Thesis

# Contents

# 1 State of the art

## 1.1 Similar Games

### 1.1.1 Keep Talking and Nobody Explodes

The game consists in defusing a ticking bomb by solving puzzles attatched to it. It's a two player game, one player is the defuser and the other has a bomb-defusal manual. The players have to exchange the puzzle descriptions and solutions.

### 1.1.2 Unrailed

The game consists in constructing a railway in front of a moving train. If the trains runs out of track, the game is over. In order to construct rails, the players have to chop trees, mine stone and combine materials in a specialized train cart. It's a 2 to 4 player game. The players usually talk to split tasks among themselves and get time-sensitive tasks done quickly.

### 1.1.3 Among Us

The game is a mafia game between two teams, impostors and crewmates. A team wins if the other team is dead, crewmates also win when they complete all tasks. Impostors may kill crewmates. If a crewmate finds a corpse, he may call a vote. The player who has to most votes is killed. The players have to discuss who to kill in a limited time.

## 1.2 Asymmetric Multiplayer

A game has asymmetrical multiplayer when the players play the game differently. There are multiple levels of asymmetry: from a slight imbalance in the mechanics to a completely separate set of rules. Keep Talking and Nobody Explodes is the most asymmetrical game in the cited games: a player just has a manual. The other two games have a lesser degree of imbalance: in Unrailed the roles and tasks can be exchanged during the games and in amongus every player is controlling a character and interacting in the game's map.

## 2 Applications

### 2.1 Efficent communication skill development

The game is focused in training the communication, planning and problem solving skills of the players. The games puts the players under time pressure and requires a significant amount of information to be exchanged quickly. The game also forces a half duplex communication protocol emulating a radio, such that only one player may speak at a time. Some players will also have to solve space navigation puzzles, obstacle avoidance and local route planning; others will have to make a general plan, assign tasks and deal with multitasking.

### 2.2 Applicable scenarios

This skills are useful in environments where the team has no way to take decisions based on their local space and has to rely on another team that has the global picture to make decision for them. Examples of scenarios in such enviroments are cave exploration or cave rescues, firefighting operations in thick fog, bomb defusal.

### 2.3 Feedback on skill improvements

This is the data gathered from testers.

## 3 Technology used

### 3.1 Input and Devices

The game will be played using mouse and keyboard. The courier players will only have to use the keyboard after the game phase is started.

### 3.2 UDP Sockets with packet fragmentation

The connection is handled using native C# asynchronous UDP sockets. The packet is formed by an id part and a data part. The id is the name of the player, which is a potential security vulnerability. This can be solved by hashing the player string. There is a further vulnerability, as a player can send any id. This is solved by getting the id string from an external authentication service. This approach decouples the ip address from the player identification, therefore a change in the players ip address does't pose any identification problem.
The data part of the packet is further subdivided into a protocol integer and the actual data. The protocol dictates the structure of the data.
This socket infrastructure is used for gamesyncronization packets, audio packets and video packets interchangeably.

### 3.3 Model View Controller

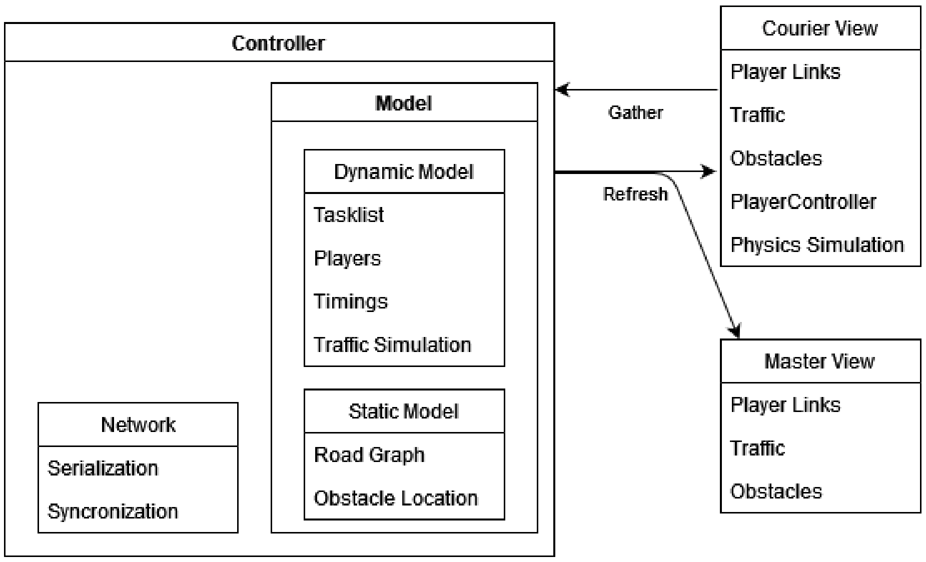

Figure 1: Game architecture diagram

### 3.4 Optimizations Techniques

Culling, LODs, events

# 4 Design

## 4.1 Game Design

The game is set in a city, with gridstyle roads. There are up to 2 couriers, which have to fulfil a list of delivery orders. There is a master player who has a map, which shows the courier's positions and the streets status. The master has the order list, the couriers do not. The orders consist of a pickup site and dropoff site. The streets are blocked by many obstacles:

- variable level of traffic makes the street difficult to navigate
- a drawbridge over a river or a rail
- a rail passage getting closed when a train crosses

The master has to plan the route during the time and relay the path to each courier in the broadcast channel. The master's map is outfitted with a shortest path algorithm to ease finding a fast route. The couriers drive following the Master path, they only see the pickup points and if they have picked up an item, the corrisponding dropoff point.

## 4.2 Traffic simulation

The traffic simulation can be tuned to different levels of complexity.

### 4.2.1 Static

The streets are filled with a random amount of immoble cars.

### 4.2.2 Dynamic

Every street has a traffic level associated to it. The traffic levels oscillate smoothly following a sine wave with randomized period and phase. The cars are immoble, but when they are not seen by any players, the amount of cars in a street is adjusted to be proportional to the traffic level of the street.

### 4.2.3 Agent Based

The cars are agents that move thorugh a graph representation of the streets. The cars roam through the city by turning randomly at intersections. The cars respect traffic lights and obstacles on the streets, so that traffic jams happen dynamically.

## 4.3 Game Variations

### 4.3.1 Events

Throughout the city there are billboards and signs that contain the time and place an event will happen. The event (for example a parade, a festival, the street market, a strike) generates a lot of foot traffic and car traffic in a particular place. The master has no information about these events, so he has to rely on the couriers who will see the billboards and relay the information.

#### 4.3.2 Delivery Timers

The delivery orders may have up to three additional timers: a time after which the order is failed if it's not picked up, a delivery time constraint and a time after which the order is failed if not completed.

## 4.4 Master's Information

The information that the master has of the players can be reduced in the settings. The master has three channels of information: geolocalization, video feed and audio from the radio. Turning off any of these channels the game becomes significantly harder, except when turning off the radio. The radio is essential for completing the game, so it cannot be switched off. A chatting system may substitute it partially.

## 4.5 Interface Design

## 4.6 Syncronization model

The game's state is split in static state and dynamic state. The static state is comprised of the data that does not change during the game and thus is syncronized only at the start of the game. The dynamic state instead is all the data that needs to be syncronized every tick. Every tick, which are a few milliseconds apart, each client sends their position to the server. The server integrates the positions in it's state and computes the next state, which is sent back to the clients. This approach has the advatage that it's naturally resilient to packet loss as the whole state is sent each time. Moreover, there is no network lag on the client character. The approach however has the disadvantage that it's weak against cheating, as the server trusts the position given by the client. This can be solved by running the simulation on the server to check for anomalies or by running it exclusively on the server, gathering the raw inputs of the clients instead of the position.

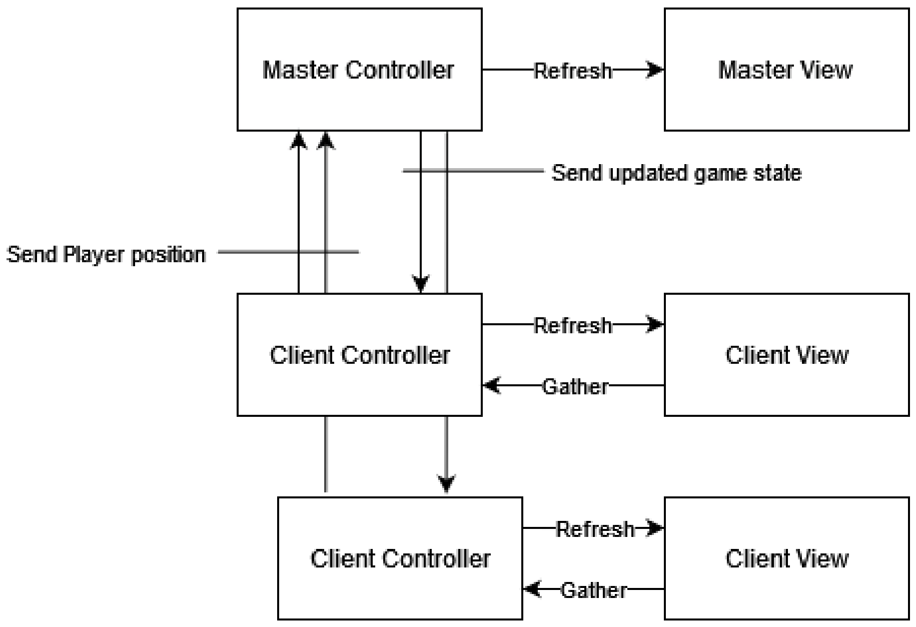

Figure 2: Network Architecture diagram

The static state is constructed by using the configuration of the lobby. In the lobby of the game, the master has the possibility to change map, set the number of tasks, set the available time and turn off video feed and geolocalization. The changes are seen by all clients.

## 4.7 Kinematic Bycicle Model [1]

The model is a simplified car steering model, where the front and back wheels are collapsed into one front wheel and one back wheel. The only steering wheel is the front one.
Known constants:

$$l_r = \text{distance from the center to the rear wheel}$$
$$l_f = \text{distance from the center to the front wheel}$$

State variables:

$$[x, y] = \text{absolute position of the bycicle in the plane.}$$
$$v = \text{velocity of the bicicle.}$$
$$\psi = \text{heading angle}$$

Control variables:

$$u_1 = \text{acceleration}$$
$$u_2 = \text{steering angle of the front wheel.}$$

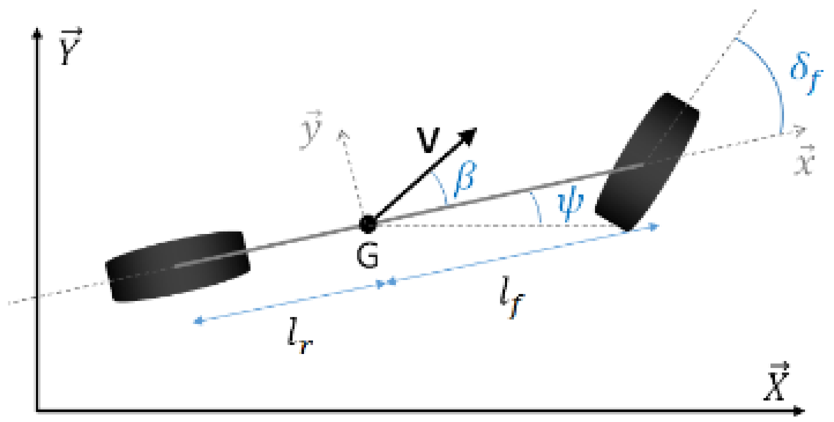

Figure 3: Kinematic bicycle model

The model's differential equations:

$$\dot{x} = v\cos(\psi + \beta(u_2)) \tag{1}$$

$$\dot{y} = v\sin(\psi + \beta(u_2)) \tag{2}$$

$$\dot{v} = u_1 \tag{3}$$

$$\dot{\psi} = \frac{v}{l_r}\sin(\beta(u_2)) \tag{4}$$

where $\beta(u_2)$ is the slip angle given by

$$\beta(u_2) = \arctan(\tan(u_2)\frac{l_r}{l_f + l_r}) \tag{5}$$

The model is integrated discretely.

$$x_{t+1} = x_t + \dot{x}\Delta t \tag{6}$$

$$y_{t+1} = y_t + \dot{y}\Delta t \tag{7}$$

$$\psi_{t+1} = \psi_t + \dot{\psi}\Delta t \tag{8}$$

## 4.8 Radio

The radio in the game is a simulation of a real half-duplex radio. This means that only one player at the time may speak. This is enforced by transmitting white noise while two ore more player are using the radio channel simultaneously. The simulation relies on a server based network to centralize the audio mixing process. All clients send their audio to the server, which mixes all the sources and sends customized mixes to each client and the local speaker. As a convention, all audio passing though the mixer has a sampling rate of 48kHz.

### 4.8.1 Voice Loopback

The client audio doesn't loop back to the sender. When the voice returns to the speaker it's delayed, causing confusion. The server processes all audio sources to calculate if noise is needed, then for every client it mixes all sources except the client one.

### 4.8.2 Buffers

The client audio is sent to the server and cached in a buffer in order to account for desyncronizations and network jitters. The server creates and fills a buffer for every client. Periodically, the server polls the buffers and mixes the polled data. The mix is sent to each client where it is cached in a buffer. This buffer is constantly polled by an Unity audio callback. The draining and filling of the buffers is balanced: the microphones produce as many samples as the ones consumed by the speakers, so the intermediate buffers serve as a transfer node.

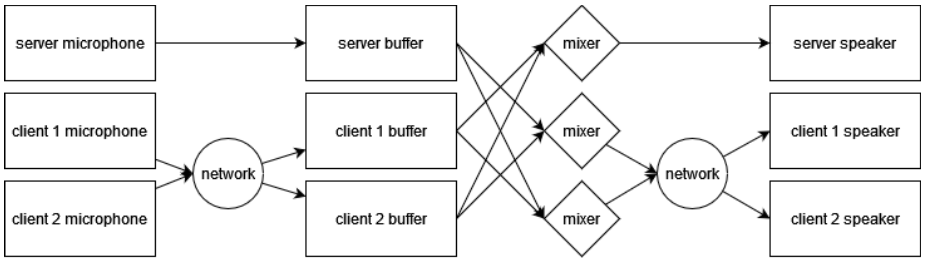

Figure 4: Audio pipeline diagram, the boxes represent the buffers.

Every buffers delays the signal by a set amount. This delay is crucial to alleviate the network variance, which otherwise would cut the signal and override samples. The buffers, when queried to read a certain amount of data, return silence if the data available is less than the requested amount. This ensures that the mixer will always have either a full set of data or silence and that the buffers will not cut the audio.

### 4.8.3 Sampling frequency and Resampling

The client's microphone may have a different sampling frequency from each other. The audio sent is resampled to 48 kHz and the audio received is resampled to the speakers sampling frequency. The sampler used is a custom sample and hold algorithm, it inserts or deletes samples following the ratio of sample rates.

### 4.8.4 Noise and Filters [2]

The noise generated and mixed when multiple clients are speaking is white noise at 48 kHz. The audio generated by the clients is filtered by the server with a custom hi pass filter, which attenuates the low frequencies. This effect is done to make the simulation closer to an old style radio.

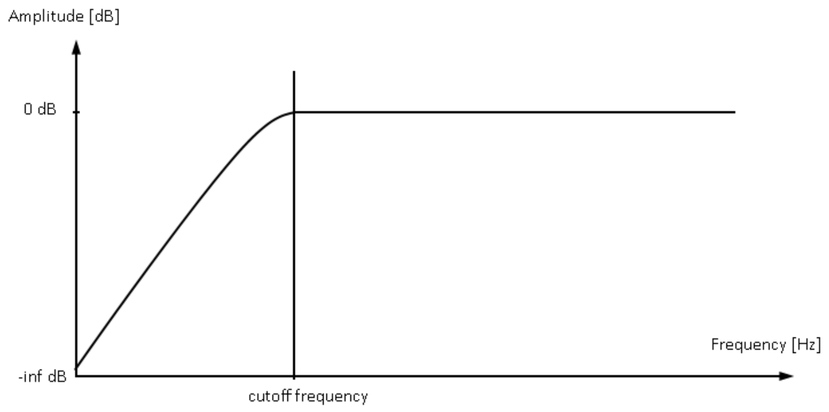

Figure 5: Hi-Pass filter Bode diagram

The filter is a recursive digital filter, defined by the following differential equation:

$$y(n) = \alpha(y(n-1) + x(n) - x(n-1)) \tag{9}$$

## 4.9 Video

The video feed is a frame of the player's camera which is sent to the master by each player. The amount of data required scales rapidly with the resolution of the frame and the frequency of the transmission.

## 4.10 Editor Tools

The development process was sped up by the construction of tools for Unity. This tools are based on editor scripts, which allows to automatize some actions in the editors such as instantiating prefabs (keeping the prefab link) and destroying GameObjects.

### 4.10.1 Map Generator

The map generator places prefabs on a grid, randomizes their z rotation snapping on the 4 cardinal rotations and swaps these prefabs with their updated ones. This script also places road segments ensuring that no segment overlaps another and giving each segment a unique identifier.

### 4.10.2 Graph Generator

This tool generates a graph (a nodes and arcs tuple) from the elements present in the scene. The graph is saved directly in the Assets/Resources folder in json format. The graph can be visualized in the scene with a toggle in the script.

# 5 Testing

## 5.1 Unit Testing and TDD

The model of the game is developed using the test driven development technique. This technique allows writing testable code by first writing a failing test and then the code to pass it. This makes the code more robust and easier to clean. Other aspect of the game such as the controller, the view or the map tools are not easily testable this way as they depend on the unity framework.

## 5.2 Integration Testing

The integration testing phase is largely done by hand. This is necessary as unity does't have a way to run multiple deployed instances inside a test environment, the testing routine is to deploy an executable and run it parallel to the editor instance. Some components are tested automatically, such as the serialization of the model and the interaction between smaller model components.

## References

[1] Philip Polack, Florent Altché, Brigitte Novel, and Arnaud de La Fortelle. The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles? pages 812–818, 06 2017.

[2] Julius O. Smith. *Introduction to Digital Filters with Audio Applications*. W3K Publishing, 2007.