**Title of the Paper   
Subtitle of the Paper**

****

**Matura Paper, Kantonsschule Sargans**

Attila Vizhanyo 4bNP

**Supervisor:**Herr Lukas Fehr

**Submitted on:**  
6 January 2020

# Preface

In the pursuit of writing my Matura paper, my objective was to combine two of my passions: chess and coding. I have been playing chess for a long time. My grandfather taught me how to play when I was little. He and my uncle played frequently against each other, and by watching them play, I was inspired to start to play as well. I never indulged in chess competitively, but I enjoy playing against my family and friends. Its simple rules and complicated strategies appealed to me. My second interest revolving my work is coding. Since both of my parents are programmers, I was introduced to the language of computers early on. Coding presented a distinct language that was straightforward and made perfect sense to me. I like its simplicity and logical way of thinking.

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. With my adjustments, my goal was to make chess more enjoyable for people that may not appreciate its strategic complexity. By changing the rules accordingly, I want to make the game less competitive and more fun.

# Table of Content

[Preface II](#_Toc454175648)

[Table of Content III](#_Toc454175649)

[1 Introduction 1](#_Toc454175650)

[1.1 Objectives and Guiding Questions 1](#_Toc454175651)

[1.2 Procedure and Method 1](#_Toc454175652)

[1.3 Structure of the Paper 1](#_Toc454175653)

[2 Original Chess Code 2](#_Toc454175654)

[3 Color Chess 2](#_Toc454175654)

[3.1 Rules of Color Chess 2](#_Toc454175655)

[3.2 Coding of Color Chess 2](#_Toc454175656)

[4 Chaotic Chess 3](#_Toc454175657)

[4.1 Rules of Chaotic Chess 3](#_Toc454175658)

[4.2 Coding of Chaotic Chess 3](#_Toc454175659)

[5 Elemental Chess 4](#_Toc454175660)

[5.1 Rules of Elemental Chess 4](#_Toc454175661)

[5.2 Coding of Elemental Chess 4](#_Toc454175662)

[6 Summary 5](#_Toc454175663)

[Bibliography IV](#_Toc454175664)

[Table of Figures V](#_Toc454175665)

[Appendix VI](#_Toc454175666)

[A. Appendix 1 VI](#_Toc454175667)

[B. Appendix 2 VII](#_Toc454175668)

[C. Declaration of Authenticity VIII](#_Toc454175669)

# Introduction

## Objectives and Guiding Questions

When beginning with my work, I had to decide in which coding language I wanted my project to be written in. Since at that time I was a beginner programmer, I chose Python.

The next question that came up regarding my project, was how many chess variants I wanted to program. The answer was not straight forwards, since I first had to figure out what my chess variations rules will be, to estimate how much effort it would take to code each variation. In the beginning, I presented two variations, “color chess” and “chaotic chess” to my supervisor. After he analyzed the two game-modes, I was told that he liked the idea, and I should come up with a third variation to fulfill the requirements of a diploma work. Subsequently, I figured out a third variation that involved around the four elements of the world, “elemental chess”.

When coming up with these variants of chess, I had to ask myself which aspects of chess do I want to modify, and which ones do I want to keep? I did not want to reinvent the game, therefore, I preserved the fundamentals characteristics of chess, by not changing the board, pieces, and movements. Instead, I fancied to add prominent features of video games to chess.

For the first variant I took inspiration from a modern-day game called “Splatoon”. The objective of the game is to cover as much area as possible with a color that is assigned to you. To achieve victory your team must color more area than the opponent team. This game concept is not unique however, there are many similar games that are based on the same idea. I determined that his game idea can be well adapted onto chess. By assigning a color to each square that a white piece moves onto, and another color for which squares a black player moves onto, the two players can compete for the control over the coloring of the squares on the chessboard.

Items are a core part of video games. Most games have some sort of item that does a special action inside of the game. In video games like Super Mario you have items all over the world that make you stronger. In chess however there aren’t any items you can interact with. So, my goal with my second variant was to implement four different items that could be picked up by chess pieces, to intensify the in-game experience.

With my third variant I wanted to incorporate the four elements of the world into chess, earth, water, air, and fire.

## Procedure and Method

## Structure of the Paper

# Original Chess Code

Process of original chess code

Ein Bild, das Text, Screenshot, Schrift, Design enthält.

Automatisch generierte BeschreibungIn the first step, the program sets the turn to white, since in the game of chess white always plays the first move. The program then proceeds by creating an eight-by-eight field that represent the chess board. Afterwards, the position is set up by adding the chess pieces to their corresponding starting location.

Next the program waits for an input of the user. This will be a click on a field that the user wants to move a piece from. The program must now check if the selected field does indeed have a white piece on it. By taking the value of the field, the program can determine if there is a piece on the selected field. If the field is empty, the user must select a field again. If the user succeeds in picking a field with a piece on it, it colors that field green, to indicate that it has been selected. Afterwards, the program must calculate every possible move with the selected piece and writes down coordinates to which fields the piece can travel to. These fields will be marked with a green circle to show the user, what moves are available to him.

Then the computer waits for the second input of the user that will let the computer know to which field to move the selected piece to. If the destination field corresponds to the possible moves of the selected piece, the program allows the move to be played. If not, the program asks the user to select a piece again and make another move.

Once the move was successful, the turns switch and black is asked to select one of his pieces.

# Color Chess

## Rules of Color Chess

## Coding of Chaotic Chess

„Unter wörtlichem Zitat wird ein Text originalgetreu wiedergegeben. Rechtschreibfehler oder eine veraltete Orthographie werden unverändert übernommen. Der zitierte Text steht in einem wörtlichen Zitat immer in Anführungszeichen, das einleitende Anführungszeichen steht unten und das beendende oben.“ (Metzger, 2010, p. 15)

Gemäss Metzger (2010, p. 20) sind wörtliche Zitate nur sehr sparsam einzusetzen. Sinnvoll sind wörtliche Zitate bei Definitionen oder bekannten Zitaten.

Bei einem sinngemässen Zitat stehen keine Anführungszeichen. Wenn der ganze Absatz aus einer Quelle stammt, steht die Quelle nach dem Punkt. Wird nur ein Satz zitiert steht die Quelle vor dem Punkt. Wenn möglich sind Primärquellen zu verwenden. Falls Sie also aus Wikipedia Informationen verwenden, versuchen Sie die Primärquelle herauszufinden. (Bonati & Hadorn, 2009, p. 20)

# Chaotic Chess

## Rules of Chaotic Chess

In the game of Chaotic Chess there are four items that modify the common rules of chess. These items are randomly distributed on the fields of the chess board. Pieces can interact with the item by stepping on the field that are occupied by the item.

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

**Bomb:**

Picking up the bomb induces an explosion of a three-by-three area. Every piece in that area is defeated and removed from the bord. This way clever sacrifices can be made by sacrificing your own pieces to take down more and stronger pieces of your opponent. The bomb can blow up every piece on the board, so checkmate can attain, by stepping on a bomb near the king. The king himself can step onto the bomb as well, which would lead to a victory of the enemy team. 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.

Example of bomb usage Example of a bomb explosion at the edge

**Ein Bild, das Brettspiel, Schachfigur, Spiele, Tabletopspiel enthält.

Automatisch generierte BeschreibungEin Bild, das Schachfigur, Brettspiel, Schach enthält.

Automatisch generierte BeschreibungEin Bild, das Schachfigur, Schach, Brettspiel, Tabletopspiel enthält.

Automatisch generierte BeschreibungCoin:**

The piece that stepped on the coin, gets converted into the piece that is resembled on the coin. To determine which piece is portraited by the coin, a random choice is made between knight, bishop, and rook. The king is the only piece that is not allowed to pick up the coin.

Ein Bild, das Schachfigur, Schach, Brettspiel enthält.

Automatisch generierte BeschreibungEin Bild, das Schachfigur, Brettspiel, Tabletopspiel, Spiele enthält.

Automatisch generierte Beschreibung Example of a coin usage

**Shield:**

The shield makes the piece that stepped on to the shield invincible for three rounds. The shield is anchored on its field and does not move with the piece that picked the shield up. Every piece can pick up the shield.

Example of a shield usage

Ein Bild, das Schachfigur, Brettspiel, Schach enthält.

Automatisch generierte BeschreibungEin Bild, das Brettspiel, Schach, Schachfigur, Spiele enthält.

Automatisch generierte Beschreibung

**Barrier:**

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

Jump over the barrier. Check mate with the barrier.

Ein Bild, das Brettspiel, Schachfigur, Schach enthält.

Automatisch generierte BeschreibungEin Bild, das Schachfigur, Brettspiel, Schach, Spiele enthält.

Automatisch generierte Beschreibung

## Coding of Chaotic Chess

**Basic properties of an item**

To explain what each item individually does; I will first describe what these items have in common. Because they all have the same underlying code for the generation, visualization, interaction and reappearance of an item.

Generating the position of items randomly

Ein Bild, das Schachfigur, Brettspiel, Tabletopspiel, Spiele enthält.

Automatisch generierte BeschreibungIf I wanted chaotic chess to be a balanced game, I had to generate the items equally on both sides. To make this happen I declared two list of buttons. A button represents a field on the chess board. The white buttons only featured buttons from rows 1-4 and the black buttons rows 5-8. With these two lists, I could now tell the program to randomly select a button from each list, and on each side an item would appear. However, the problem would occur that the program selects a button that is already occupied by a chess piece. This way the item would overtake the piece and delete it. Since I only wanted the items to spawn on empty fields, I had to tell the program to make a random selection of the empty fields on each side. Luckily in the original program two lists called “position\_white\_players” and “position\_black \_players” were available. These lists contained the pieces of white and black and most importantly, their positions. This way I could tell the program that it should randomly choose a button to put the item on, but if that button is already occupied by a chess piece, it should randomly choose another button to place the item on.

White button list

Black button list

  while True:

    bomb\_button = random.choice(button\_list)

    bomb\_pos = getButtonPosition(bomb\_button)

    if bomb\_pos not in players\_pos\_list and bomb\_pos != endbutton\_pos:

      break

Embodiment of an item

Once the item has found a button to be placed on it should store the buttons position, since we will need to fall back on it in the future. Afterwards the item should be visually displaying its location to let the user now where it is. For the sake of simplicity and testing I started off by representing the item with a unique letter and if the item was on the white side, it was upper-case if it was on the black side, it was lower-case. Items on the white side were labelled with “\_1” and items on the black side were labelled “\_2” at the end of the item’s name.

With the command button.config(text = x ) I could edit the text that appears on the button. Since we stored the position of the randomly selected button earlier, we can edit the button that the item is placed on.

BOMB\_1 = "O" # white side bomb

# Bomb\_button is the randomly selected button, where we will place our bomb item on

# We visualize our bomb by editing the buttons text to the bomb symbol

bomb\_button.config(text=BOMB\_1)

Picking up an item

Next, we want to let the program now if the user picked up an item. For this to work, we can use our “endbutton\_text” variable again. By saying that if the “endbutton\_text” is equal to the item symbol, it should know that the user stepped on the item. Inside of this condition, we will code what each item will do if it gets picked up.

  # If text of destination field and text of bomb is the same

  if endbutton\_text == BOMB\_1 or endbutton\_text == BOMB\_2:

    # Let the user now that he picked up the bomb

    action\_label.config(text = "YOU PICKED UP A BOMB!", fg = "black")

Regenerating position of items randomly

With the next step I wanted to make the game more random and chaotic, so I decided to change the position of the items after a certain number of rounds. To make this happened I had to delete the original items place if it already existed and choose another position for it. We should also note that if the item has already been picked up it should not delete its former place since the player already removed the item from the board by picking it up. However, this code does not apply to the barrier, since it’s impossible to pick up the barrier and we always have to delete it’s former position. In order to delete the former item, we can make use of button.config(text = “”). The (“”) indicate to the computer that it should display a string, which is simply a text. By not entering anything between the (“”) it tells the computer it should edit the buttons text to nothing.

**Bomb**

When the bomb is picked up it has to color the fields surrounding the bomb black. In order to achieve this 3-by-3 black area I took use of the nested for loop function.

**Coin**

One unique feature about the coin is that it generates with a random choice of text. To get a random piece represented on the coin, I had to use the random module by importing it with “import random”. Now I could tell the program to make a random choice between three different figures including bishop, knight, and rook.

COIN\_LIST = [BB, BN, BR]

coin\_symbol = random.choice(COIN\_LIST)

With this code our coin item will pick a random symbol in coin list and use that symbol to be represented on the coin. Next up the program has to know if the coin is picked up, which symbol is represented on it in order to transform the piece that stepped on the field into the piece on the coin. For this I used “if functions”. At first the program has to determine which player stepped on the coin. In the original chess code, there was already a function that determine which players move it is. So if the coin is picked up when white is on the turn, the code should use the white pieces to transform the piece into the coin. Then when the program nows which color stepped on the coin, it should read what symbol is on the coin. We can use the endbutton\_text again. So if the endbutton\_text is the rook and the turn is white, it should turn the piece that stepped on the coin into a white rook.

  elif player\_color == 'W':

    if endbutton\_text == BB:

      endbutton.config(text = WB)

      figure.name = 'WL'

    if endbutton\_text == BN:

      endbutton.config(text = WN)

      figure.name = 'WS'

    if endbutton\_text == BR:

      endbutton.config(text = WR)

      figure.name = 'WT'

However, there is the same problem that occurs like with the bomb. The program only replaces the look of the piece but not it’s properties. So, although our piece now transformed into the coin piece, it still moves and behaves like before stepping on the coin. Therefore we have to now tell our code that the piece that stepped on the coin should not only take the symbol of the coin piece, but also take the values and properties of coin piece.

**Shield**

**Barrier**

The last item is the barrier. With the help of the code that was already in the original chess program, the coding of this item was simpler. To make the barrier work I had to use the function “check\_if\_move\_legit”, which calculates if a move is possible and the variable “endutton\_text”, which stores the text of the button that the user clicked on the second time. Within the “check\_if\_move\_legit” function, I coded that if the “endbutton\_text” is equal to the barrier’s text, the program should tell the user he can’t step onto a barrier and not allow this move by returning “False”. This then leads to an error and the program ask the user to play another move.

  def check\_if\_move\_legit(self, ps, pe, all\_positions\_enemy, all\_positions\_own\_team, all\_possible\_moves\_enemy, turn):

    # if user chooses barrier as it's destination field,

    if endbutton\_text == barrier\_text:

      # tell the user it cannot step onto the barrier

      action\_label.config(text = "YOU CAN'T STEP ONTO A BARRIER", fg = "red")

      # makes the move impossible to make

      return False

# Chapter 4

## Subchapter 4

## Subchapter 4

# Summary

# Bibliography

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

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

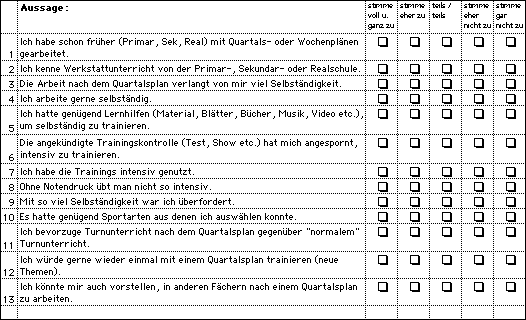
*Panorama*. (2015, December 15). Retrieved December 6, 2019, from Statistik Schweiz - Panorama: http://www.bfs.admin.ch/bfs/portal/de/index/themen/01/01/pan.html

# Table of Figures

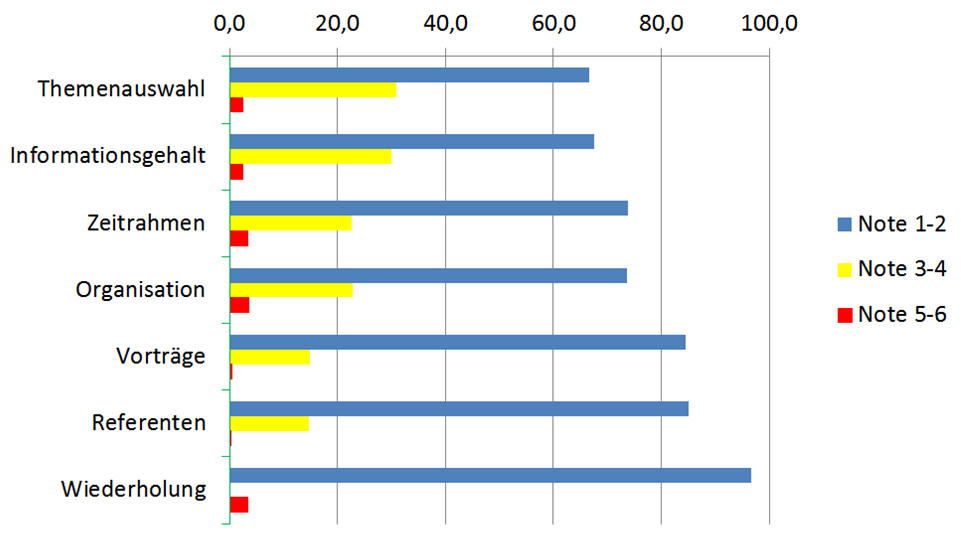
[Figure 1: Demographics of Switzerland in 1900 and 2014 3](file:///D:/Sargans%202016-2017/3-Info/9)%20Maturavorlage%20erstellen/Kurs%20MA-Vorlage%20Reuteler/Vorlage%20Maturaarbeit%20Englisch.docx#_Toc454176099)

# Appendix

## Appendix 1

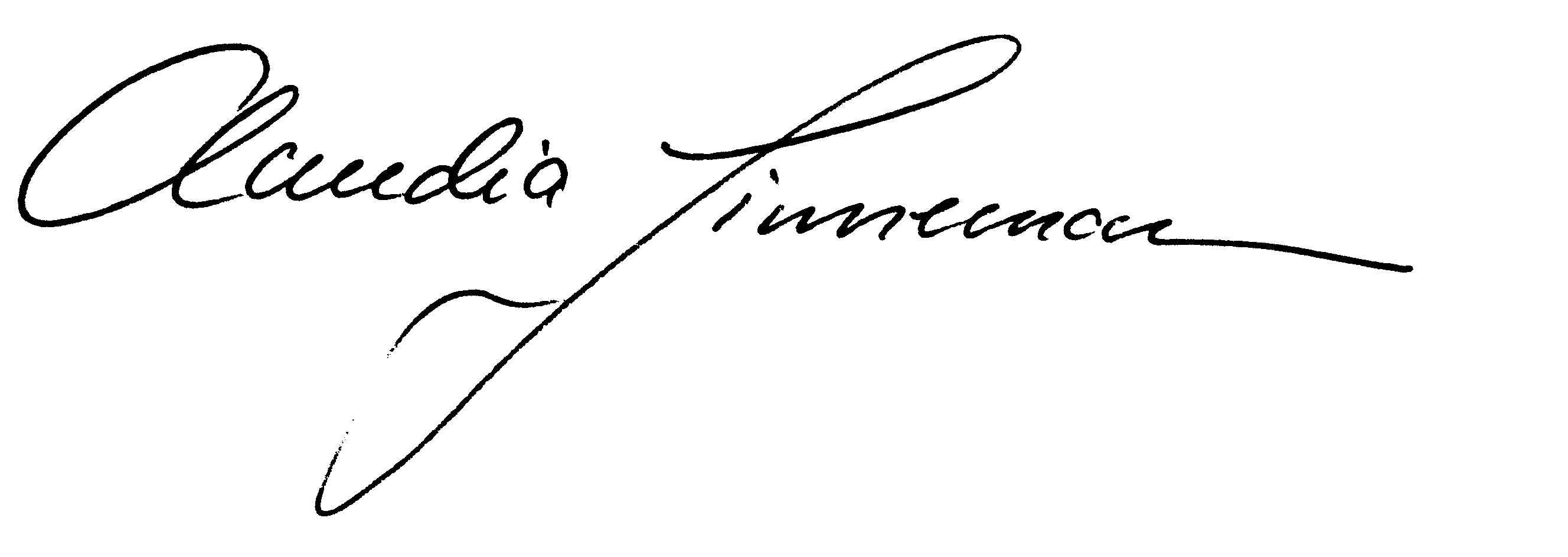


## Appendix 2



## 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