Fox and Geese

The digital multiplayer and multiplatform rediscover of an ancient board game

Jordan Gottardo (1179739)
Dipartimento di Matematica
University of Padua via Trieste, 63, 35121 Padua, Italy
jordan.gottardo@studenti.math.unipd.it

Giulia Petenazzi (1180066)
Dipartimento di Matematica
University of Padua via Trieste, 63, 35121 Padua, Italy
giulia.petenazzi@studenti.math.unipd.it

***Abstract*—**

The technological evolution in the mobile phone sector has had an exponential growth in the last years. The number of functionalities has grown, and of course the way we use these devices has changed. Starting with calls and messages, introducing cameras, implementing internet connectivity, and finally introducing the sensor technology, we arrive to today's smartphones. But if you think for a while about this, you probably realize that we use them to respond to basic human needs: communication, feeling accepted from society, knowing something about a topic and having fun. In this paper we want to focus on the "having fun" part, because mobile games represent one of the most important kinds of application in terms of number of apps, downloads, and users. Following this trend, a popular application as gaming has extended its domain from home computers and consoles to the mobile realm. On the one hand, nowadays there are a lot of developers working on modern game genres, but on the other hand, we could attempt to create mobile games that join the rediscovery of ancient board games with the pleasure to play on a smart device. To this aim, we have created an application for mobile phones exploiting features available in every modern smartphone (in particular the internet wireless connectivity), to digitally rediscover and enhance an ancient board game called "Fox and Geese".

I. Introduction

A. State of the art

We are living in a revolutionary era for communication: smartphones, in the last few years, have increased their capabilities in terms of memory storage, CPU power, and connectivity (e.g., 3G, 4G, Wi-Fi, Bluetooth) and almost two-thirds of the world's population now has a mobile phone and more than half of the world's web traffic now comes from mobile phones. [1]

The monthly mobile data used by the average smartphone worldwide in august 2017 was 2.3GB, and is exponentially increasing every year. [2]

B. Italian people

With regard to Italy, Italians are connecting to the Internet more and more, and they do it more often using their smartphones. The percentage of people who navigate from a computer has fallen by 14%, but the percentage of people connecting from any other device has increased (+ 44% from a smartphone, + 8% from a tablet) compared to 2016. [3]
A curious thing is that Italians prefer 3G / 4G mobile networks to Wi-Fi. This fact assigns our country an European record in this matter. The main reason of this behaviour is the lack of availability of public and private Wi-Fi networks in our country to which you can connect for free , but users cite also the better performance of 3G / 4G, and the need to share live video of their most important moments. In our work we will take in consideration also this aspect.
As a result of this growth, users are using more and more applications, in terms of both numbers and complexity. For application developers, 2016 was a golden year: the number of downloads increased by 15%, while revenues jumped by 40%, compared to the previous year. [4]

C. Cross-platform design

In this jungle of devices, we notice that the world's smartphone market is clearly split between Android and iOS. The use of well known and open software platforms like Android and the free availability of development native frameworks (Android studio for native Android) and cross-platform frameworks that allows the "write once run anywhere" approach has attracted the business world together with researchers from various fields: gaming, networking, artificial intelligence, communication, imaging, and many others.

In Italy Android remains the most widespread platform and conquers 72% of the market, but on the other hand Apple users are more likely to spend money on apps. For this reason we think developing a multi-platform application could be a winning point. [7]

D. Gaming applications

With regard to game app downloads, this year the Play Store has tripled the figures of the App Store: 27.2 billion

titles downloaded on the BigG store, while for Apple's counterpart we have 8.3 billion downloads. In total, the revenue generated by the video game industry in 2016 was $30.4 billion and 53% of the most frequent gamers play multiplayer games at least once a week, spending an average of 6 hours per week playing with others online and 5 hours playing with others in person. [5] [6].

E. Our purpose

Our approach tries to merge the above points: a digital and cross-platform version of a medieval game that emphasizes the real-time aspect and allows users to play with any other player interested in the same game, regardless of any distance. We believe this purpose can be reached with common devices, such as smartphones, using the typical equipment supplied with them, such as internet connection to play with another player. An additional challenges in a multiplayer implementation is emphasizing the "real time" aspect, in a way that the player sees the opponent's moves at every moment, and not only an approximation of them.

New technologies permit to make ancient games more amusing and to increase their fun potential through remote gaming, automatic check of game rules, score ranking etc. The small size of mobile phones and easy games allow people to have fun everywhere and anytime.

Summarizing, we wanted to develop an application that shows how certain games are playable on a home PC/consoles (Xbox for instance) and also on Android and iOS smartphones, thus allowing a more well rounded playing experience. Therefore, mobile games do not have to be to be limited by the type of network that connects two players.

Users should be able to play this kind of game not only if they are in the same room or building/area, but also if they are far away from each other. For this reason we not to limit the game to be playable only with Bluetooth or Wi-Fi technology, but also allowing users to play on the Internet.

Indeed, our game at the current version is specifically tailored for mobile devices, exploiting their peculiar features, but thanks to the framework we used we could deploy our game to many other platforms.

F. Game introduction

Fox and Geese is a turn-based hunt game from northern Europe. The game seems to have surfaced in late medieval times; one of the earliest record of Fox and Geese is from the accounts of XV century king Edward IV of England. Starting from that period, various examples and modifications have been played historically throughout Europe. The game eventually spread to North America and has been adopted by the native Americans.

Later games diversified to the point that none of these variants can now be claimed to be the standard. The game is played on a cross-shaped board, known to modern players as a peg solitaire board. Thirteen geese must trap a fox on this board, while the fox attempts to capture enough geese to prevent them from achieving their objective.

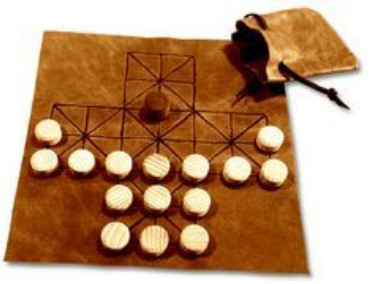


Figure 1: real Fox and Geese board

We have chosen to create a version of this game which can be played on smartphones: a player moves his pawn and once the move is declared as valid move, the game moves forward.
The rest of this paper is organized as follows. In Section II we analyze basic requirements for our application. Then, in Section III, we delve into design and implementation issues related to the project, whereas in Section IV we discuss several interesting directions for expanding the current work. Finally, Section V concludes this paper.

II. Application Requirements

Before discussing the project design, we need to describe our game and to point out some basic requirements for the project; here, we present a list and a description of the main ones.

A. Game mechanics

This game is part of the category of unbalanced games, in fact we can notice that he probability of victory for each player is not the same. In order to solve this situation we decided to let on of the players choose the number of geese, ranging from 13 to 17. Another feature we added to preserve fairness is the randomization of the player's roles at the start of each match.

This game is played in turns by two users. Below are listed the game rules.

- The geese take the first turn. The player has to move one of them from its starting position, along any marked line, to an adjacent empty point.
- The fox then takes a turn, moving in exactly the same manner as the geese. Play then alternates between the two players.
- No goose can kill the fox.
- Instead of moving as already described, the fox may kill an adjacent goose by jumping over it onto the empty point beyond, providing that the points are linked by a marked line. The goose is then removed from the board.
- If the fox, having jumped, is in a position to kill a second goose in the same manner, he can do so immediately. Any number of subsequent geese can be so killed during the fox's turn, if their opponent is foolish enough to leave them arranged in that way.

- The geese win the game by trapping the fox, so that it is unable to move during its next turn.
- The fox wins by capturing most of the geese. When 4 or less geese remain on the gameboard they are unable to trap the fox, so the winner is the fox . [8]

We decided to focus our efforts in the multiplayer mode of the game because, even historically, this is the main game mode, and of course the user-experience is improved if the player knows that there's a real person and not an artificial intelligence behind his opponent. However the single player mode could easily be implemented using the modules we created for the multiplayer version. Players can be split into the following two type of players: (a) Server: who starts a new session of the game; (b) Client: who wants to join an existing match.

First of all, when a user opens the application, he decides if he wants to join an existing match or if he wants to create a new one. In the first option the user becomes a "client", so he will see a list of all the existing matches, and when he selects one of them the game starts. If the user chooses the second option he becomes a "server", so he establishes the name of the match he wants to create and waits for players. When a client joins the match, the server will choose the geese number, then the game starts. As we said before, the roles are chosen randomly after the number of geese is chosen to preserve fairness between the players.

B. *Device compatibility*

This game has been designed to be a cross-platform application. For practical reasons we have tested the game on two Android phones (Nubia Z11 mini S and Nexus 5, both with Android 6.0.1) and on two laptops (Lenovo G500 and Dell Inspiron 7559, both with Windows 10).

III. Design and implementation

A. *General architecture*

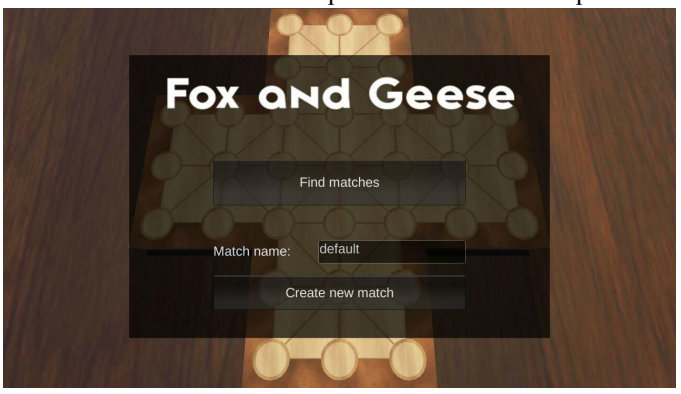
The application has been developed using the Unity framework, which has provided a baseline for developing both 2D and 3D games and for deploying on multiple platforms.

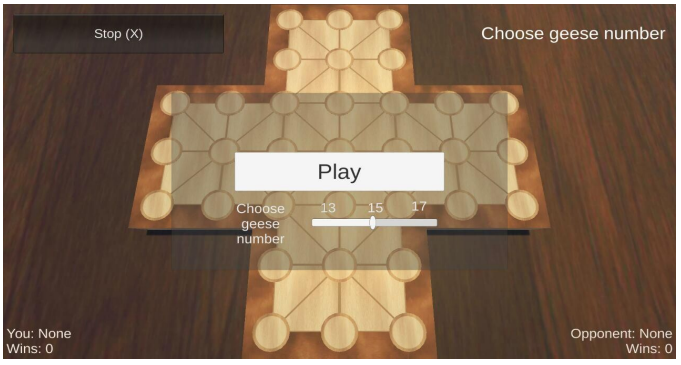
To better promote decoupling and code reuse, we have split our code into modules.

- *UI*. This module manages the user's interaction while creating and joining games.
- *Board*. This module manages the rendering of the game board and the various pawns placed on it.
- *Game*. This module manages the game's logic by maintaining a local state of the game and modifying it whenever players move the pawns. It also provides a means to check the validity of moves and victory conditions.
- *Network manager*. This module manages the communication between player's clients and the server. It uses UDP-based Unity networking (UNET) high level scripting API (HLAPI) to manage communication in a client/server architecture. We have made this choice due to the fact that the HLAPI are platform-agnostic, so they can ensure compatibility with all platforms supported by the Unity engine. In addition to that, UNET offers useful Internet services, such as a matchmaking service and easy routing of messages between clients.
- *Controller*. This module contains a series of controllers that manages the interactions between the Game and Board modules. These controllers respond to events coming from other modules and act accordingly.

With this separation into modules, it should be fairly easy to extend the application with other functionalities and even reuse some module in a future application. For example, we deem that changes in the game logic are simple to bring about due to the fact that the Game module is highly cohesive and other modules don't make assumptions on its behaviour.

At the moment the network manager only supports Internet connectivity. We believe that implementing Wi-Fi or Bluetooth communication into our application is feasible. More information about this topic can be found in chapter IV.





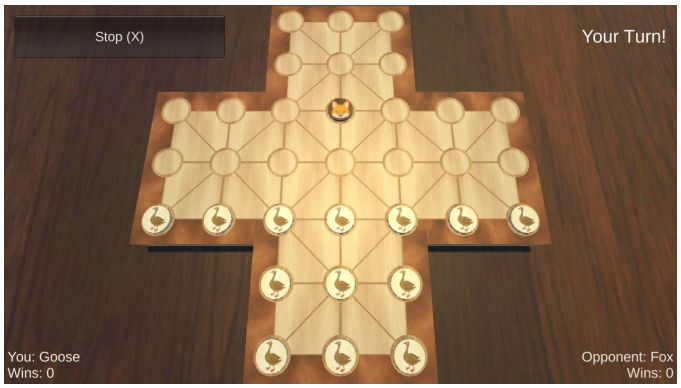


Figure 2: the geese win by capturing the fox

B. Client/server communication

As stated before, the application's network manager uses UNET. This means that the network architecture is based on client/server communication. When a player chooses to host a game, he takes the role of the server and the Unity matchmaking service adds an entry to the list of open games. If another player wants to play, he can query the matchmaking service to view all open games and join one of them. The hosting player has to have the possibility to play as well, so actually he is both a server and a client.

When two players connect with each other, a representation of each player is created within each client. We will call that representation "player prefab". Even if this game has no first or third person controls of the player's character, the prefabs are created nevertheless since this is at the base of UNET. Figure 3 depicts this situation. In that picture we can see that each client has two player characters. The prefab which represents the player using that client is marked as "localPlayer".

All communication between client and server is routed through the player prefabs and is executed via API calls.

- Client -> server: the communication uses "Cmds", or "Commands". These are methods invoked by the clients and executed on the server.
- Server -> client: the communication uses "ClientRpcs" or "Remote Procedure Calls". This methods, invoked by the server, are executed on each client.

By combining Cmds and ClientRpcs we have managed to implement all the client/server interaction needed.

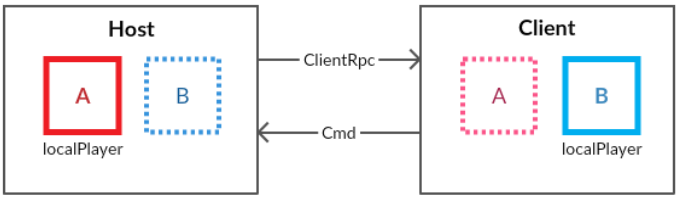


Figure 3: host and client player prefabs

C. Game phases

From a logical standpoint, the game can be divided into several phases.

- *Waiting for players*. The host has started a game and is waiting for another client to connect.
- *Choosing geese number*. The players have connected and the host is choosing how many geese will be used in the next match.
- *Coin toss*. The host has chosen the geese number and a coin is tossed to determine who will play as fox and who as goose.
- *Playing*. The game is underway.
- *Victory*. The game state has reached a victory condition and the game ends, declaring one of the players as the winner.
- *Restarting*. The game automatically restarts from the "Choosing geese number" phase.

In particular, the coin toss phase is managed by the server which tosses a coin and lets the clients know about the result. This communication is carried out via a combination of Cmds and Rpcs. Figure X.X depicts this situation.

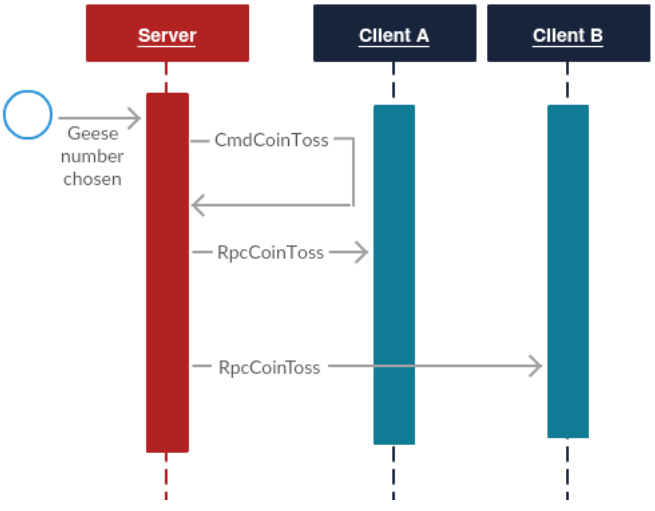


Figure 4: coin toss representation

D. *Real time animations*

Since we deemed that implementing real time animations would have enhanced the players' experience, we have added this functionality. Each player can see the movements of the other player's pawns at each moment, and not only when the move is completed. These kind of animations offer a seamless experience and help the players believe they are playing with another human and not with a cold and computerized AI.

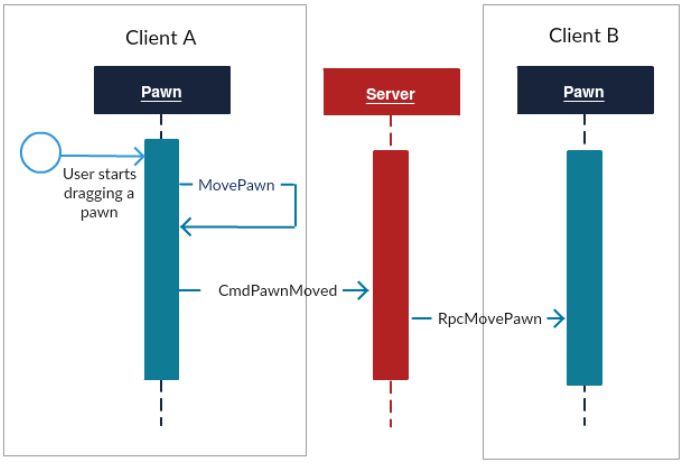


Figure 5: movement representation

We see that whenever a player moves a pawn, a Cmd is invoked on the server which then invokes a ClientRpc. In this way, the other client is notified of the movement and can render it immediately.

The movement animation and validity check on the move on the client side is carried out immediately and doesn't wait for the Rpc. While being more prone to cheating, we found out this solution to be best. Waiting for the Rpc coming back from the server would mean waiting 2*RTT, and animation quality would suffer greatly.

IV. Future work and extensions

In this section we describe additional features we would like to implement in the future.

A. *Bluetooth and Wi-Fi support*

At the moment the application only supports connectivity via the Internet. We believe adding other types of connectivity could be useful for various reasons, such as:

- saving battery;
- playing even when the Internet connectivity is unavailable;
- promoting "in person" play, which could better resemble the way the original game was meant to be played: in front of another person while sitting at a table or on the floor.

An attempt to implement one of the aforementioned kinds of connectivity should not be using UNET. In fact, neither Bluetooth nor Wi-Fi are supported by UNET.

With regard to Bluetooth implementation, we have found some feasible solutions. The first one is based on using a paid plugin in the Unity asset store [11], which is well integrated with the HLAPI, so code modifications should be expected to be minimal. Unfortunately this plugin only works on Android and is fairly pricey. Another solution could be exporting the Unity project into a platform specific project (e.g. Android or iOS) and starting to implement the functionality on that project. The second solution, while requiring a lot of man-hours, would work for all platforms, but would require specific coding for each platform.

B. *Pre-game lobbies*

The game would benefit from having a pre-game lobby before the start of the game. Players would be able to chat with each other, talk about what settings the next game should be played with (e.g. number of geese) and choose their pawn type (fox or goose) before the start of the game.

We deem the best way to implement this functionality would be using Unity multiplayer lobbies, which provide facilities to develop these kind of requirements.

C. *iOS deployment and testing*

At the moment we have deployed and tested our application on the Android and PC platforms. As said in chapter I, iOS is another major player in the mobile app department. Since we have implemented the app using a cross-platform framework, we are fairly confident the game will work on iOS too, but we haven't carried out deployment nor testing due to missing devices and time constraints.

D. *Single player game mode*

While multiplayer is inherently more fun, a player may want to play by himself to train his strategies or avoid the stress caused by competitiveness. To satisfy this need, the game would need a single player mode with an AI capable of playing the game at various difficulty levels.

V. Conclusions

In this paper we have discussed about the importance of mobile games in the modern market. We have discovered that supporting various platforms is important now and may be even more in the future.

In a world which is more and more connected, we have rediscovered, explained and implemented a classic board game. The resulting proof of concept application works on Android and PC at the moment.

Finally, we have proposed several extensions to our game to improve platform support and connectivity and to make it more fun by introducing new game modes.

References

[1] Digital in 2017: Global Overview - [https://wearesocial.com/special-reports/digital-in-2017-global-overview]

[2] Global Digital Statshot Q3 2017 - [https://www.slideshare.net/wearesocialsg/global-digital-statshot-q3-2017]
[3] Digital in 2017: Italy and the world - [https://wearesocial.com/it/blog/2017/01/digital-in-2017-in-italia-e-nel-mondo]
[4] App market in 2016 - [http://www.fastweb.it/smartphone-e-gadget/il-mercato-delle-app-nel-2016/]
[5] Play Store doubling App Store in 2016 - [http://www.mobileworld.it/2018/01/09/app-installate-guadagni-play-store-app-store-141318/]
[6] 2017 sales, demographic, and usage data essential facts - [http://www.theesa.com/wp-content/uploads/2017/06/!EF2017_Design_FinalDigital.pdf]
[7] The State of Cross Device Commerce - [http://www.criteo.com/it/wp-content/uploads/sites/9/2017/12/criteo-state-of-cross-device-commerce-2016-h2-it.pdf]
[8] Simon Says the Color: The Digital Evolution of an Outdoor Kids Game - [http://www.math.unipd.it/~cpalazzi/files/Relazioni.rar]
[9] Unity website - [https://unity3d.com/]
[10] Unity networking overview - [https://docs.unity3d.com/Manual/UNetOverview.html]
[11] Unity bluetooth plugin - [https://www.assetstore.unity3d.com/en/#!/content/9643]