.3 | Definition of data requirements | 13 |
| 2.4 | Database Design | 14 |
| 2.5 | Storage | 14 |
| 2.5.1 | File Organisation and Processing | 14 |
| 2.5.2 | Identification of Storage Media | 15 |
| 2.6 | Identification of suitable algorithms | 15 |
| 2.7 | Class definitions | 19 |
| 2.7.1 | Sprite Classes | 20 |
| 2.7.2 | Server Classes | 21 |

| | | |
|----------|--|-----------|
| 2.8 | User Interface Rationale | 22 |
| 2.9 | UI Sample of Planned Data Capture and Entry Designs | 22 |
| 2.10 | Description of measures planned for security and integrity of data | 23 |
| 2.11 | Description of measures planned for systems security | 25 |
| 2.12 | Overall Test Strategy | 25 |
| 3 | System Maintenance Document | 25 |
| 3.1 | System Overview | 25 |
| 3.2 | Sample algorithms | 25 |
| 3.2.1 | Angle of Intersection | 25 |
| 3.2.2 | Login Procedure | 26 |
| 3.2.3 | Get rebound angle | 26 |
| 3.2.4 | Map Generation | 27 |
| 3.2.5 | Modified Game Rendering Engine | 28 |
| 3.2.6 | Network Communication Method | 28 |
| 3.2.7 | Effective Armour Calculations | 29 |
| 3.3 | Procedure and Variable lists | 30 |
| 3.3.1 | Clientside procedure list | 31 |
| 3.3.2 | Serverside procedure list | 32 |
| 3.3.3 | Clientside Variable list | 33 |
| 3.3.4 | Serverside variable list | 34 |
| 3.4 | Annotated Samples of GUI & System Outputs | 34 |
| 3.4.1 | Tank selection screen | 34 |
| 3.4.2 | Tank upgrade screen | 35 |
| 3.4.3 | Server setup screen | 36 |
| 3.4.4 | General Error Message | 37 |
| 3.4.5 | Main Game Screen | 37 |
| 4 | Testing | 39 |
| 4.1 | Test Data | 39 |
| 4.1.1 | Login Server Test | 39 |
| 4.1.2 | Tank Selection Screen | 39 |
| 4.1.3 | Tank Purchase Screen | 39 |
| 4.1.4 | Join Game Screen | 39 |
| 4.1.5 | Bullet Calculations | 39 |
| 4.2 | Test Plan | 40 |
| 4.3 | Evidence | 41 |
| 4.3.1 | Test 1 | 41 |
| 4.3.2 | Test 2 | 42 |
| 4.3.3 | Test 3 | 43 |
| 4.3.4 | Test 4 | 44 |
| 4.3.5 | Test 5 | 44 |
| 4.3.6 | Test 6 | 45 |
| 4.3.7 | Test 7 | 45 |
| 4.3.8 | Test 8 | 46 |
| 4.3.9 | Test 9 | 47 |
| 4.3.10 | Test 10 | 48 |

| | | |
|--------|--------------------------------|----|
| 4.3.11 | Tests 11-15 | 49 |
| 4.4 | Error Log | 50 |
| 4.4.1 | Incorrect port given | 50 |
| 4.4.2 | Penetration Boundary Condition | 51 |

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 compatible 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

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

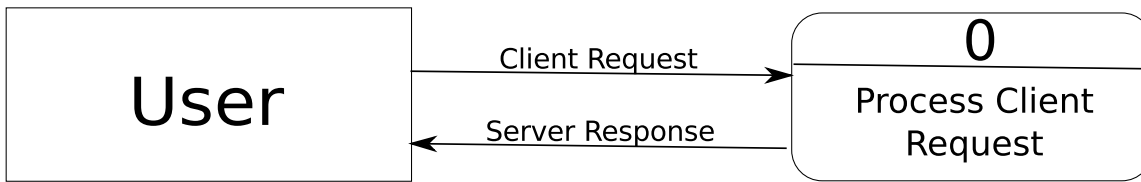
A: No, that ruins it for me. I like predictable gameplay .

Q: Is there anything else you enjoy in multiplayer games?

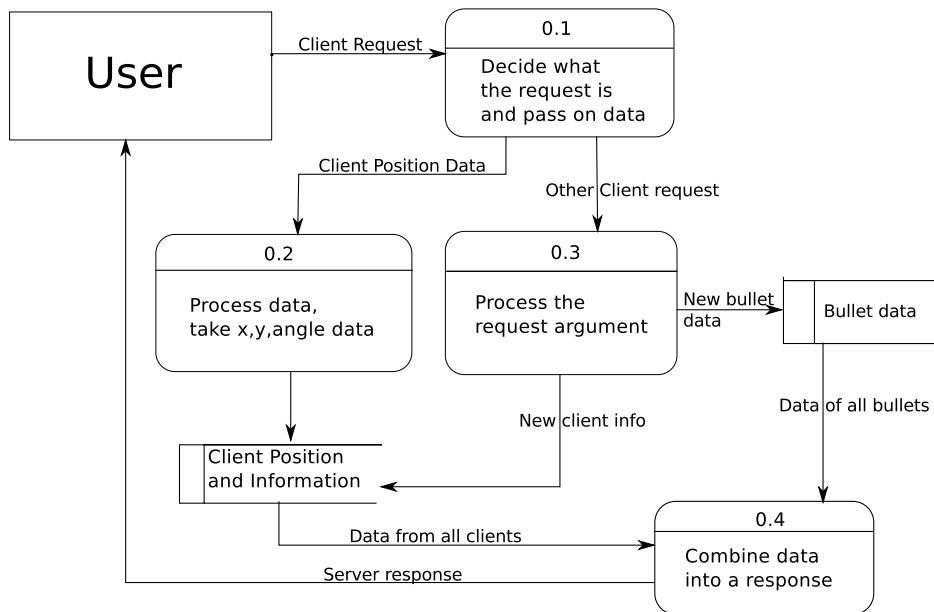
A: I enjoy an element of customisation, visual mainly. Strategy is also fun, quick decisions leading to either victory or defeat can be really good. Also winning should be secondary to having fun, and the game should allow for this by not focusing too much on victory. That is everything, I think.

From this interview I have found that my typical user puts emphasis 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 dictionaries

Below are the data dictionaries 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

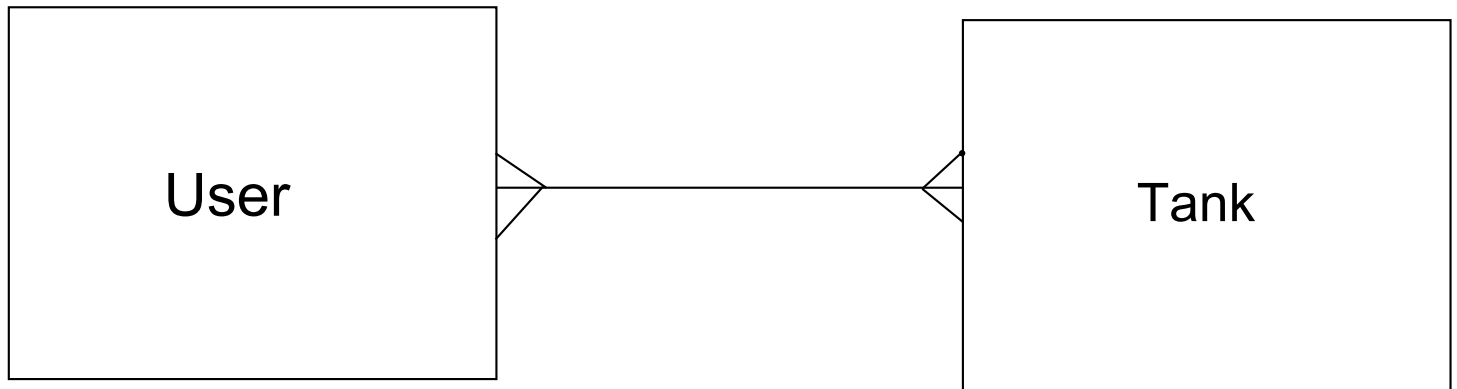
| Field name | Field purpose | Field Type | Example Data | Validation |
|---------------|--|----------------|--------------|------------|
| Id | To determine which client is connected | Int | 7 | Not null |
| x_pos | To store the clients x position | List of floats | [7.7,7.7] | Not null |
| y_pos |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

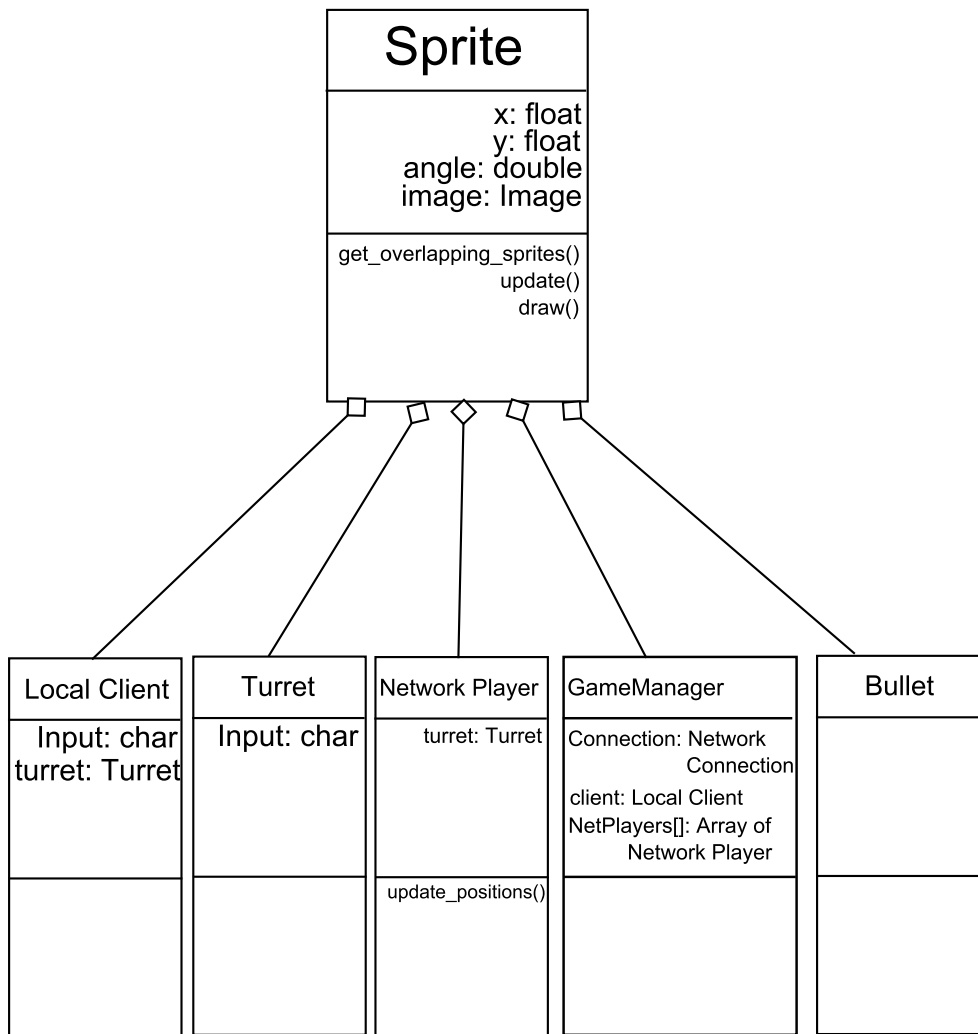
Main database design



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 players on t |
| Tank Database gives the tank stats | Client takes it's stats to correctly update it's own information, no out |
| Player gives input to the client | Outputted to the server in the form of updates client information |

1.3 Constraints

1.3.1 Hardware

This is the main constraint of my proposed system as the problem I am solving is 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 receive requests and responses respectively from the server.

1.3.2 Software

My 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 familiar 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 familiar 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.

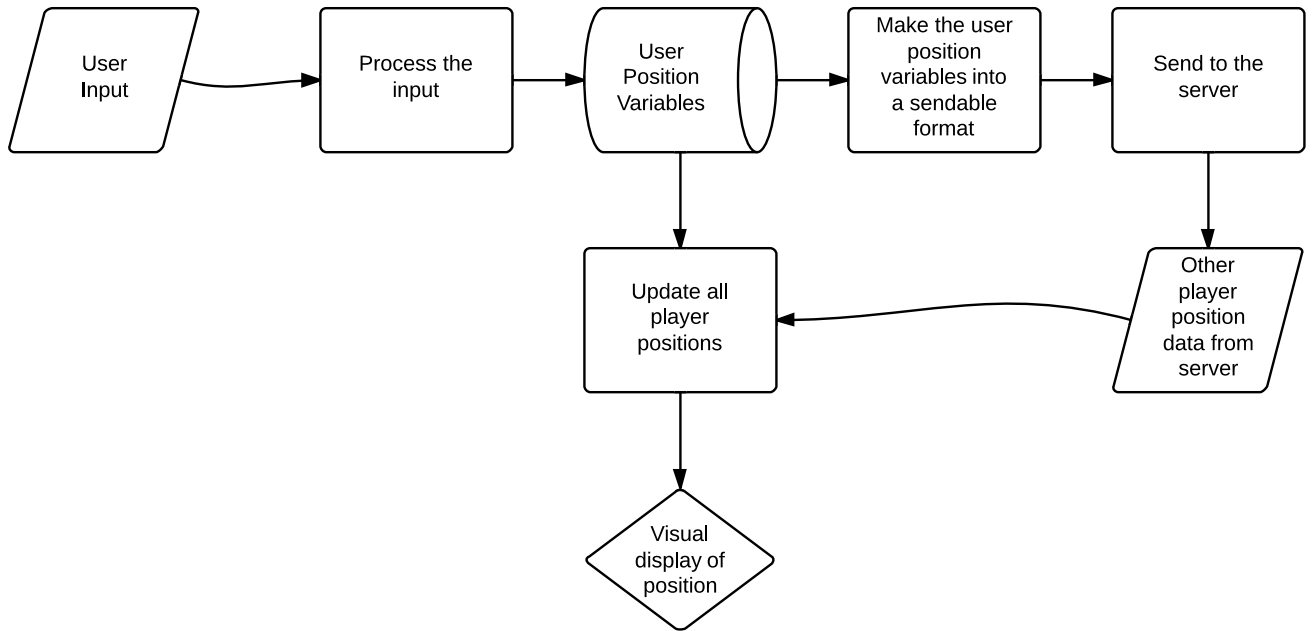
2 Design

In this section I shall outline my proposed system and its components, and explain how I shall implement each part of the solution.

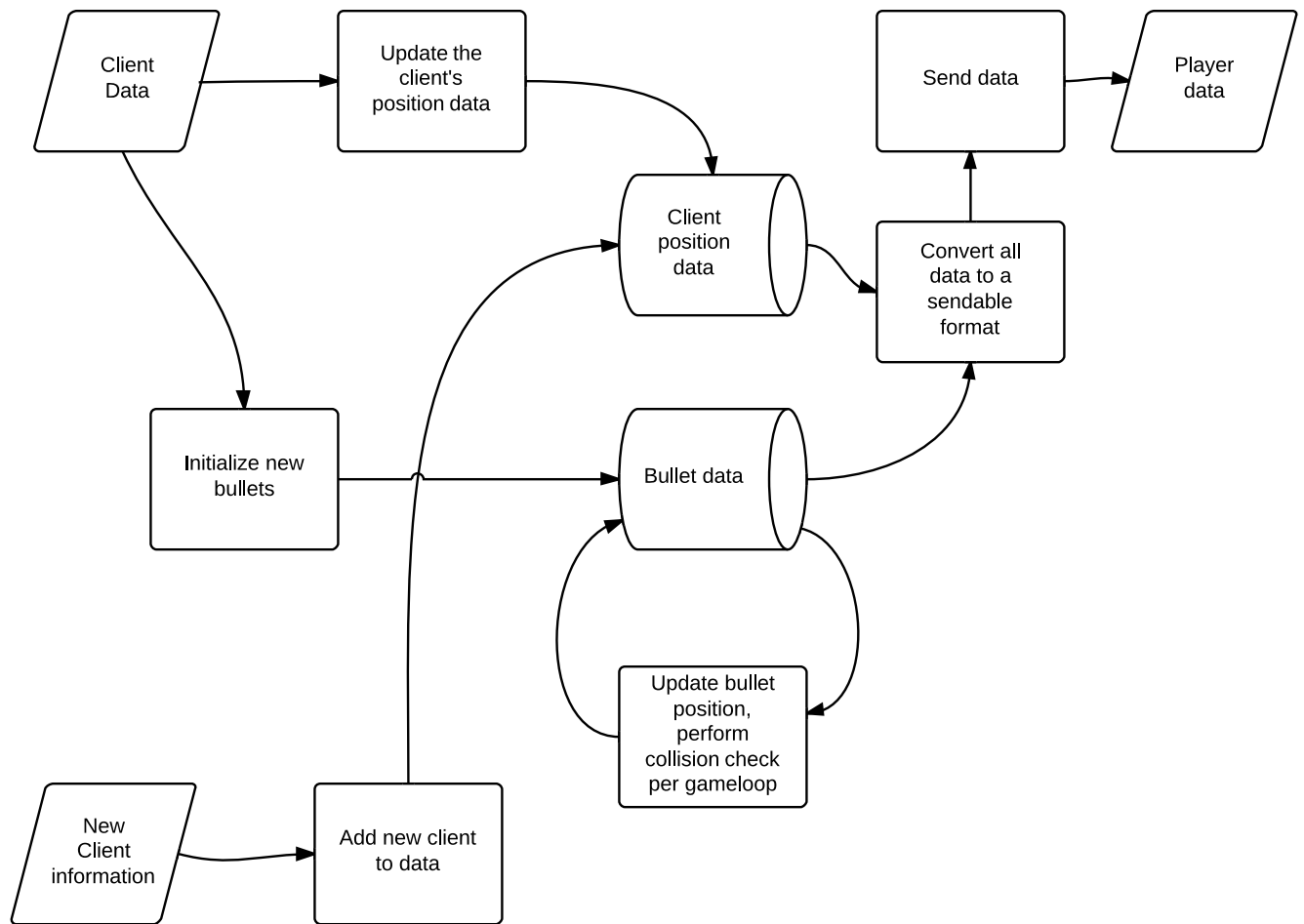
I shall start with an overall system overview, then go into more depth on the different parts of the system.

2.1 Overall System Design

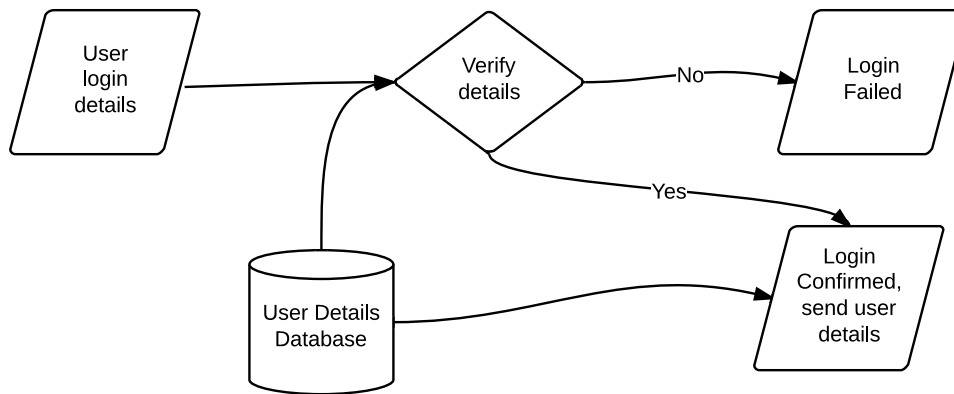
Client Side



Server Side

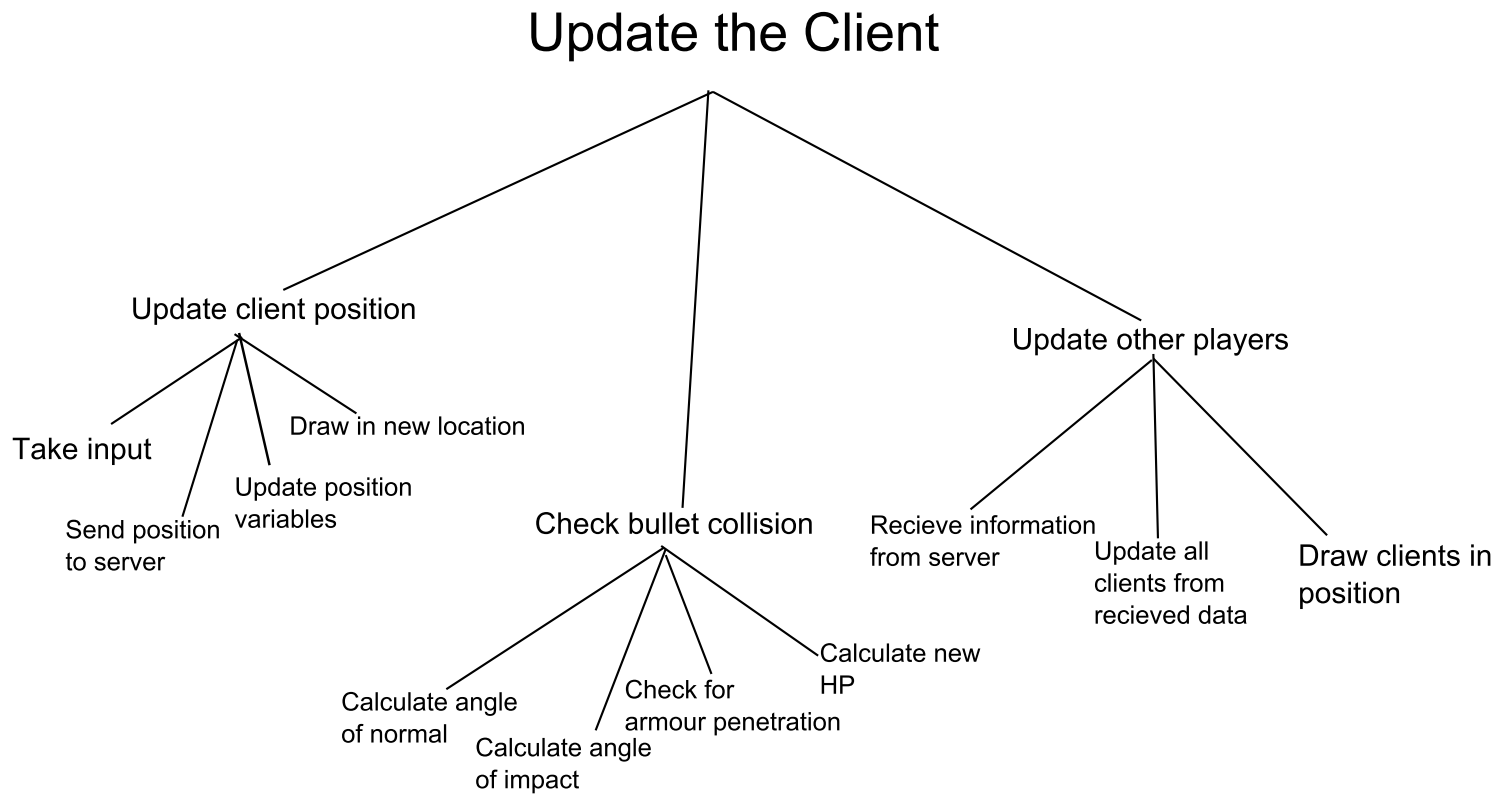


Login Server



2.2 Description of Modular Structure of System

Top-down design for the system



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]

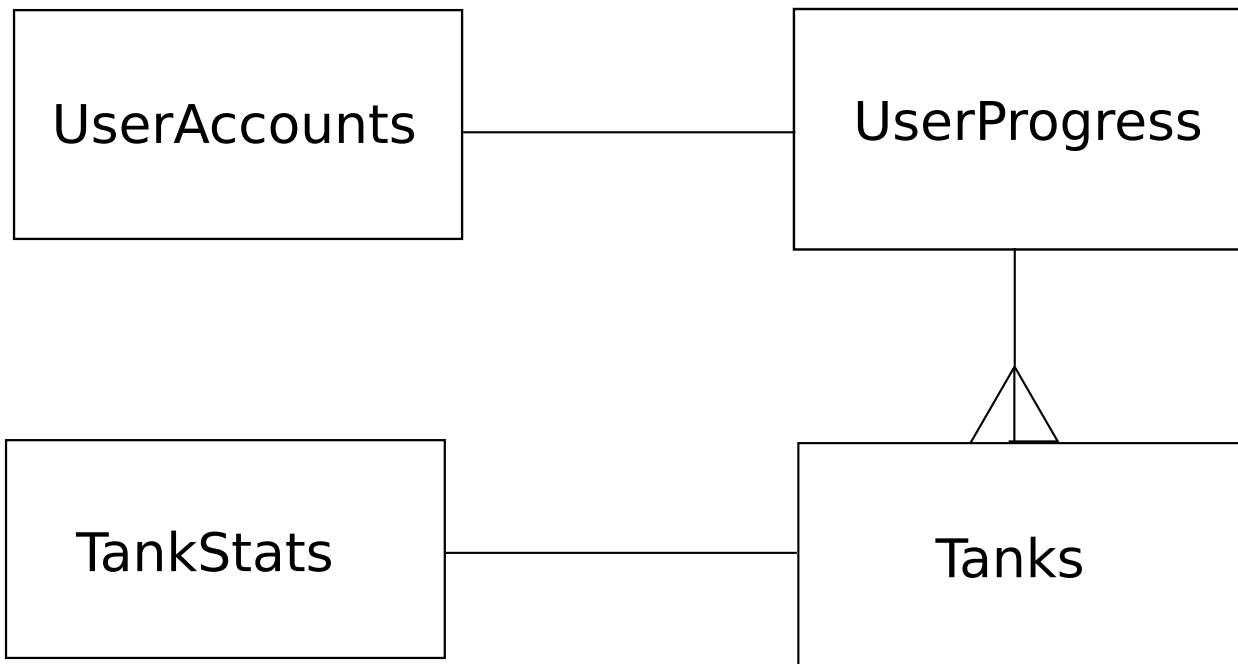
The stats for these tanks are stored in a final table with structure:

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, $\mathbf{A} \cdot \mathbf{B}$ shall be used to denote the dot product of vectors \mathbf{A} and \mathbf{B} , where the dot product is defined as

$$\mathbf{A} \cdot \mathbf{B} = A_1 B_1 + A_2 B_2 = |\mathbf{A}| |\mathbf{B}| \cos \theta$$

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

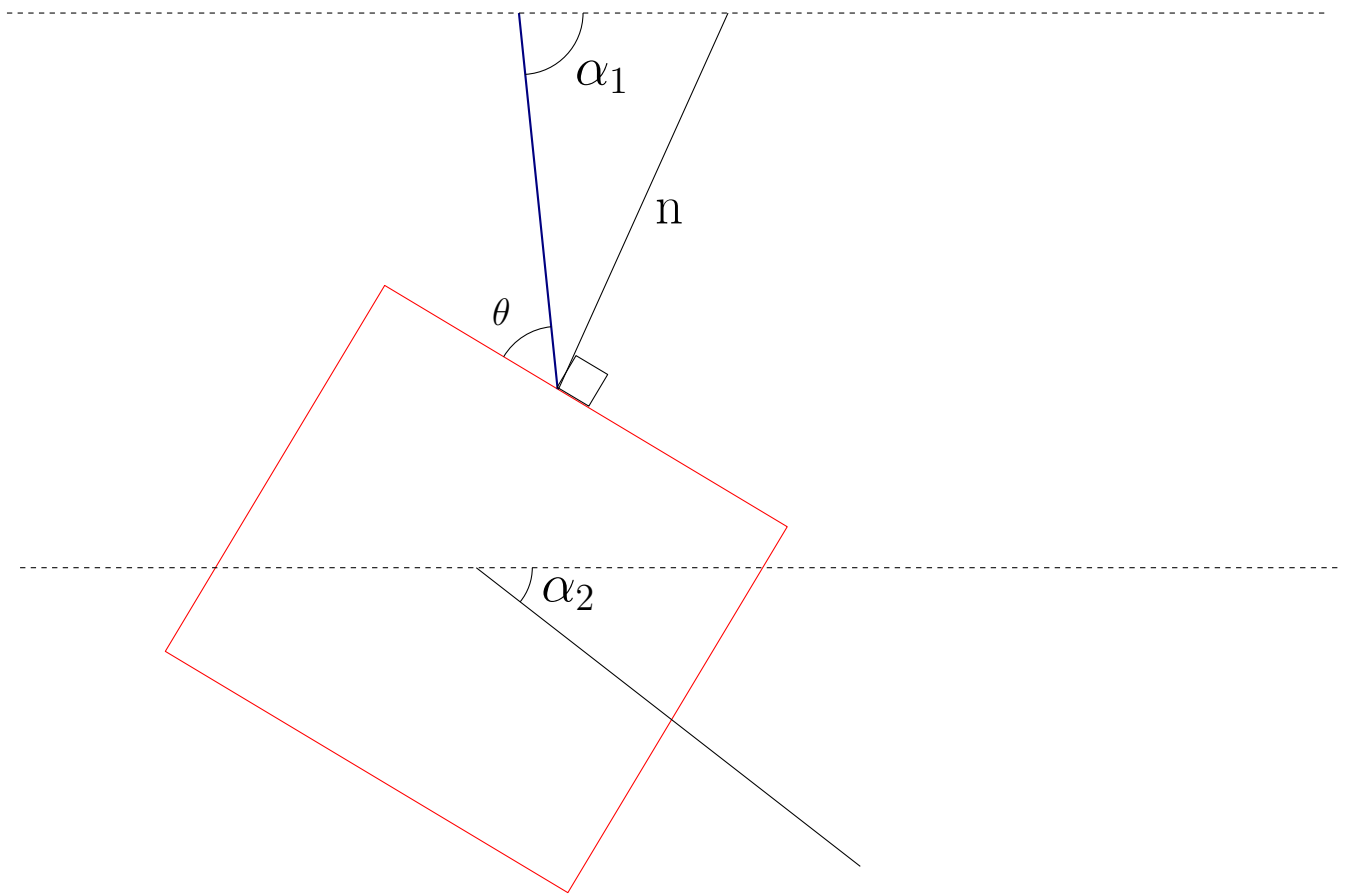
$$|\mathbf{A}| = \sqrt{A_1^2 + A_2^2}$$

Hence:

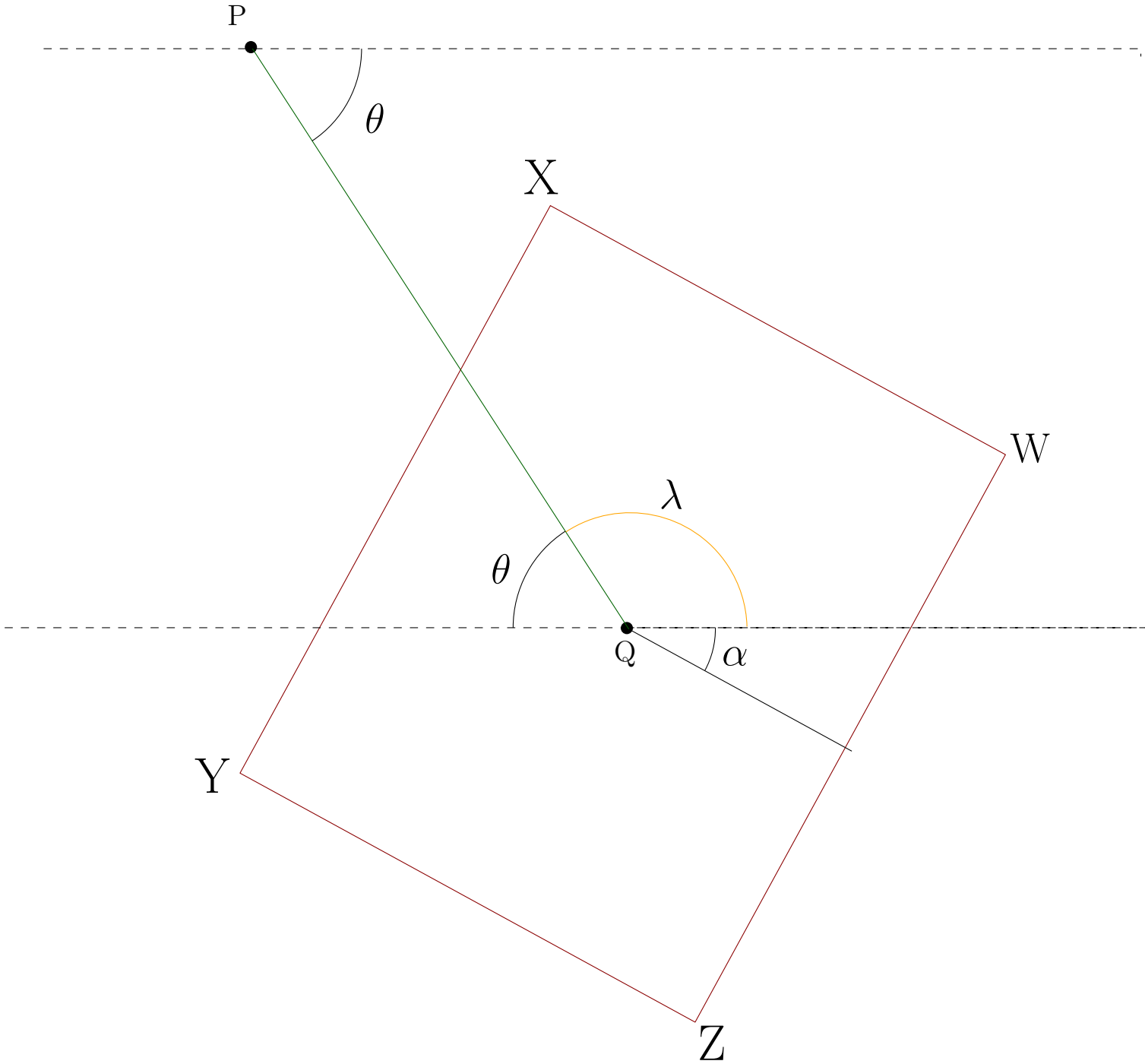
DotProduct :

```
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)

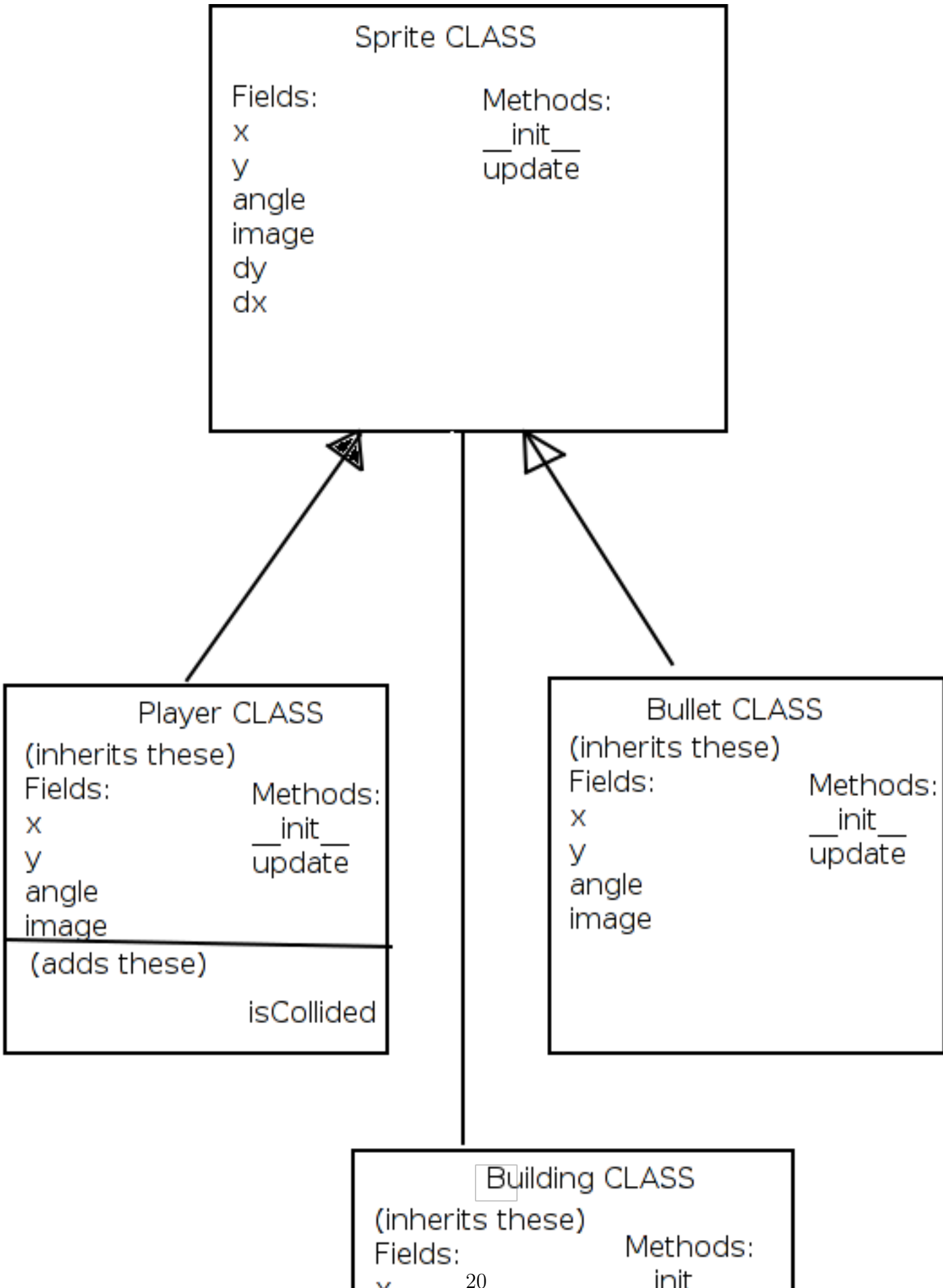
Output arccos(90 - CosTheta)

```

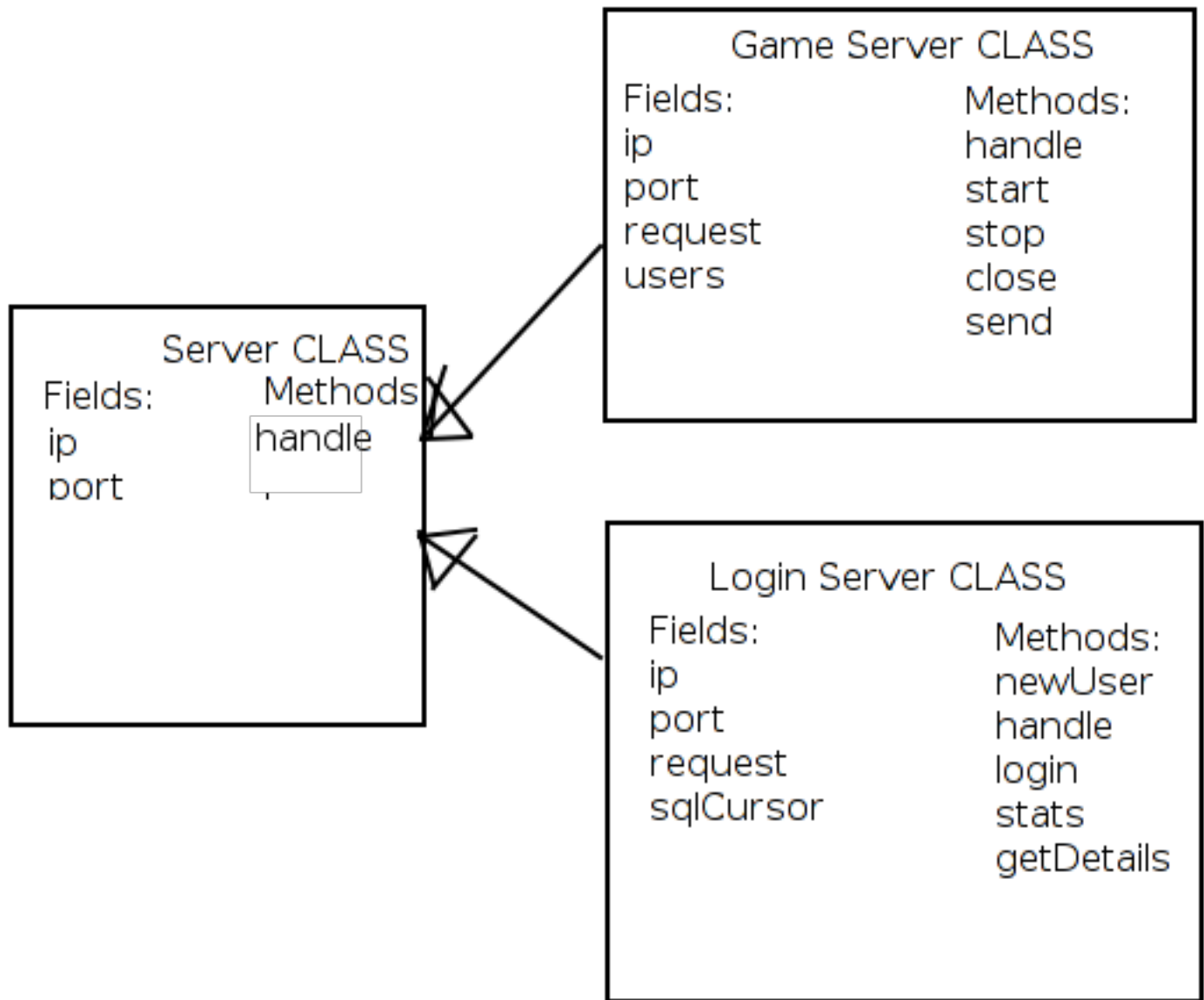
2.7 Class definitions

In this section I will use diagrams to explain my planned class structures.

2.7.1 Sprite Classes



2.7.2 Server Classes



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

Armoured Warfare Simulator 2014

[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

Upgrade

Server Address:Port: localhost:9999

Roll out!

Clear button to
start the game.
easy to find

Default value to
let the user know the
format of input

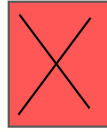
2.10 Description of measures planned for security and integrity of data

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 erroneous data to be modified, in the case of the HOST:PORT error.

2.11 Description of measures planned for systems security

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 inaccessible 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 receive 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) |

3 System Maintenance Document

In this section I will overview how my system works, first generally, then per-algorithm.

3.1 System Overview

Armoured Warfare Simulator 2014 is an online multiplayer game that takes simple shooting mechanics and turns them into a strategic game of angles. It is run over a peer-to-peer network with anyone able to host a server and start playing with their accounts which are handled on a unified login server, keeping track of progress on each account.

I have designed the system to only have trivial tasks for the end-user, as this allows anyone to use the system. The only non-trivial task in my program is the “updateAllSettings.py” script for moving login server location. The client/server settings can be updated by entering the new login server IP address (string) and the port (integer) into the terminal.

The data in my system is stored in 2 sqlite3 databases, one is clientside and holds the base tank statistics and the other is serverside, holding all user information and progress made.

3.2 Sample algorithms

3.2.1 Angle of Intersection

```
def getAngleOfIntersection(vecA, vecB):  
    """a dot b / mag(a) mag(b)"""  
    try:  
        # Set numerator and denom  
        num = vecA.dotProduct(vecB)  
        denom = vecA.getMagnitude() * vecB.getMagnitude()  
        # Angle is  $\cos^{-1}(A \cdot B / |A||B|)$   
        ang = math.degrees(math.acos(num / denom))
```

```

        return ang
    except Exception as ex:
        print "Exception in getangleofintersection: " + str(ex)

```

Line by line explanation of the algorithm 1. Define algorithm getAngleOfIntersection, taking parameters vector A and vector B

2. The documentation for the algorithm, describes the use.
3. Try the below code and check for errors
4. Set num to be the dot product of vector A and vector B (See 2.1)
5. Set denom to be the magnitude of A multiplied by the magnitude of B
6. Comment explaining that the angle is the inverse cosine of num/denom
7. Calculate the angle of intersection
8. Return the result.

3.2.2 Login Procedure

```

def login(self , dat):
    """Compare the users credentials to the database and login if they match"""
    #Check if these credentials are in the database
    if len(LogServer.cur.execute("""SELECT UserInfo.UserName FROM UserInfo
INNER JOIN UserPass ON UserInfo.UserId=UserPass.UserId
WHERE UserInfo.Username=? AND UserPass.HashPass=?""", dat).fetchall()) > 0:
        userId = LogServer.cur.execute("SELECT UserId FROM UserInfo
                                         WHERE Use

        print "ID LOGGING IN: " + str(userId)
        a = (LogServer.cur.execute("""SELECT UserInfo.Username, UserProgress.*, UserO
INNER JOIN UserProgress, UserOwned
                                ON UserInfo.UserId = UserProgress.UserId
AND UserInfo.UserId = UserOwned.UserId
                                WHERE UserInfo.Username = ?;""", [dat[0]]

        self.request.sendall(pickle.dumps(a))
    else:
        print "Login failed"
        self.request.sendall(pickle.dumps("LoginFailure"))

```

1. Define a new function, login that takes argument dat
2. Documentation string
3. Comment
4. Query the database for matching credentials
5. Set userId as the database stored unique identifier associated to the username
6. Print a message logging that a login attempt has been succesful
7. Set a as the database stored info for the user's progress
8. Send the user progress back to the client
9. If there are no matching credentials:
10. Log that a failed login attempt has been made
11. Tell the client that login was unsuccessful.

3.2.3 Get rebound angle

This is an excerpt from get() in server.py

```

#If bullets don't pen, they should rebound
if len(req[4]) > 0:
    for bid in req[4]:
        id = bid[0]
        angleOfImpact = bid[1]
        angleOfNormal = bid[2]
        angleOfBullet = bid[11]
        anglePointingAway=(angleOfBullet + 180) % 360
        angleToNormal = math.fabs((angleOfNormal - anglePointingAway) % 360)
        newAngle = (anglePointingAway + (2 * angleToNormal)) % 360
        for b in TankServer.Bullets:
            if b.bulletID == id:
                toEdit = b
                toEdit.angle = newAngle

```

1. Comment for documentation
2. If req[4] (list of bullets the client has said should rebound) has elements:
3. the bullet identifier is stored to ID
4. The client calculated angle of impact is stored
5. The client's normal angle is stored
6. The angle the bullet is pointing is stored
7. For calculations, we store the opposite angle of the bullet, i.e. it's angle - 180
8. We calculate the absolute angle from the normal to the angle of the bullet
9. We use this calculated difference to find the new angle of the rebounded bullet
10. Iterate through the bullet array
11. If we've found the right one, do something
12. Save the needed bullet entity
13. Change the angle of the bullet.

3.2.4 Map Generation

```

def generateMap(width, height):
    """Generate an array that represent a map"""
    #Split the screen into 50px blocks
    toplefts_x = [x for x in range(0, width + 1, 100)]
    toplefts_y = [x for x in range(150, (height + 1) - 150, 100)]
    numX = len(toplefts_x)
    numY = len(toplefts_y)
    #Initialise map array
    Map = []
    #We want to have some buildings here and there
    #But we cannot put them over the tanks
    #Tanks spawn in the top and bottom, y=100 and y=700
    #So we cut out ranges 0>y>150 and height-150 > y > height
    #We now want a small chance for a building to gen, and perhaps a tiny chance of
    possibleBlocks = len(toplefts_x)
    for i in range(possibleBlocks):

```

```

#Random number generator picks the fill for the block
toSpawn = random.randint(0, 60)
#Add a single building
if toSpawn > 50:
    xPos = i % numX
    yPos = i % numY
    Map.append([xPos, yPos, 1])
#Add a 2x2
elif toSpawn == 30:
    xPos = i % numX
    yPos = i % numY
    Map.append([xPos, yPos, 2])

return Map

```

This procedure is pretty well documented, so I'll spare you the line-by-line explanation. I will expand on these lines though:

```

xPos = i % numX
yPos = i % numY

```

These find the x and y block positions from the current iteration in order to allow the client recreate the map after transmission.

3.2.5 Modified Game Rendering Engine

In a fast-paced game, pygame with a livewires wrapper just doesn't cut it; It's not meant for fast rendering and hence I had to make it comply. After digging around the source I found the rendering method. These are the changes I made.

```

while not self._exit:
    self._wait_frame()
    for object in self._objects:
        object._erase()
    self._display.fill(color.white)
    self._display.blit(self._background, (0, 0))
    # Take a copy of the _objects list as it may get changed in place.
    for object in self._objects[:]:
        if object._tickable:
            object._tick()
    self.tick()
    for object in self._objects:
        object._draw()

```

The important lines here are those that reference `self._display`. I modified this engine to fill the screen with colour between ticks to stop the sprites flickering. The first line fills the screen and the second draws the current background on the screen.

3.2.6 Network Communication Method

```

def send(self, message):
    """Send a specified message to the server (pickled)"""

```

```

#Dial the 9th circle of hell and ask for lucifer to process our request
self.toSend = pickle.dumps(message)
try:
    self.sock.sendall(self.toSend)
    self.recieved = pickle.loads(self.sock.recv(2048))
    if self.last != self.recieved:
        self.last = self.recieved
    else:
        self.lastRetries += 1
        if self.lastRetries == 1000:
            raise HostDisconnectedException()
    self.retries = 0
except Exception as e:
    del self.sock
    self.sock = socket.create_connection((self.ip, self.port))
    self.retries += 1
    if self.retries == 5:
        raise HostDisconnectedException()

```

1. Define a new method send that takes argument message
2. Documentation string
3. Nothing to see here, moving on.
4. make a pickle (binary format) object from the specified message.
5. Try to do the following:
6. Send the pickle object over the network
7. Try to decode the server's response which is read as a maximum of 2048 bytes
8. Check if the recieved data is different from the last recieved
9. If it is different, store this new data for furture reference
10. If it isn't different, do the following
11. Increment "retries"
12. After a set number of retries...
13. Timeout.
14. If the data is different, reset retries
15. If there is an error:
16. Delete the connection object
17. Try to reconnect
18. Add 1 to retries
19. If 5 connections fail:
20. Timeout.

3.2.7 Effective Armour Calculations

```

def doesPenetrate(self, angle, bullet):
    """Returns true if the bullet has enough penetration, false otherwise"""
    #This is the critical angle at which any bullet will auto-bounce
    if angle < 50 or angle > 180:
        return False
    #Calculate effective armour via trigonometry.

```

```

    armourValue = self.client.armour
    effectiveArmour = armourValue / math.sin(math.radians(angle))
    #Check if the bullet has enough penetration and return
    if bullet.penetration > effectiveArmour:
        self.damageDone.append([bullet.damage, bullet.ownerId])
        return True
    else:
        return False

```

Again, this function is pretty well commented, but I shall expand.

1. Define a new function, doesPenetrate that takes arguments angle and a bullet object
2. Documentation string
3. Comment
4. Check whether the bullet is at an autobounce angle
5. If it is at autobounce, say that the bullet does not penetrate
6. If it is not autobounce, retrieve the client's armour thickness
7. Find the actual armour thickness at the specified angle
8. Check if the bullet is able to penetrate
9. If it does, put it in an array to let the server know, and say "yes, it penetrates"
10. If not, return false

3.3 Procedure and Variable lists

In this section I will list ever function and variable used throughout my project.

3.3.1 Clientside procedure list

| Procedure | Description |
|-----------------------------|---|
| Vector.update | Update the vector with new co-ordinates |
| Vector.useAngle | Create a new vector using a magnitude and angle |
| Vector.add | Return the resultant vector of this vector added to another |
| Vector.getMagnitude | Returns the absolute length of the vector |
| Vector.getDx | Gets the X length of the vector |
| Vector.getDy | Gets the Y length of the vector |
| Vector.dotProduct | Returns the scalar product of this and another vector |
| Vector.collides | Finds whether this vector collides with another |
| getAngleOfIntersection | Returns the acute angle of vector intersection |
| getPoints | Returns a list of pixels along a given vector |
| intersect | Checks whether 2 vectors collide or not |
| getConfiguration | Reads the config file and returns the option required |
| Login.createAccount | Asks the login server to add the account |
| Login.process | Processes the server response |
| Login.readConfig | Reads the configuration file for ip address and port |
| Login.sendCredentials | Send the login credentials to the server |
| Login.clearBoxes | Clear the username/password boxes |
| Login.startCreate | Start the account creation window |
| Login.loginComplete | Server has logged us in, proceed to next screen |
| startLogin | The main method for starting the login GUI |
| YesNo | Brings up a simple yes or no dialog |
| Info | Brings up an information box |
| Warn | Brings up a warning box |
| netComms.send | Send a specified message to the server (pickled) |
| netComms.close | Close the active connection |
| BuyTank.populate | Put the tank names into the list |
| BuyTank.changeTankPrice | Show the correct price of the selected tank |
| BuyTank.buyTank | Button press event to send the command to buy the tank |
| BuyTank.cancel | Close the GUI |
| UpgradeTank.populateBoxes | Put the selected tank stats into the boxes |
| UpgradeTank.upHP | Button press event to upgrade HP |
| UpgradeTank.upDam | Button press event to upgrade damage |
| UpgradeTank.upArm | Button press event to upgrade armour |
| UpgradeTank.upPen | Button press event to upgrade penetration |
| UpgradeTank.upHTr | Button press event to upgrade hull traverse |
| UpgradeTank.upTTr | Button press event to upgrade turret traverse |
| UpgradeTank.upRel | Button press event to upgrade reload |
| UpgradeTank.upSp | Button press event to upgrade speed |
| UpgradeTank.convertToString | Convert a list of stats into a string |
| UpgradeTank.getNewStats | Put the upgraded stats into a server-readable format |
| UpgradeTank.confirmEdit | Ask the user if they are really sure they want to upgrade |
| UpgradeTank.cancelEdit | Close the GUI without changing anything |
| UpgradeTank.toInt | Convert a list of floats to ints |
| SelectATank.battleThread | Set up the client to launch the game engine and then run the game |
| SelectATank.setHost | Take the user input of the host and store it |
| SelectATank.getStats | Get the users stats for that specific tank |
| SelectATank.doStats | Update the text box with the tanks stats |

3.3.2 Serverside procedure list

| Procedure | Description |
|--|---|
| getDeltaT | Returns the difference between 2 time.time instances |
| getPoints | Returns a list of pixels on the line joining (x1,y1) to (x2, y2) |
| LoginServer.handle | Handle the client request |
| LoginServer.buy | add a tank to the users account |
| LoginServer.allXP | Returns the amount of XP the user has earned on each tank |
| LoginServer.costs | Return the amount that each tank costs |
| LoginServer.login | Compare the users credentials to the database and login if they match |
| LoginServer.create | Add a new user to the database |
| LoginServer.get | Get the stats on the tank requested by the user |
| LoginServer.convertToString | Self-explanatory, converts a list to a string |
| LoginServer.createNewAccount | insert all data into the database |
| LoginServer.xp | Gets the users XP |
| LoginServer.makeListSane | Make the list how I need it - all strings |
| LoginServer.openCloseConn | Close and reopen the database connection |
| LoginServer.update | Update all of the data in the database |
| LoginServer.owned | Return the tanks owned by the user |
| start | Start the login server |
| main | Run game server from another file |
| superMain | Run game server without checking the login server |
| generateMap | Generate an array that represent a map |
| Player.returnValues | Return player values in a list |
| Player.set | Set all the players data |
| Bullet.update | Update the bullet based on a time difference |
| GameServer.ServerUpdatingThread | A thread to keep the server up-to-date |
| GameServer.finish | End the request process |
| GameServer.getVictor | Count the players and see who won the game |
| GameServer.stringRequest | Takes a string and outputs accordingly |
| GameServer.listRequest | Redirect method for requests in the form of a list |
| GameServer.isEndOfGame | Looks at all players and decides if the game is over |
| GameServer.convertToList | This will take Player objects and shove the x,y,angle,turret angle data into a list |
| GameServer.doHandshake | Add the new player to arrays and get going |
| GameServer.countdown | Create a 30 second timer at the start of the game |
| GameServer.convertToListHandshake | Initial return value |
| GameServer.get | Acts as a 'getter', returns every other player's information and send it back |
| GameServer.getBuildingRanges | Gets the pixel ranges of the buildings for collision testing |
| GameServer.isCollidedWithMap | Checks if the bullet is collided with any buildings |
| GameServer.endgame | Closes the server |
| ServerSetupForm.stopEvent | In case the user wants to cancel the startup at the last moment |
| ServerSetupForm.loginThread | A thread to start the login server |
| ServerSetupForm.getOperatingSystem | Gets the OS of the server machine to bind to IP address |
| ServerSetupForm.changeInterface | Gets the IP of interface (i.e. wlan, ethernet etc) |
| ServerSetupForm.startServer | Button press event to start the server |
| ServerSetupForm.stopServer | Button press event to stop the server |
| ServerSetupForm.beginTheSatanHailing | Begin to handle requests |
| ServerSetupForm.watchtheServerIntently | A thread to say if the server should stop |
| ServerSetupForm.updateTheBulletsThread | A thread that will update all bullets every 0.05 seconds |
| ServerSetupForm.startServerThread | A thread to handle the server |

3.3.3 Client-side Variable list

| Variable name | Description | Main Data Type |
|----------------|---------------------------------------|----------------------|
| message | What to send to the server | Pickle object |
| x | A sprite's x position | int |
| y | A sprite's y position | int |
| x1 | One end of a vector | int |
| y1 | One end of a vector | int |
| x2 | The other end of a vector | int |
| y2 | The other end of a vector | int |
| angle | The angle of a vector | float |
| myVectors | The vectors surrounding an image | List |
| Parent | The parent window for the GUI | Form |
| ipAddr | The IP address of the server | string |
| port | The port of the server | int |
| conn | The connection to the server | Connection object |
| toSend | The message to pass to netComms | String |
| recieved | The response from the server | Pickle object/String |
| alltanks | Every tank in game | List |
| xp | The amount of XP on a specific tank | int |
| owner | Who owns the opened form | Form |
| hp | The HP of the tank | int |
| damage | The damage of the tank | int |
| penetration | The penetration of the tanks gun | int |
| Reload | The time for the gun to reload | float |
| Armour | The amount of armour the tank has | int |
| HullTraverse | How fast the tank turns | float |
| TurretTraverse | How fast the turret turns | float |
| Speed | How fast the tank goes | float |
| owned | The tanks the user owns | list |
| username | The username of the current player | string |
| team | The team the user is fighting for | int |
| userTag | The username floating above the tank | text object |
| nametag | The tank name floating above the tank | text object |
| maxHP | The maximum health of the tank | int |
| state | Is the tank damaged? | int |
| image | The tank's sprite | image |
| ownerID | Who fired the bullet? | int |
| bulletID | Unique ID for the bullet | int |
| damage | how much damage will the bullet do? | int |
| ded | Do we update the bullet? | Boolean |

3.3.4 Serverside variable list

| Variable name | Description | Main Data Type |
|-----------------|---|----------------|
| message | What to send to the client | pickle object |
| x | The X position of an object | int |
| y | The Y position of an object | int |
| name | The name of the user's tank | String |
| username | The user's username | String |
| id | The user's ID | int |
| hp | The HP of the tank | int |
| isOnWinning | Is the user on the winning team? | Boolean |
| xpGained | The amount of XP earnt from the game | int |
| damage | The amount of damage done by the user | int |
| ownerID | The ID of the bullet owner | int |
| kills | The number of kills by the user | int |
| angle | The angle of the user's tank | int |
| turret_angle | The angle of the user's turret | int |
| ded | Is the bullet "alive"? | boolean |
| penetration | The level of penetration the bullet has | int |
| cur | the database cursor | cursor object |
| data | The data to push to the database | list |
| newID | The ID of a new user account | int |
| countdownThread | A thread to keep track of countdown | Thread |
| opSys | The operating system the server is running on | String |
| toClose | Shall we close the server? | Boolean |

3.4 Annotated Samples of GUI & System Outputs

3.4.1 Tank selection screen

This screen is for selecting a tank and giving server information, it also shows the stats for the currently selected tank. When the "TankBox" is clicked, it will give a list of available tanks. The "Upgrade Tank" and "Buy a tank" open the respective screens.

The screenshot shows a window titled "Armoured Warfare" with a standard Windows-style title bar (minimize, maximize, close buttons). The interface is divided into several sections:

- Select a tank:** A dropdown menu currently displaying "PanzerIV" with a downward arrow button.
- Tank stats:** A text area containing the following statistics:
 - HP: 400
 - Damage (HP average): 100
 - Penetration (mm): 110
 - Reload (ticks): 200
 - Armour (mm): 40
 - Hull Traverse Speed: 1
 - Turret Traverse Speed:A vertical scrollbar is visible on the right side of the text area.
- Server Address:Port:** A text box containing "localhost:9999".
- Buttons:** At the bottom, there are three buttons: "Roll out!" on the left, and "Upgrade tank" and "Buy a tank" side-by-side on the right.

3.4.2 Tank upgrade screen

This screen on the clientside allows for the upgrading of tank stats. The text boxes are read-only. If "Upgrade" is clicked on any row, the relevant stat will update, but nothing will be sent to the server until "Confirm" is clicked. If "Cancel" is clicked, the box will close to leave the main tank selection screen.

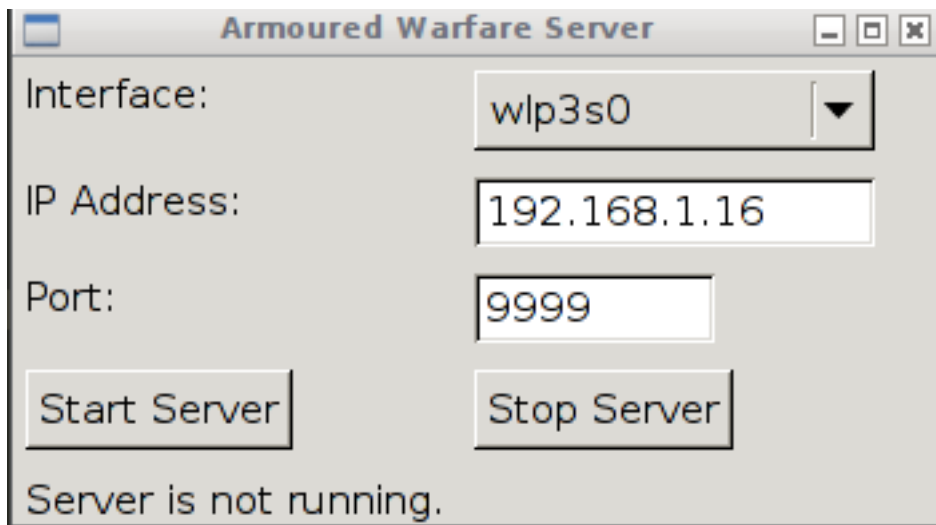
Upgrade A Tank

Upgrading Tank: PanzerIV XP to spend: 990

| | | |
|------------------|----------------------------------|--|
| Hit points: | <input type="text" value="400"/> | <input type="button" value="Upgrade"/> |
| Damage: | <input type="text" value="100"/> | <input type="button" value="Upgrade"/> |
| Penetration: | <input type="text" value="110"/> | <input type="button" value="Upgrade"/> |
| Reload: | <input type="text" value="200"/> | <input type="button" value="Upgrade"/> |
| Armour: | <input type="text" value="40"/> | <input type="button" value="Upgrade"/> |
| Hull Traverse: | <input type="text" value="6"/> | <input type="button" value="Upgrade"/> |
| Turret Traverse: | <input type="text" value="1"/> | <input type="button" value="Upgrade"/> |
| Speed: | <input type="text" value="1"/> | <input type="button" value="Upgrade"/> |

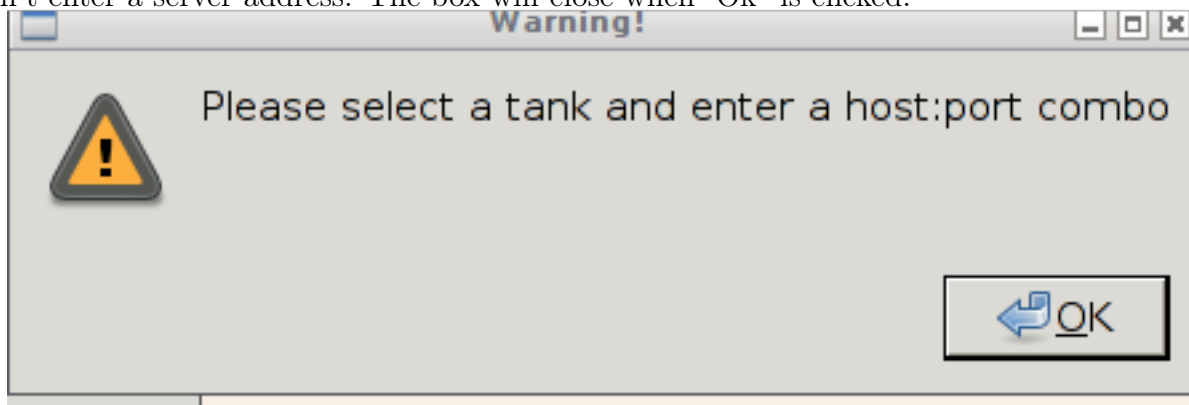
3.4.3 Server setup screen

This screen is on the serverside, it allows for the server operator to select an interface and port on which to run the game server. If the InterfaceBox is clicked, a list of currently attached interfaces (Wlan, Ethernet, etc) appear and the user may pick. The port box allows for any free port to be entered. The server will start when “Start Server is clicked”.



3.4.4 General Error Message

This box appears when you attempt to launch the game without either a. Selecting a tank to use or b. Don't enter a server address. The box will close when "Ok" is clicked.



3.4.5 Main Game Screen

This is the main screen of the game. It is controlled using the WASD keys for hull traverse and the arrow keys to traverse the turret. The username and tank name along with the tank HP are displayed above each respective sprite, and the reload timer is situated in the top left of the screen.

Reload in: 38

testaccount
Crusader 300

floatingghost
PanzerIV 325



4 Testing

4.1 Test Data

4.1.1 Login Server Test

| Username | Password | Reason for Choice |
|-------------|-----------|----------------------------------|
| testaccount | password | Normal Data |
| testaccount | pasword | Erroneous Data |
| testaccount | [No data] | Erroneous Data (Incorrect login) |
| [No data] | password | Erroneous Data (Incorrect login) |

4.1.2 Tank Selection Screen

| Tank Selection | Reason For Choice |
|----------------|-------------------|
| Panzer IV | Normal Data |
| M4 Sherman | Normal Data |
| [No selection] | Erroneous Data |

4.1.3 Tank Purchase Screen

| Tank To Buy | XP to spend | Reason For Choice |
|-------------|-------------|--------------------------------|
| Maus | 5000 | Normal Data |
| Maus | 100 | Erroneous Data (Not enough XP) |
| Maus | 300 | Boundary Data (Exact price) |

4.1.4 Join Game Screen

| IP:Port input | Reason for Choice |
|----------------|------------------------------------|
| localhost:9999 | Normal Data |
| localhost:9998 | Erroneous Data (no server on port) |
| [No data] | Erroneous Data |
| localhost | Erroneous Data (no port) |

4.1.5 Bullet Calculations

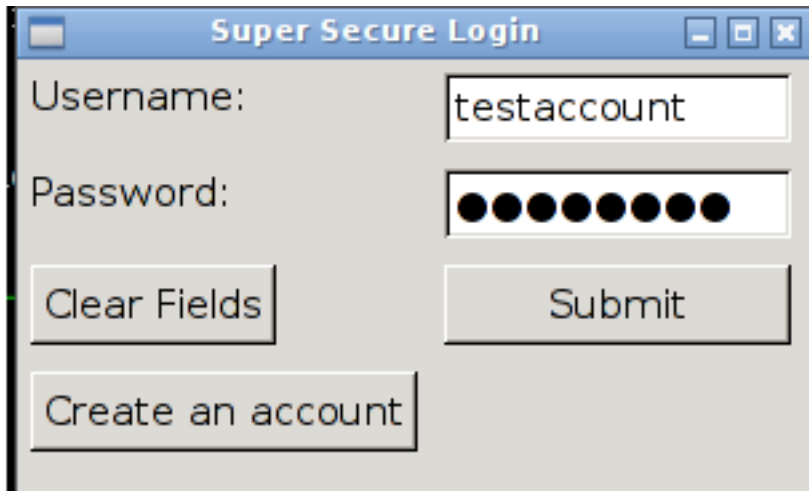
| Bullet Angle | Armour Penetration | Armour Thickness | Reason For Choices |
|--------------|--------------------|------------------|--------------------|
| 90 | 100 | 50 | Normal Penetration |
| 90 | 100 | 100 | Boundary Condition |
| 45 | 100 | 100 | Normal Ricochet |
| 45 | 150 | 100 | Normal Penetration |
| 10 | 200000 | 1 | Normal Ricochet |

4.2 Test Plan

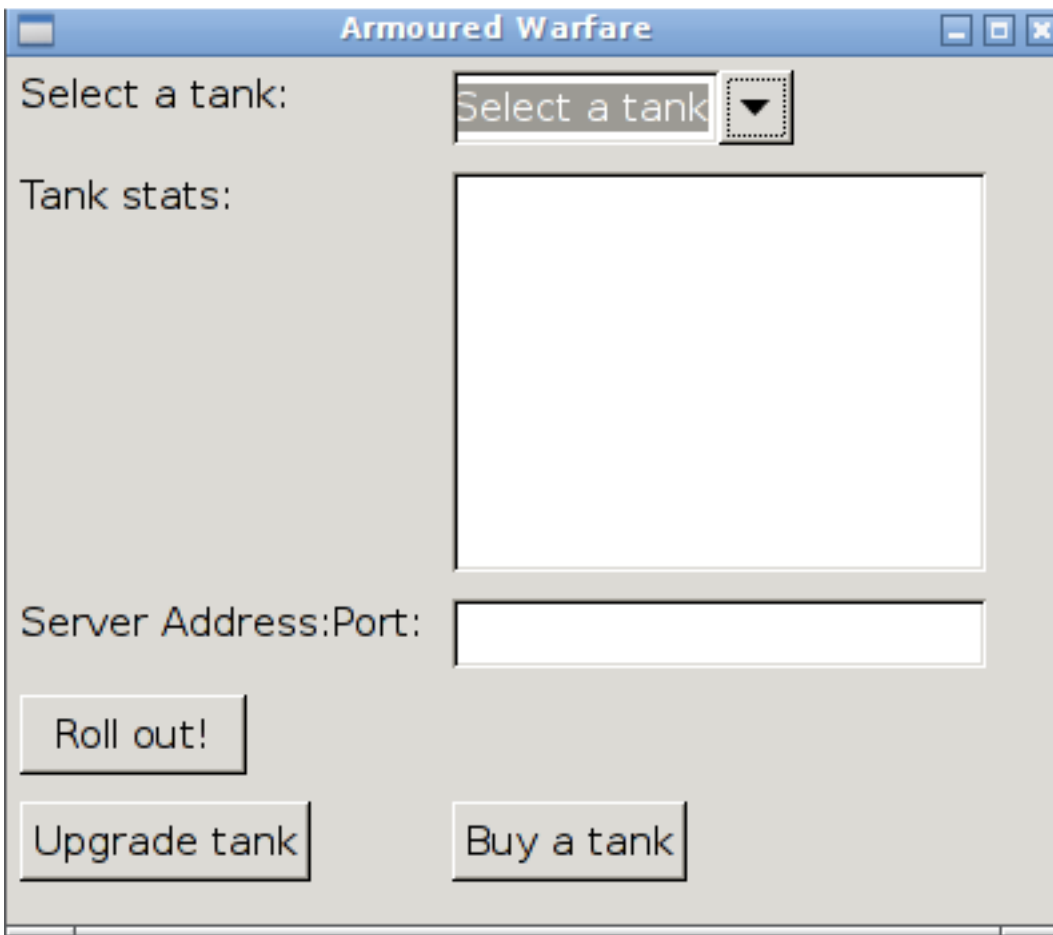
| Test | Purpose | Description | Test Data | Expected Result | Pass/Fail | Evidence Ref. |
|------|-------------------------|-------------------------|----------------------|------------------------|-----------|---------------|
| 1 | Test login server | Give normal data | testaccount/password | “login succesful” | Pass | |
| 2 | Test login server | Give Incorrect Data | testaccount/password | “login failed” | Pass | |
| 3 | Test Tank Selection | Select a tank | Panzer IV | Stats appear | Pass | |
| 4 | Test Tank Selection | No tank selected | [No data] | “Please select a tank” | Pass | |
| 5 | Test Tank purchase | Have enough XP | Maus/5000XP | “Tank purchased” | Pass | |
| 6 | Test Tank Purchase | Just enough XP | Maus/300XP | “Tank purchased” | Pass | |
| 7 | Test Tank Purchase | Not enough XP | Maus/0XP | “You need more XP” | Pass | |
| 8 | Test Server Connection | Give correct Host/Port | localhost:9999 | Connection | Pass | |
| 9 | Test Server Connection | Give incorrect port | localhost:9998 | Error window appears | Fail | |
| 10 | Test Server Connection | Give no information | [blank] | Error window appears | Pass | |
| 11 | Test Bullet Calculation | Give penetrating values | 100/90/50 | Penetration | Pass | |
| 12 | Test Bullet Calculation | Give boundary condition | 100/90/100 | Penetration | Fail | |
| 13 | Test Bullet Calculation | Give angled ricochet | 100/45/100 | Bounce | Pass | |
| 14 | Test Bullet Calculation | Give angled penetration | 150/45/100 | Penetration | Pass | |
| 15 | Test Bullet Calculation | Give autobounce | 20000/10/1 | Bounce | Pass | |

4.3 Evidence

4.3.1 Test 1

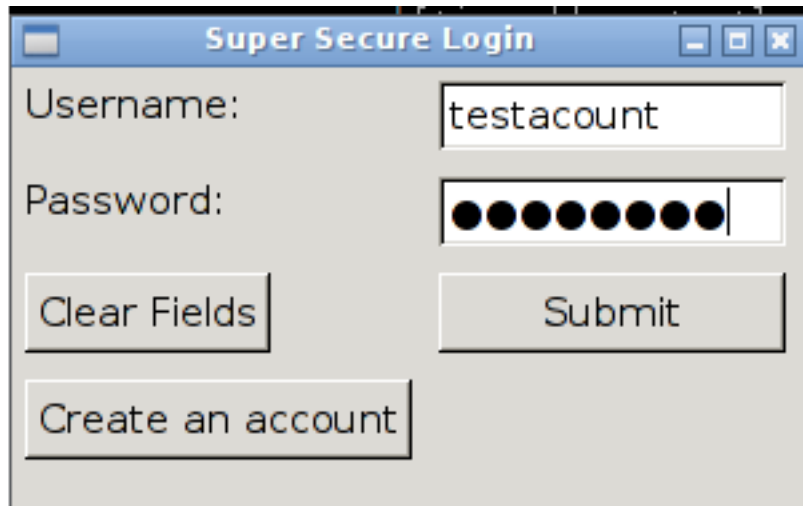


A screenshot of a web application window titled "Super Secure Login". The window has a blue title bar with standard minimize, maximize, and close buttons. The main content area is light gray. It contains two input fields: "Username:" with the text "testaccount" and "Password:" with ten black dots. Below the password field are two buttons: "Clear Fields" and "Submit". At the bottom left, there is a button labeled "Create an account".



A screenshot of a web application window titled "Armoured Warfare". The window has a blue title bar with standard minimize, maximize, and close buttons. The main content area is light gray. It contains a "Select a tank:" label next to a dropdown menu showing "Select a tank" with a downward arrow. Below this is a "Tank stats:" label next to a large empty rectangular box. Further down is a "Server Address:Port:" label next to an empty text input field. At the bottom, there are three buttons: "Roll out!", "Upgrade tank", and "Buy a tank".

4.3.2 Test 2



A screenshot of a web application window titled "Super Secure Login". It features a username field containing "testaccount" and a password field with ten black dots. Below the fields are three buttons: "Clear Fields", "Submit", and "Create an account".

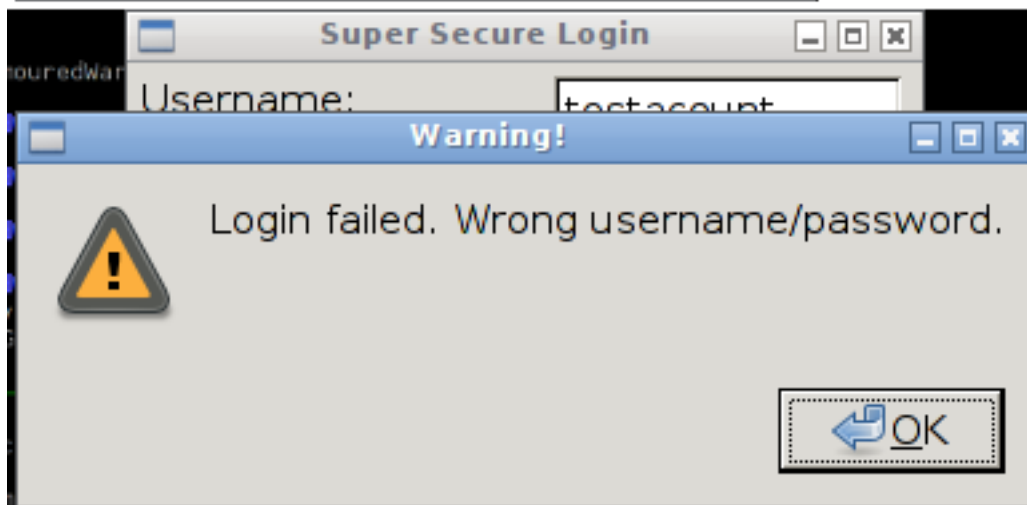
Super Secure Login

Username: testaccount

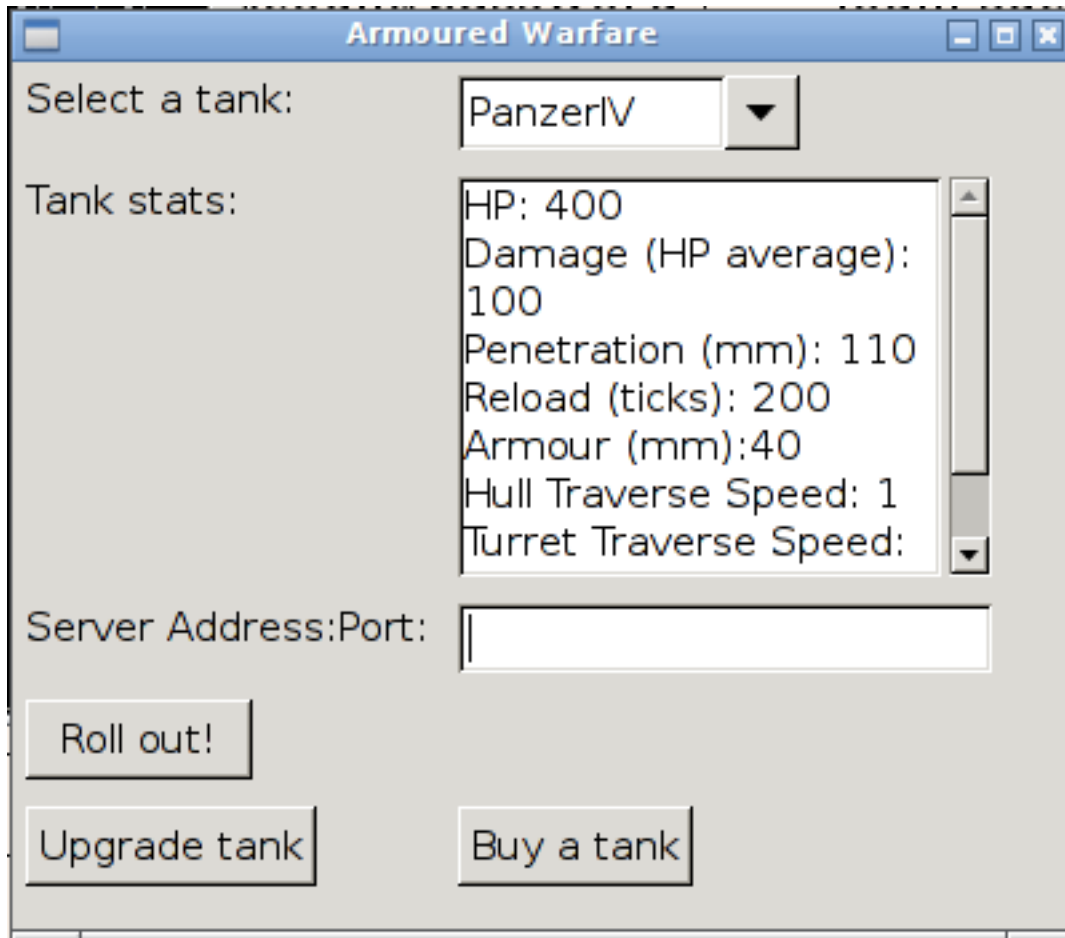
Password: ●●●●●●●●●●

Clear Fields Submit

Create an account



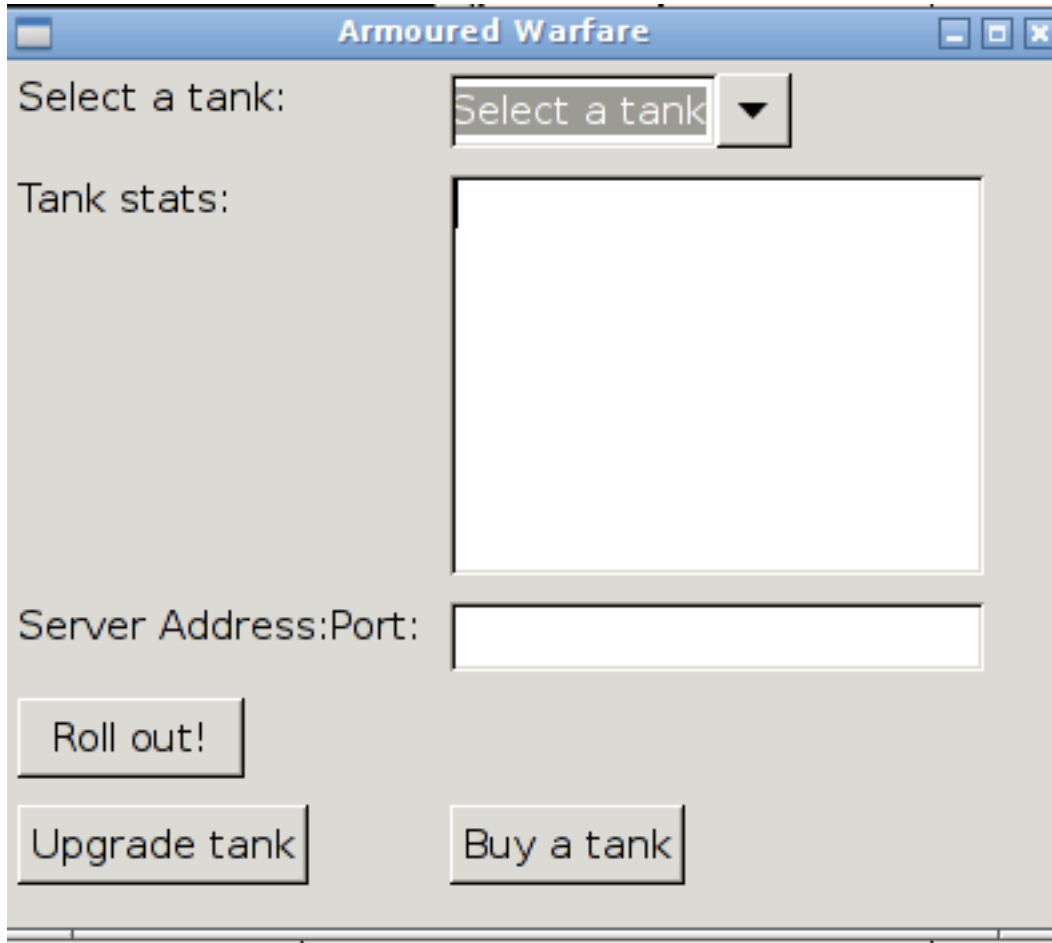
4.3.3 Test 3



The screenshot shows a window titled "Armoured Warfare" with a standard Windows-style title bar (minimize, maximize, close buttons). The window contains the following elements:

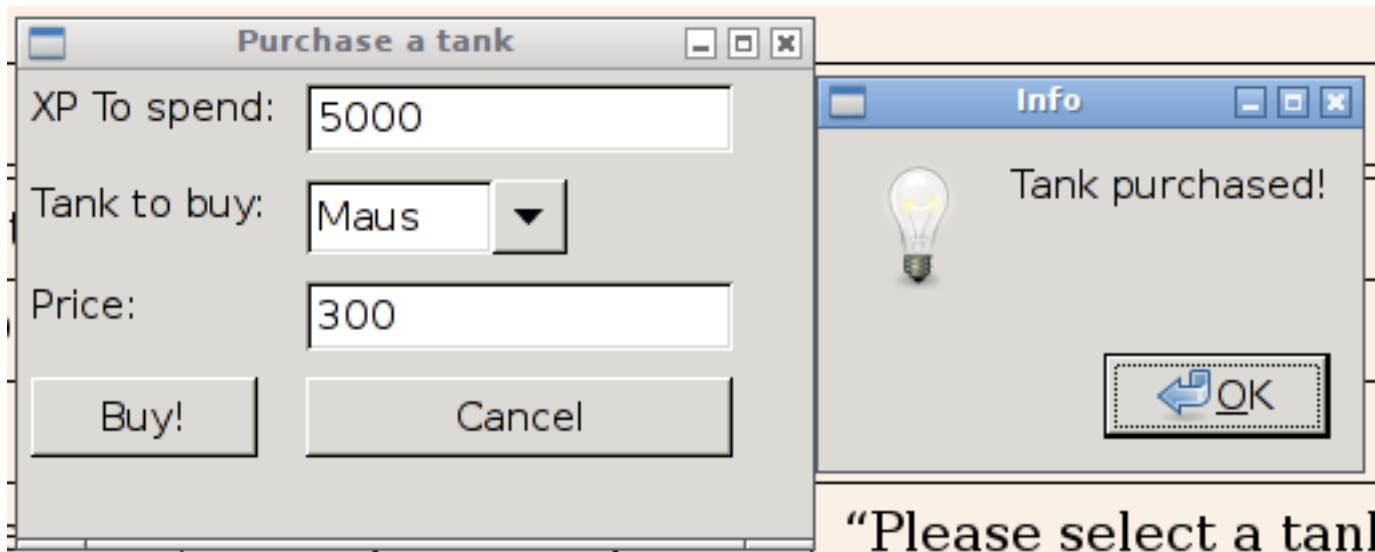
- Select a tank:** A label followed by a text box containing "PanzerIV" and a dropdown arrow button.
- Tank stats:** A label followed by a scrollable text area containing the following statistics:
 - HP: 400
 - Damage (HP average): 100
 - Penetration (mm): 110
 - Reload (ticks): 200
 - Armour (mm): 40
 - Hull Traverse Speed: 1
 - Turret Traverse Speed:
- Server Address:Port:** A label followed by an empty text input field.
- Roll out!** A button.
- Upgrade tank** and **Buy a tank** buttons.

4.3.4 Test 4



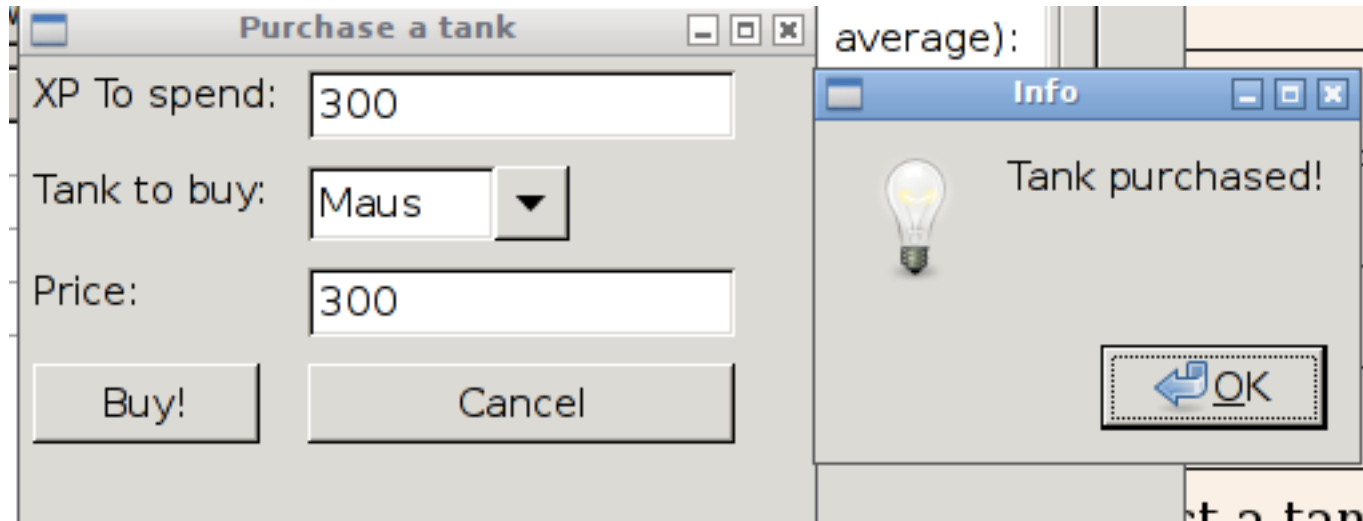
A screenshot of a Windows-style application window titled "Armoured Warfare". The window has a blue title bar with standard minimize, maximize, and close buttons. The main content area is light gray and contains the following elements: a label "Select a tank:" followed by a dropdown menu showing "Select a tank" with a downward arrow; a label "Tank stats:" followed by a large, empty rectangular box; a label "Server Address:Port:" followed by an empty text input field; a "Roll out!" button; an "Upgrade tank" button; and a "Buy a tank" button.

4.3.5 Test 5

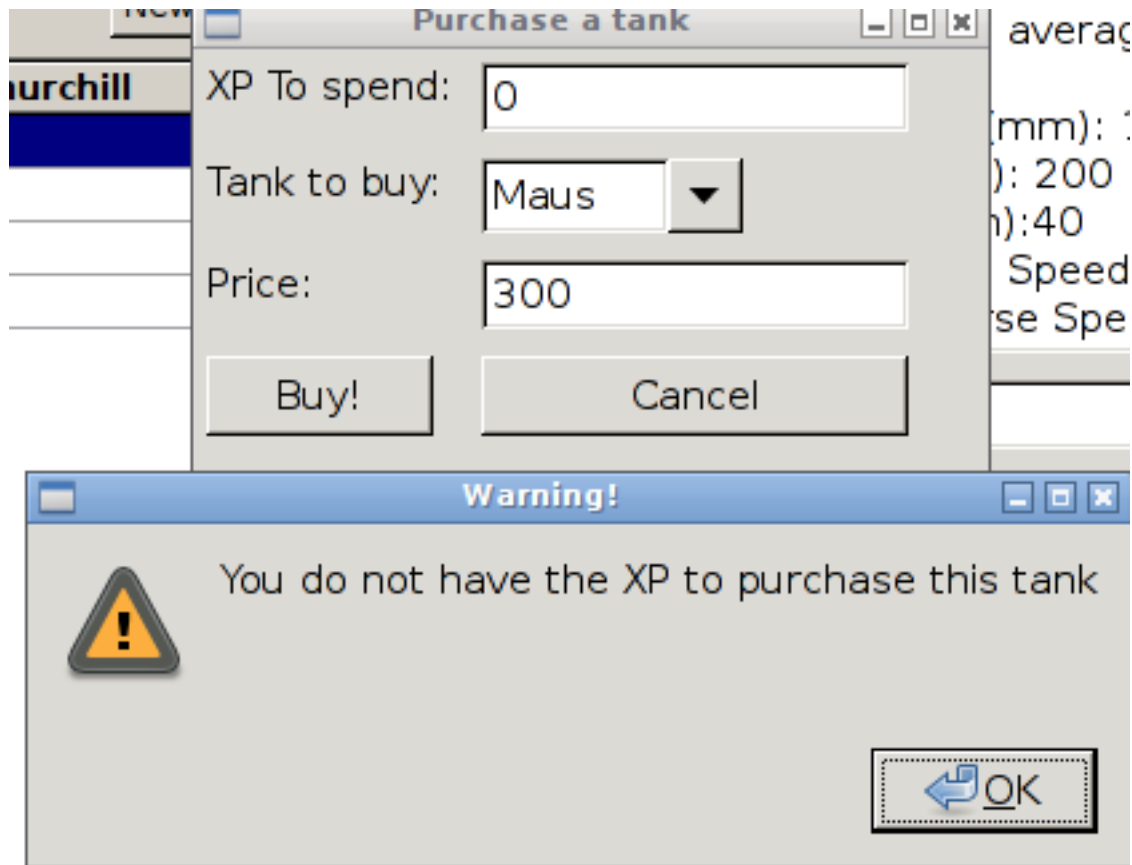


A screenshot showing two overlapping Windows-style application windows. The foreground window is titled "Purchase a tank" and contains: a label "XP To spend:" with a text input field containing "5000"; a label "Tank to buy:" with a dropdown menu showing "Maus"; a label "Price:" with a text input field containing "300"; a "Buy!" button; and a "Cancel" button. The background window is titled "Info" and contains: a lightbulb icon; the text "Tank purchased!"; and an "OK" button with a blue arrow icon. Below the "Info" window, the text "Please select a tank" is visible.

4.3.6 Test 6

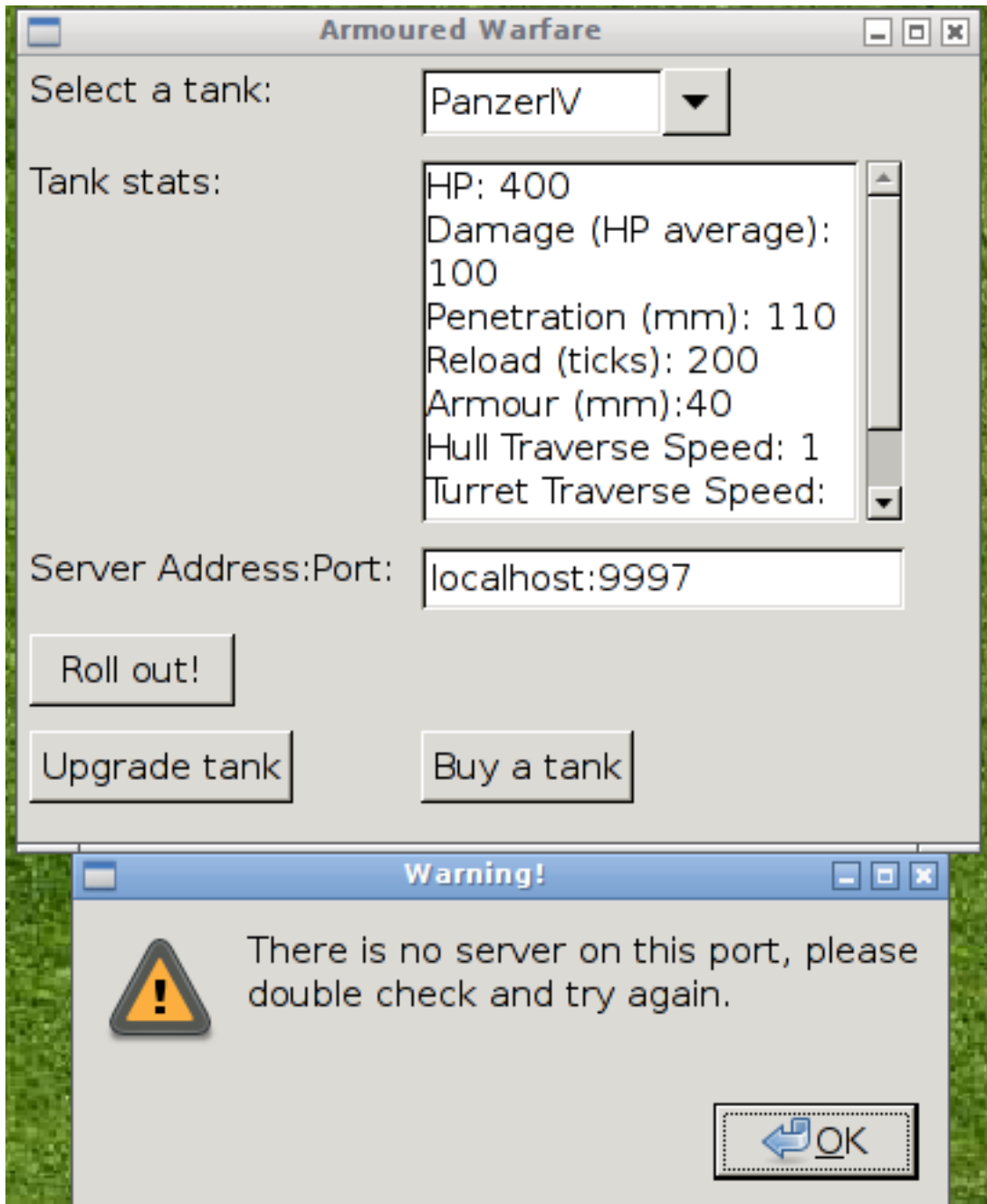


4.3.7 Test 7





4.3.9 Test 9



4.3.10 Test 10

The screenshot displays two windows from a Java Swing application titled "Armoured Warfare".

The main window, "Armoured Warfare", has a title bar with standard window controls. It contains the following elements:

- A label "Select a tank:" followed by a text box containing "PanzerIV" and a dropdown arrow.
- A label "Tank stats:" followed by a scrollable text area containing the following statistics:
 - HP: 400
 - Damage (HP average): 100
 - Penetration (mm): 110
 - Reload (ticks): 200
 - Armour (mm): 40
 - Hull Traverse Speed: 1
 - Turret Traverse Speed:
- A label "Server Address:Port:" followed by an empty text box.
- Three buttons: "Roll out!", "Upgrade tank", and "Buy a tank".

A secondary "Warning!" dialog box is open in the foreground. It has a blue title bar and a yellow warning triangle icon. The message inside reads: "Please select a tank and enter a host:port combo". An "OK" button with a blue arrow icon is located at the bottom right of the dialog.

4.3.11 Tests 11-15

```
~/ArmouredWarfareSimulator/Client(branch:master*) » python clientTests.py
test_penetration (__main__.MyTestCase) ...
Testing 100 pen at 90 degrees against 50mm of armour
Testing 100 pen at 90 degrees against 100mm of armour
Testing 100 pen at 45 degrees against 100mm of armour
Testing 150 pen at 45 degrees against 100mm of armour
Testing 20000 pen at 10 degrees against 1mm of armour
ok
-----
Ran 1 test in 1.003s

OK
```

4.4 Error Log

4.4.1 Incorrect port given

```
Error whilst init netcomms: [Errno 111] Connection refused
Traceback (most recent call last):
  File "/home/harry/ArmouredWarfareSimulator/Client/SelectATank.py", line
n goToBattle
    a = self.battleThread(instance)
  File "/home/harry/ArmouredWarfareSimulator/Client/SelectATank.py", line
n battleThread
    a = TankClient.mainGame(instance)
  File "/home/harry/ArmouredWarfareSimulator/Client/TankClient.py", line
mainGame
    fat_controller = GameController(stats, host, port, username)
  File "/home/harry/ArmouredWarfareSimulator/Client/TankClient.py", line
__init__
    self.setupGame(stats, host, port, username)
  File "/home/harry/ArmouredWarfareSimulator/Client/TankClient.py", line
setupGame
    self.connection = netComms.networkComms(host, int(port))
  File "/home/harry/ArmouredWarfareSimulator/Client/netComms.py", line 16
__init__
    raise NoConnectionException()
Errors.NoConnectionException
^C
```

This screenshot is the python output if you try and connect where there is no server.

This was an easy one to fix. I simply appended this to the game launch method:

```
except NoConnectionException:
    messages.Warn(self.parent, "There is no server on this port, please double
self.Show(True)
```

It works just fine.

4.4.2 Penetration Boundary Condition

```
~/ArmouredWarfareSimulator/Client(branch:master*) » python clientTests.py
test_penetration (__main__.MyTestCase) ...
Testing 100 pen at 90 degrees against 50mm of armour

Testing 100 pen at 90 degrees against 100mm of armour
FAIL

=====
FAIL: test_penetration (__main__.MyTestCase)
-----
Traceback (most recent call last):
```

Simple change of boolean conditions here.

In Client/TankClient.py, line 744 was the culprit.

```
if bullet.penetration > effectiveArmour:
```

was changed to become

```
if bullet.penetration >= effectiveArmour:
```

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/>