**The Coding of Chess Variations  
Implementing Different Ways to Play Chess Into a Standard Chess Program**



**Matura Paper, Kantonsschule Sargans**

Attila Vizhanyo, 4bNP

**Supervisor:**Herr Lukas Fehr

**Submitted on:**  
8 January 2024

# Preface

In the pursuit of finding a desirable topic for my Matura project, my objective was to combine two of my passions: chess and coding. I have been playing chess for an extended period. My grandfather first introduced me to the game when I was young. He and my uncle played frequently against each other, and by watching them play, I was inspired to start playing as well. I never indulged in chess competitively, but I enjoy playing with my family and friends. Its simple rules and complicated strategies appealed to me. My second interest revolving around my work is coding. Being raised by parents who were programmers, I came across the subject of computer languages often, however I did not have proper coding experience prior to my work. Despite my limited programming knowledge, I found coding appealing, because of its logical way of thinking and creativity. So, I was driven to learn this skill in the result of my work.

With my goal in mind, I came up with the idea of programming different versions of chess by changing the rules of the age-old game. I was inspired by a website called chess.com that has published multiple variations of chess, which I found enjoyable to play. These alternatives ways of playing chess made the game less serious and more diversified. Consequently, I was excited to come up with my own ideas of ways to play chess. By changing the rules accordingly, my goal was to make chess more enjoyable for people that may not appreciate its strategic complexity and pure memorization.

A game of chess pieces

Description automatically generatedI took inspiration from a quote of the famous chess grand master Bobby Fisher, whose respond was the following to an interview question:

Figure 1 Chess 960 Starting Position

“**Interviewer**: Why do you hate Chess? Being the be… probably, possibly, the best Chess player ever?

**Bobby**: Because I know what Chess all is about! It’s all about memorization. It’s all about pre-arrangement…” (algekalipso, 2022)

In this statement, Fisher aims to emphasize that a significant aspect of playing chess involves memorization. Because the starting position of chess is always the same, at advanced levels of chess, the initial moves, known as the opening, are preplanned. In this phase experienced players know the optimal response to each move, which they learn prior the game. Fisher heavily criticizes this aspect of the game because it does not involve creative thinking, rather than rote learning. In response Fisher came up with a new way of playing chess. In this variation, the initial row on each side, where the pieces, excluding the pawns, are positioned, are randomly rearranged. With each new game, players encounter a randomly shuffled board that they have not prepared for. In this way players must prioritize strategy and creativity rather than relying solely on pregame knowledge. This game mode is called Fischer Random Chess or Chess960 and it was published in 1996.

Following my work, Fisher inspired me to mitigate the memorization aspect of the game and focus on creativity.

# Table of Contents

[Preface III](#_Toc155093227)

[Table of Contents IV](#_Toc155093228)

[1 Introduction 1](#_Toc155093229)

[1.1 Objectives and Guiding Questions 1](#_Toc155093230)

[1.2 Procedure and Method 1](#_Toc155093231)

[1.3 Structure of the Paper 1](#_Toc155093232)

[2 Initial Chess Code 2](#_Toc155093233)

[2.1 Logic of initial chess code 2](#_Toc155093234)

[2.1.1 Overview 2](#_Toc155093235)

[2.1.2 Concept 2](#_Toc155093236)

[2.2 GUI 4](#_Toc155093237)

[2.2.1 Overview 4](#_Toc155093238)

[2.2.2 Visualization Chess Pieces 4](#_Toc155093239)

[2.2.3 Visualization Chessboard 4](#_Toc155093240)

[2.2.4 Update Game after a move 4](#_Toc155093241)

[2.3 Function catalog 4](#_Toc155093242)

[3 Color Chess 5](#_Toc155093243)

[3.1 Rules of Color Chess 5](#_Toc155093244)

[3.2 Implementation of Color Chess 5](#_Toc155093245)

[4 Chaotic Chess 6](#_Toc155093246)

[4.1 Rules of Chaotic Chess 6](#_Toc155093247)

[4.1.1 Barrier 6](#_Toc155093248)

[4.1.2 Shield 6](#_Toc155093249)

[4.1.3 Coin 6](#_Toc155093250)

[4.1.4 Bomb 6](#_Toc155093251)

[4.2 Implementation of Chaotic Chess 6](#_Toc155093252)

[4.2.1 Overview 6](#_Toc155093253)

[4.2.2 Barrier 6](#_Toc155093254)

[4.2.3 Shield 6](#_Toc155093255)

[4.2.4 Coin 6](#_Toc155093256)

[4.2.5 Bomb 6](#_Toc155093257)

[5 Summary 7](#_Toc155093258)

[Bibliography IV](#_Toc155093259)

[Table of Figures V](#_Toc155093260)

[Appendix VI](#_Toc155093261)

[A. Appendix 1 VI](#_Toc155093262)

[B. Appendix 2 VII](#_Toc155093263)

[C. Declaration of Authenticity VIII](#_Toc155093264)

# Introduction

## Objectives and Guiding Questions

Compute the legit moves of each piece. Visualize the game.

Enforce the rules of the game, which includes…

## Procedure and Method

At the start of my project, I had to determine in which programming language I wanted my project to be written in. Since I had little knowledge about programming, I opted for an easy to understand and beginner friendly computer language. Python is one of the most well-known starting programming languages, making it an ideal choice for my project. Since I had no prior experience with Python, I had to learn the language from scratch. I started my learning journey with a great introduction tutorial I found on YouTube. The author of the video explains the basic functions and variables of Python and reinforces the information with step-by-step projects. I found this guide helpful and engaging, resulting in my improvement in programming.

Next up, I had to choose a chess code, that I could understand well and that can be modified to create new chess variations. I found my way over to GitHub where thousands of programmers are sharing their code with the world. GitHub search features allowed me to look for a chess code that is written in python. After looking through dozens of chess programs, I finally found one to my liking. The one I found did not have major errors and its board was constructed with buttons. I found working with buttons simple, because each button had a distinct identity and when pressed it can call a function.

GitHub allowed me to save my projects in a history data base, which is crucial in programming. When writing any code, it is important to make many backups to jump back to the code if a long-term error or mistake happens.

For my workspace of my program, I chose visual studio code. Visual studio is a well-known source code editor that I have used in school prior to my work. It’s supporting a wide range of programming languages with loads of extensions. During my work however I came across one problem with this workspace, from time to time it marked that I had an indent error in my code, which wasn’t the case. To fix this I had to rearrange my code, for it not to be shown as an error.

The next step was to fully understand the chosen code functions and variables. When I first inspected the code, it appeared very unclear and was difficult to understand. It ranged with a lot of functions and variables that had a lot of code inside of them. For me to fully understand the program I had to go line by line and understand what each variable and function is for. This took a long time, since the code was written by a more experience Python user than me. So, I had to look up a lot of theory regarding Python. Eventually I started understanding the code better and I got ideas how I could modify the code to create my both chess variants.

During my coding of Chaotic Chess, I took use of modules. A module is a separate coding space, that contains functions that you want to include in your initial code.

## Structure of the Paper

# Initial Chess Code

## Logic of initial chess code

### Overview

Concept

#### Figure (or Piece)

A Class is like an object constructor, or a "blueprint" for creating objects. (W3Schools, s.d.)

The first key concept that needs to be modeled in any chess program is the concept of the chess pieces, also called the figures. There are sixteen pieces of each color, black and white: one king, one queen, two rooks, two bishops, two knights, and eight pawns. Each piece has a type and a position. Each piece has a defined starting position at the beginning of the game. During the game, the players move the white and black pieces in alternate turns, thus the position of the pieces need to be updated. The pieces can move to a square that is either unoccupied or occupied by the opponent’s piece. Each figure needs to know which squares it can move to, based on its type.

A white paper with black text

Description automatically generated

|  |  |  |
| --- | --- | --- |
| Member name | Description | Comments |
| name |  |  |
| object\_name |  |  |
| color |  |  |
| position |  |  |
| possible\_moves |  |  |

#### Chessboard

Chess is played on a square board of eight rows and eight columns.

#### Position

The squares of the board and the chess pieces have a separate grid to arrange them. The grid for the buttons ranges from row and column one to eight. Whereas the grid arranging the pieces range from row and column zero to seven. This is because computers start counting from zero. So, when comparing the two grids, we must subtract one from the grid of the buttons to correspond to the grid utilized by the pieces.

A screenshot of a game

Description automatically generated

Figure 2 This is the grid used by the buttons.

#### A game of chess with a checkerboard and chess pieces Description automatically generated

Figure 3 This is the grid used by the pieces.

In the code the chess pieces are stored as individual objects of the Figure class. If we want to interact with the piece, we must call them by their name. The pieces are named the following way.

#### A green and white checkered board with numbers and letters Description automatically generated

Figure 4 The pieces are named this way.

#### Chessboard

#### Turn

In the game of chess, each player makes the move one after the other.

Turn is a variable that the program uses to determine which player is making the next move. This variable is important to check whether the move played is legitimate. The turn variable can be set to “W” (for white) or “B” (for black). If a legit move has been played, the Turn must change to the other value. If the player does not make a valid move, the program must not change the value of this variable.

#### Capture

#### Check

#### Checkmate

### Control Flow

### Error Handling

A well-written code must deal with its errors. If an error occurs, the program freezes and the user can no longer interact with it. This leaves the user with no idea what caused the problem. To prevent this, the original chess code introduces a variable called “error”. This variable is used to tell the user what problem occurred while interacting with the program. The program continues to run and since the user is informed about the problem, he can avoid it. The error variable can be equal to four values.

If the error is set to "1", the provided move was invalid.

When the error equals “2”, the game is over.

In the case where the error is “3”, the king is in check.

Should the error be equal to “4”, there is a checkmate.

## GUI

### Overview

Tkinter (algekalipso, 2022)

### Visualization Chess Pieces

The chess pieces displayed on the GUI are represented by Unicode Characters.

Unicode is a universal character set that defines all the characters needed for writing the majority of living languages in use on computers. (W3Schools, s.d.)

The Unicodes are stored within a variable. Every identical chess piece has the same Unicode. There must be a total of twelve Unicodes to represent the black and white pieces.

A Unicode functions like a text. We can put a chess piece on a button by configuring the button to contain the text of a selected Unicode. This way we can set the starting position of the chess game by setting the text of the buttons.

A black background with a black square

Description automatically generated with medium confidence

Figure 5 '|u265C' Unicode character for a black rook

BR = '\u265C'

a8 = Button(tk, text=BR, font='Times 20 bold', bg='white', height=2, width=5, command=lambda: [btnClick(a8), btnID('a8')])

### Visualization Chessboard

### Update Game after a move

When the user clicks on the piece he wants to move with, the program saves the text of its button. Once the user clicks on the button where

## Function catalog

A function is a block of code which only runs when it is called. You can pass data, known as parameters, into a function. A function can return data as a result. (W3Schools, s.d.)

# Color Chess

## Rules of Color Chess

In Color Chess, players are assigned an individual color. When their chess pieces move to a square, that square is highlighted with the player's distinctive color. The goal of the game is to color in more squares with your own color than your opponent does with theirs. Only squares on which the pieces stand on or have been standing on are colored in. If a piece captures the opponent’s piece, it will overtake its color. Since each game of chess starts with the same position, the first two rows on each side will be colored in at the beginning. The game ends after 30 moves. The player who has colored in more squares during this period wins. Alternatively, victory can be attained by checkmating the opponent within these 30 moves.

A screenshot of a game

Description automatically generated

Figure 6 Color Chess Starting Position

A screenshot of a game

Description automatically generated

Figure 5 Color Chess Piece Capture Example Prior

A screenshot of a game

Description automatically generated

Figure 7 Color Chess Piece Capture Example After

## Implementation of Color Chess

# Chaotic Chess

## Rules of Chaotic Chess

Chaotic Chess introduces four items that the chess pieces can pick up, granting them special abilities. To make the game balanced, the items are equally distributed on empty fields on the white side (rows one to four) and on the black side (rows five to eight). A piece can pick up an item by stepping on its according square. The items change their positions every four moves successively. The game comes to an end if either king is in checkmate.

The four items consist of a bomb, shield, coin and barrier.

A screenshot of a game

Description automatically generated

Figure 8 Chaotic Chess Random Position Example

### Barrier

The square that the barrier is placed on, prevents pieces to step onto that field. Only the field that the barrier is placed on is affected, pieces can jump over the barrier.

A screenshot of a game

Description automatically generated

Figure 9 Chaotic Chess Barrier Usage Example Prior

A screenshot of a game

Description automatically generated

Figure 10 Chaotic Chess Barrier Usage Example After

### Shiel

The shield makes the piece that steps on it invincible until the shield changes its position. Anchored to its square, the shield does not move with the piece that picked it up.

A screenshot of a game

Description automatically generated

Figure 11 Chaotic Chess Shield Usage Example Prior

A screenshot of a game

Description automatically generated

Figure 12 Chaotic Chess Shield Usage Example After

### Coin

The shield makes the piece that steps on it invincible until the shield changes its position. Anchored to its square, the shield does not move with the piece that picked it up.

A screenshot of a game

Description automatically generated

Figure 13 Chaotic Chess Coin Usage Example Prior

A screenshot of a game

Description automatically generated

Figure 14 Chaotic Chess Coin Usage Example After

### Bomb

Picking up the bomb triggers an explosion in a three-by-three area. Pieces within that area are destroyed and removed from the board. The piece that initially stepped on the bomb is eliminated as well. If a bomb is near the edges or corners of the board, the explosion radius only takes up as much space as it is granted.

A screenshot of a game

Description automatically generated

Figure 15 Chaotic Chess Bomb Usage Example Prior

A screenshot of a game

Description automatically generated

Figure 16 Chaotic Chess Bomb Usage Example After

## Implementation of Chaotic Chess

### Overview

#### Modules

#### Differences In Chess Code

#### Usage Of Initial Chess Code

### Barrier

### Shield

### Coin

### Bomb

# Summary

# Bibliography

Metzger, P. (2010). *Abschlussarbeiten.* Aarau: Sauerländer.

Bonati, P., & Hadorn, R. (2009). *Matura- und andere selbständige Arbeiten betreuen* (2. Auflage ed.). Bern: Heb Verlag AG.

*Panorama*. (2015, December 15). Retrieved December 6, 2019, from Statistik Schweiz - Panorama: http://www.bfs.admin.ch/bfs/portal/de/index/themen/01/01/pan.html

algekalipso. (25. July 2022). *Qualia Computing*. Von Qualia Computing: https://qualiacomputing.com/2022/07/25/bobby-fischer-1972-world-chess-champion-on-why-chess-is-a-lousy-game-and-how-to-save-it/#:~:text=Bobby%20Fischer%3A%20No%2C%20no%2C,I%20don't%20need%20it. abgerufen

# Table of Figures

[Figure 1 Chess 960 Starting Position III](file:////Users/attila/Documents/GitHub/matura-arbeit/matura-paper-template.docx#_Toc155097643)

[Figure 2 This is the grid used by the buttons. 2](#_Toc155097644)

[Figure 3 This is the grid used by the pieces. 2](#_Toc155097645)

[Figure 4 The pieces are named this way. 3](#_Toc155097646)

[Figure 5 Color Chess Starting Position 5](#_Toc155097647)

[Figure 6 Color Chess Piece Capture Example After 6](#_Toc155097648)

[Figure 7 Chaotic Chess Random Position Example 7](#_Toc155097649)

[Figure 8 Chaotic Chess Barrier Usage Example Prior 7](#_Toc155097650)

[Figure 9 Chaotic Chess Barrier Usage Example After 8](#_Toc155097651)

[Figure 10 Chaotic Chess Shield Usage Example Prior 8](#_Toc155097652)

[Figure 11 Chaotic Chess Shield Usage Example After 9](#_Toc155097653)

[Figure 12 Chaotic Chess Coin Usage Example Prior 9](#_Toc155097654)

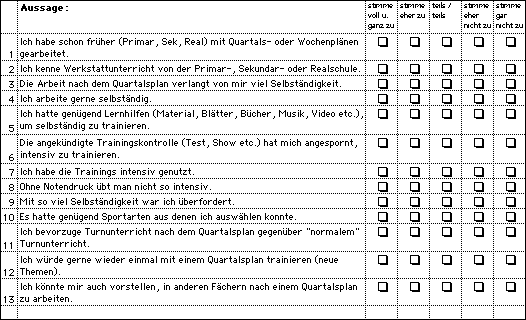
[Figure 13 Chaotic Chess Coin Usage Example After 10](#_Toc155097655)

[Figure 14 Chaotic Chess Bomb Usage Example Prior 10](#_Toc155097656)

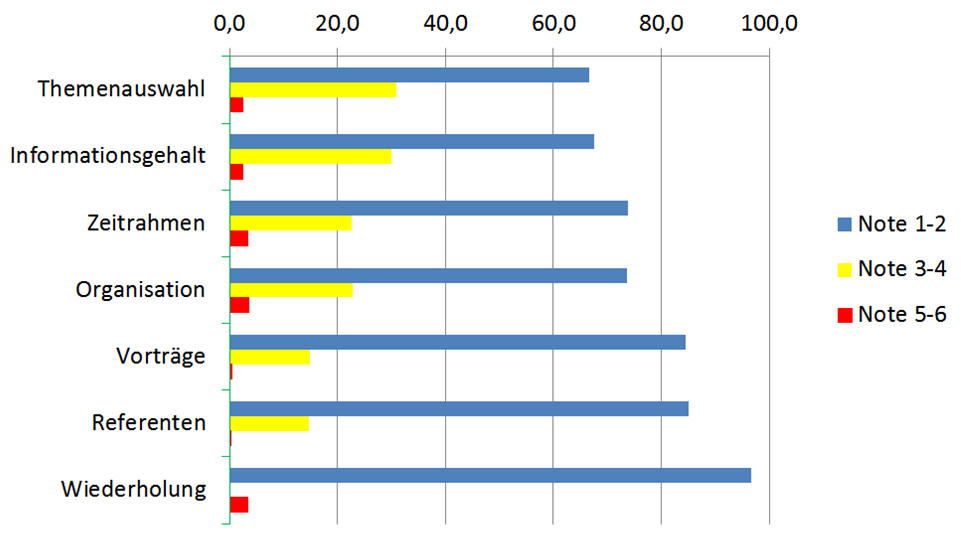
[Figure 15 Chaotic Chess Bomb Usage Example After 11](#_Toc155097657)

# Appendix

## Appendix 1

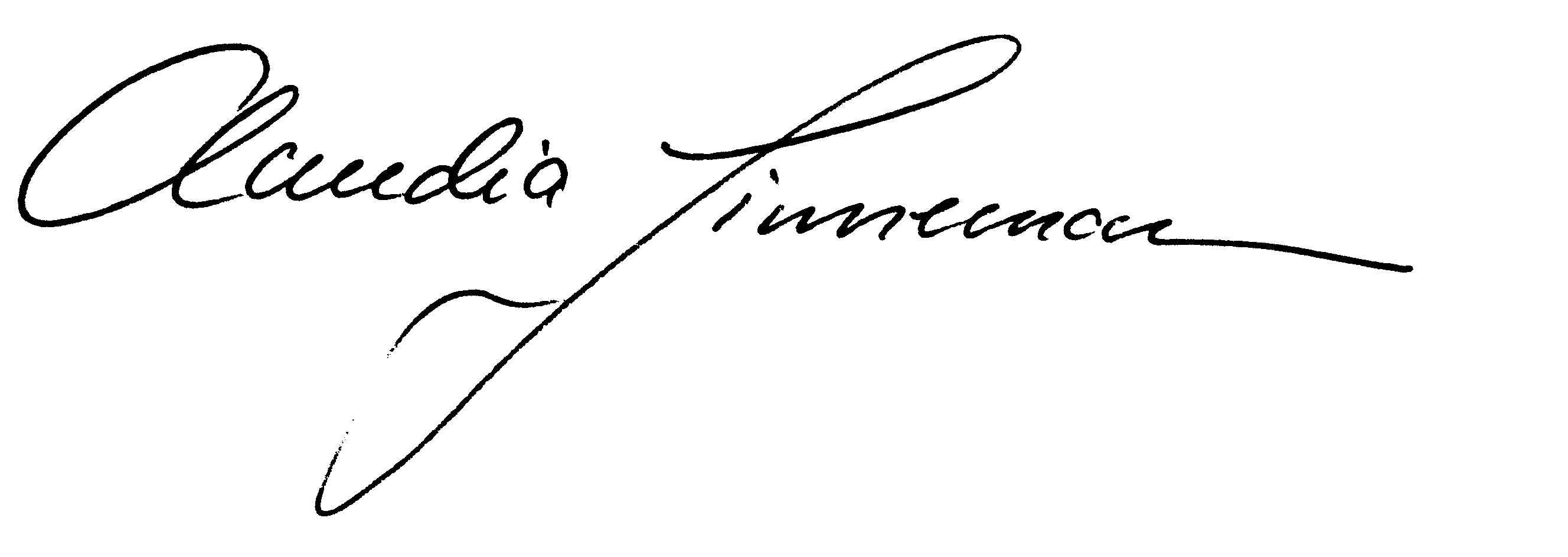


## Appendix 2



## Declaration of Authenticity

I hereby declare that the work submitted is my own and that all passages and ideas that are not mine have been fully and properly acknowledged.



Mels, 6.1.2020