UNIVERSITY OF LEICESTER

Project Mobo

Christopher John Cola

Viva

CO3015 Computer Science Project

Overview

- Background
- Project Proposal and Scope
- Software and Network Architecture
- Client “Mobo” implementation
- Server “MoboServerWPF” implementation and synchronisation
- Messaging protocol example
- Final deliverable and conclusion

Background

- Video games have become increasingly ambitious in scope
- More complex in design but also their network features

Greater importance:

- Network and software architecture
- Way synchronisation is achieved (messaging, protocols)

Examples:

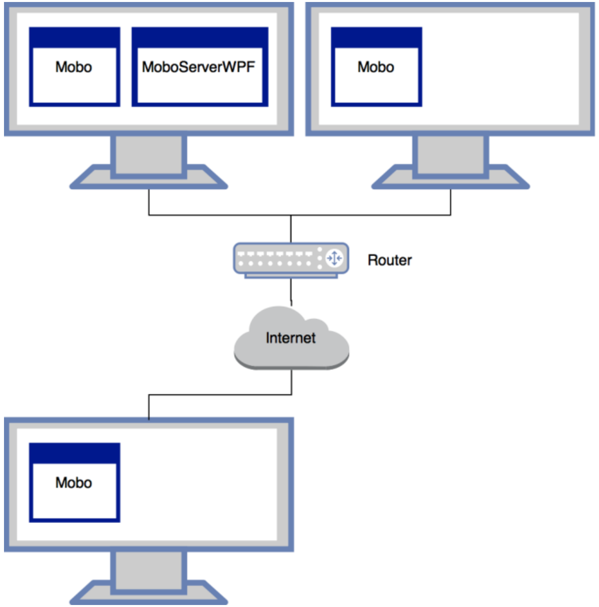
- Eve Online, The Divison, Battleborn, Overwatch

Project Proposal and Scope

- “Multiplayer XKobo clone”
- Client-server architecture
- Multiplayer enabled
- Synchronisation across clients
- Game logic programming
- GUI design and 2D sprite handling
- Procedural Station generation
- Emphasis on polished user experience

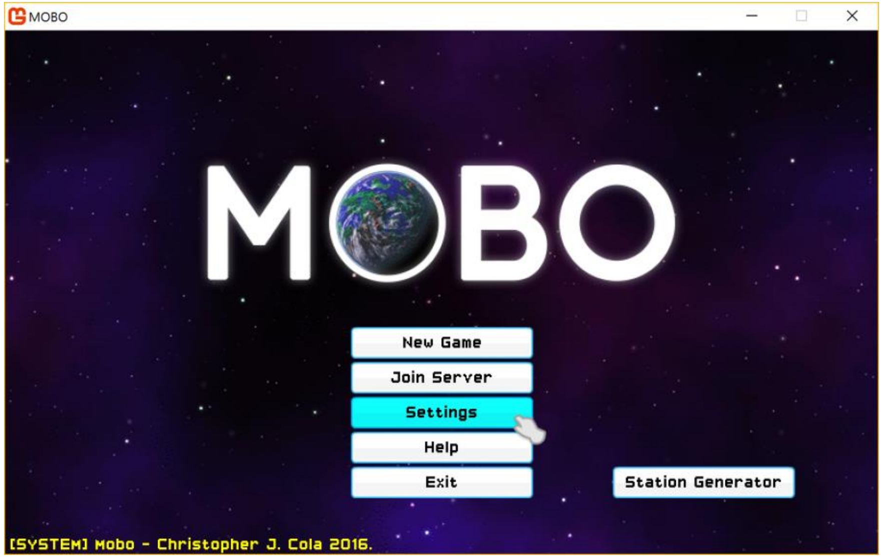
Software and Network Architecture

- Client-server
- Dedicated server – supports “Drop-in, Drop-out”.
- Non-authoritative – supports offline play
- Distributed server

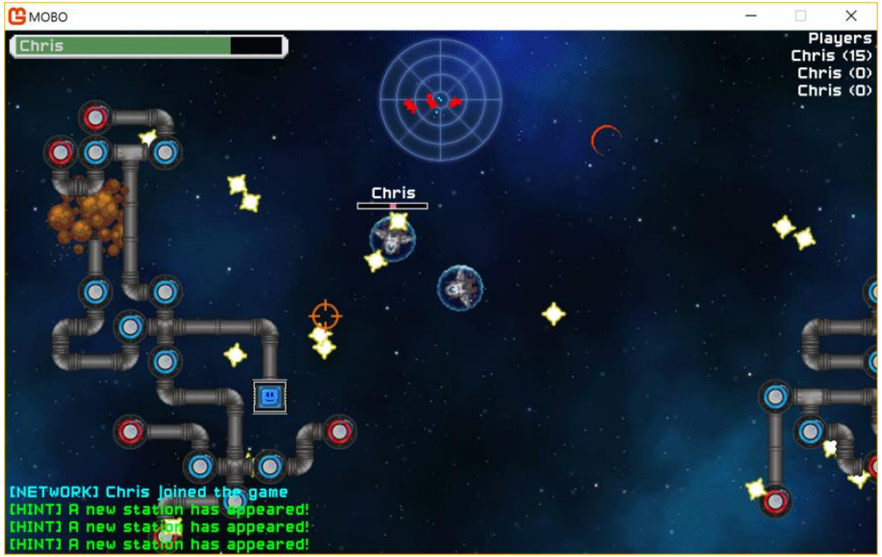


The Client - “Mobo”

- Developed using the Monogame/XNA framework
- GUI using sprites and 2D effects with emphasis on look and feel
- Contains the vast majority of the game logic (“thick client”)



[SYSTEM] Mobo - Christopher J. Cola 2016.



[NETWORK] Chris joined the game
[HINT] A new station has appeared!
[HINT] A new station has appeared!
[HINT] A new station has appeared!

Station Structure and Generation

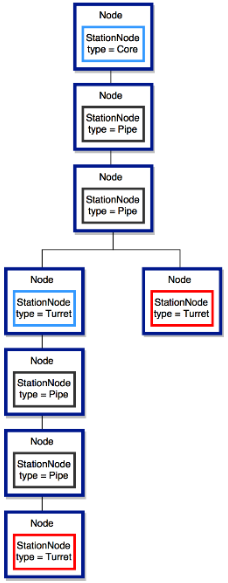
- Modelled as trees built in 2D space
- Always unique, procedurally generated using an algorithm,
- Collision enabled, destroyable, fire back at players

```
split_depth = 6;

max_depth = 15;

difficulty = 5;

max_branches = 2;
```



Node
StationNode
type = Core

Node
StationNode
type = Pipe

Node
StationNode
type = Pipe

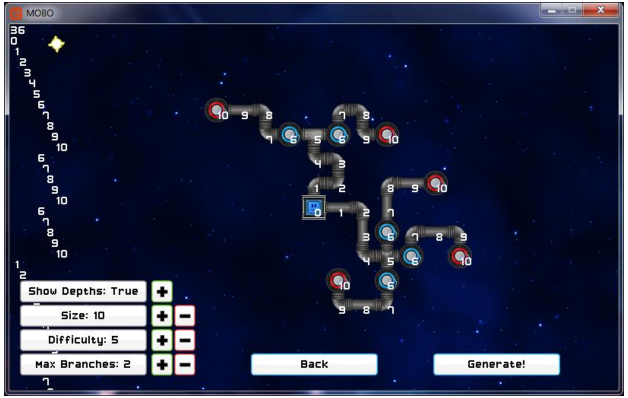
Node
StationNode
type = Turret

Node
StationNode
type = Turret

Node
StationNode
type = Pipe

Node
StationNode
type = Pipe

Node
StationNode
type = Turret



Show Depths: True

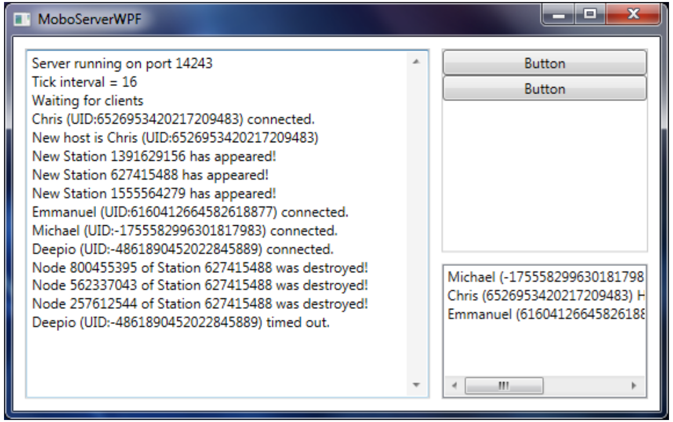
Size: 10

Difficulty: 5

Max Branches: 2

Server and Synchronisation

- Developed using Windows Presentation Foundation framework
- Messages and protocols using low-level Lidgren.Network
- Non-authoritative distributor
- Synchronisation fundamental to multiplayer gameplay
- Connections / disconnections / hosting
- Player position
- Projectiles
- Stations
- Health and score

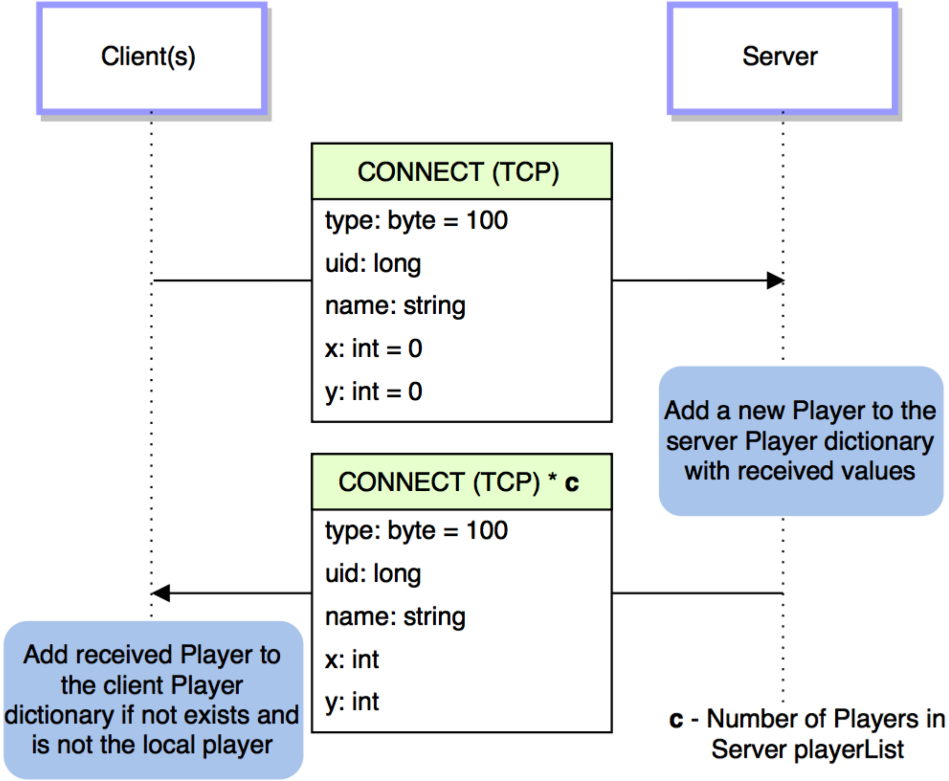


```
Server running on port 14243
Tick interval = 16
Waiting for clients
Chris (UID:6526953420217209483) connected.
New host is Chris (UID:6526953420217209483)
New Station 1391629156 has appeared!
New Station 627415488 has appeared!
New Station 1555564279 has appeared!
Emmanuel (UID:6160412664582618877) connected.
Michael (UID:-1755582996301817983) connected.
Deepio (UID:-4861890452022845889) connected.
Node 800455395 of Station 627415488 was destroyed!
Node 562337043 of Station 627415488 was destroyed!
Node 257612544 of Station 627415488 was destroyed!
Deepio (UID:-4861890452022845889) timed out.
```

```
Michael (-175558299630181798
Chris (6526953420217209483) H
Emmanuel (616041266458261888
```

UNIVERSITY OF LEICESTER

Messaging example: New connection



Client(s)

Server

CONNECT (TCP)

type: byte = 100

uid: long

name: string

x: int = 0

y: int = 0

Add a new Player to the server Player dictionary with received values

CONNECT (TCP) * **c**

type: byte = 100

uid: long

name: string

x: int

y: int

Add received Player to the client Player dictionary if not exists and is not the local player

c - Number of Players in Server playerList

Final Deliverable and Conclusion

- Polished product that resembles quality of a contemporary title
- Room for improvement in the future
- Valuable insight into challenges of producing networked software
- Highly rewarding and enjoyable logic programming and GUI design
- Retrospection on planning and design
- Large and complex overall software product