Logo

Description automatically generated

**NUMERICAL**

**CONTROL PROGRAM**

Gyarmathy Csaba

Technical University of Cluj-Napoca

2022-2023

Contents

[**Introduction** 3](#_Toc119658271)

[Project specification 3](#_Toc119658272)

[What is a CNC? 3](#_Toc119658273)

[G-code interpretation 4](#_Toc119658274)

[Objectives 5](#_Toc119658275)

[**Bibliographic study** 5](#_Toc119658276)

[CNC classification 5](#_Toc119658277)

[About the code interpreter 5](#_Toc119658278)

[Avoiding CNC crashing 6](#_Toc119658279)

[Coordinate systems 6](#_Toc119658280)

[**Design** 7](#_Toc119658281)

[Why Java? 7](#_Toc119658282)

[The Graphics library 7](#_Toc119658283)

[MVC Architecture 8](#_Toc119658284)

[Program design 9](#_Toc119658285)

[**Implementation** 10](#_Toc119658286)

[**Testing** 10](#_Toc119658287)

[**Conclusions** 10](#_Toc119658288)

[**Bibliography** 10](#_Toc119658289)

# **Introduction**

## Project specification

The purpose of this project is to design, implement and simulate a program which controls a CNC (**C**omputer **N**umerical **C**ontrol) machine.

The program should read the instructions from a file which contains a path (sequence of segments and arcs) and generate the commands corresponding to the code to move a cutting head in two directions (X and Y axis) across the working surface. The traced path should be displayed on the screen.

## What is a CNC?

Numerical control, often known as CNC or computer numerical control, is the automated computer-based control of machining equipment, such as drills, lathes, mills, grinders, routers, and 3D printers. Without a manual operator directly overseeing the machining operation, a CNC machine follows coded programming instructions to process a piece of material (metal, plastic, wood, ceramic, or composite) to match the given specifications.

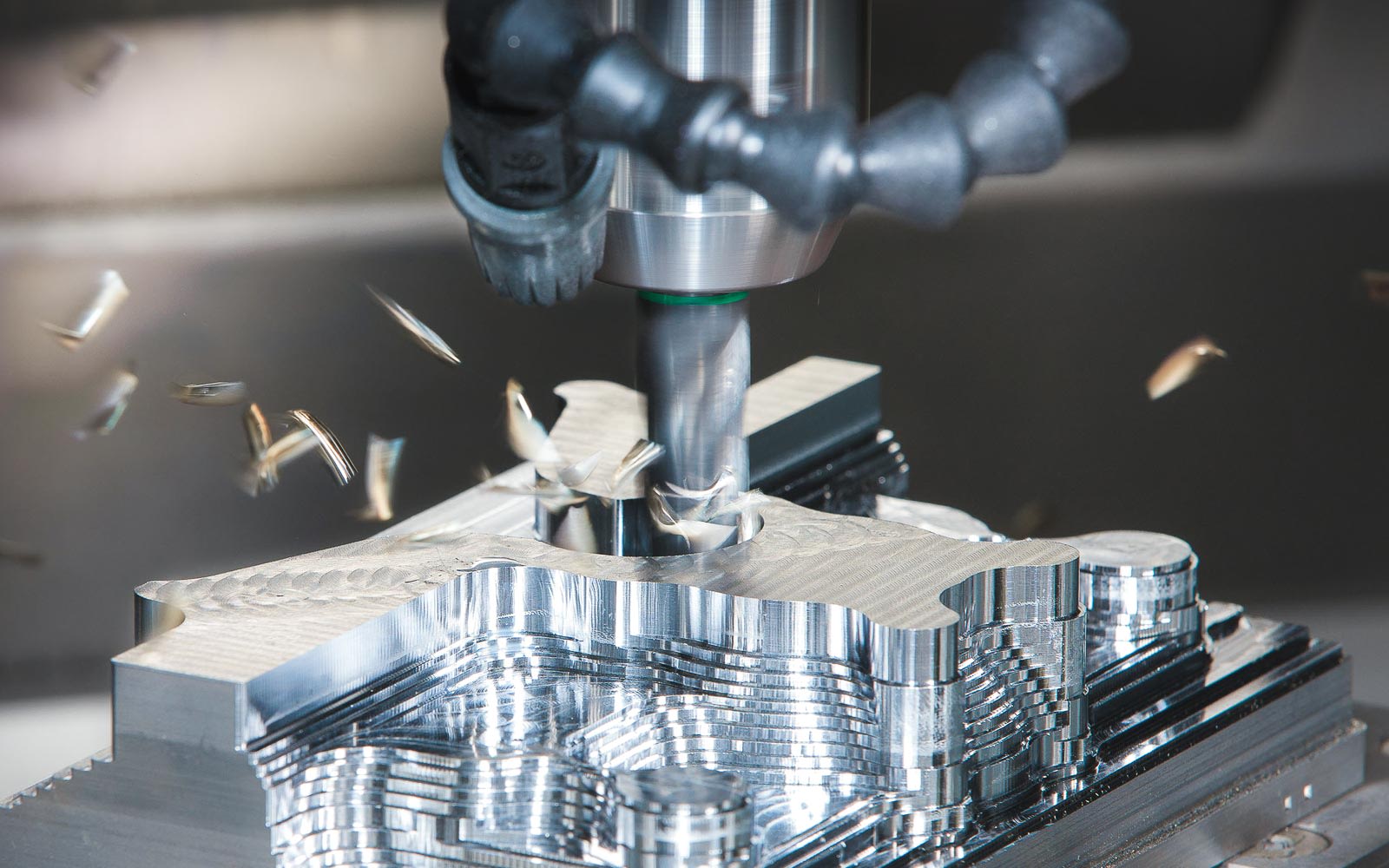


Figure 1.1. A CNC mill in action

A CNC machine is a motorized controllable tool or occasionally, a motorized manipulable platform, that is operated by a computer in accordance with precise input instructions. A CNC machine receives instructions in the form of a sequential program of machine control instructions like G-code (see Fig. 1.2) and M-code, which are then carried out. The program can be written manually by a human or, much more frequently, by CAD (Computer Aided Design) or CAM (Computer Aided Manufacturing) software that generates graphical designs.

## G-code interpretation

Our program will read G-code from a file and interpret it. The following is an example of a simplified code to help understand how it works.

G00 X0 Y0; //moves cutting head to initial position  
G01 Y160 F100; //linear interpolation, 100mm/min feed rate  
G02 X20 Y180 I20 J0; //clockwise radial move   
 //I and J are coordinates of the center  
 //current position of the head is the reference  
G01 X160;  
G02 X180 Y160 I0 J-20;  
G01 Y40;  
G02 X140 Y0 I-40 J0;  
G01 X0;

Diagram

Description automatically generated

Figure 1.2. Traced example of G-code

## Objectives

The program should be able to open and read information from a chosen file. This means that it must interpret what each command line means and be able to trace a line between the given coordinates or, in the case of a circle, an arc. The path the cutting head takes should be displayed on the user’s screen.

# **Bibliographic study**

## CNC classification

The machine tool is called "mother machine" in the sense that it is a machine that makes machines. As machine tools have advanced from manual machine tools to NC machines, these have become perfect in the role of mother machines with the improvement of accuracy and machining speed.

NC machine tools can be classified as **cutting machines** and **non-cutting machines**. A cutting machine means a machine that performs a removal process to make a finished part, milling machines, turning machines and EDM machines being good examples. In our case, which is a plasma cutter, we are talking about a cutting machine, quite literally, as it cuts through sheet metal to create the finished product. Non-cutting machine tools change the shape of the blank material by applying force. Press machines are good examples of this. In addition, robot systems for welding, cutting, and painting can be included in a broad sense.

## About the code interpreter

For this project, I will design and implement a code interpreter. The code interpreter is a software module, which translates the part program into internal commands for moving tools and executing auxiliary functions in a CNC system. And to do so, we need to understand the basics of the G-code the program will interpret. Some G-code commands are the following:

[G00 Rapid Motion Positioning]

[G01 Linear Interpolation Motion]

[G02 Circular Interpolation Motion-Clockwise]

[G03 Circular Interpolation Motion-Counterclockwise]

[G04 Dwell (Group 00) Mill]

[G10 Set offsets (Group 00) Mill]

[G12 Circular Pocketing-Clockwise]

[G13 Circular Pocketing-Counterclockwise]

To keep our project simple and relatively easy to understand, the interpreter will only recognize and execute the first 4 type of commands, as it can be seen in Fig. 1.2. We are limited to only a basic window that shows the trajectory of the cutting head, so this will benefit the interpreter greatly.

The interpreter could be considered as a simple task for the conversion of G/M codes to the CNC-understandable internal data structures. However, the design and implementation of the interpreter is a large and comprehensive task because programming rules or grammar described in a programming manual and an operating concept shown in an operation manual should be considered when developing the interpreter. Therefore, the interpreter is the representative indicator that shows the design concept and the functional aspect of a CNC and is a big part of CNC as it generally spends more than 50% of the total development time to develop the interpreter.

## Avoiding CNC crashing

One of the great benefits of not having an actual, physical, CNC machine is that we can easily avoid CNC **crashing**. In CNC, a “crash” occurs when the machine moves in such a way that is harmful to the machine, tools, or parts being machined, sometimes resulting in bending or breakage of cutting tools, accessory clamps, vises, and fixtures, or causing damage to the machine itself by bending guide rails, breaking drive screws, or causing structural components to crack or deform under strain.

## Coordinate systems

In CNC systems, a machine coordinate system, a workpiece coordinate system, and local coordinate systems are defined for convenience when editing a part program and handling machine tools.

A machine coordinate system is defined by setting a particular point of the machine tool as the origin of a coordinate system. A workpiece coordinate system is defined by setting a particular point on the workpiece as the origin to make editing a part program easier. That is, when editing a part program using one workpiece coordinate system, we can edit the part program by defining another coordinate system based on the workpiece coordinate system. We call this secondary coordinate system a “local coordinate system”.

For this project, we will use the window as the workpiece coordinate system. Since probably most of our test cases will begin from the origin, or will work only in the first quadrant, it would be wise to place the origin somewhere near the bottom left corner of the window.

# **Design**

## Why Java?

The project will be implemented in the **Java** programminglanguage. Java is a [high-level](https://en.wikipedia.org/wiki/High-level_programming_language), [class-based](https://en.wikipedia.org/wiki/Class-based_programming), [object-oriented](https://en.wikipedia.org/wiki/Object-oriented_programming) [programming language](https://en.wikipedia.org/wiki/Programming_language) that is designed to have as few implementation [dependencies](https://en.wikipedia.org/wiki/Dependency_(computer_science)) as possible. It is a [general-purpose](https://en.wikipedia.org/wiki/General-purpose_language) programming language intended to let [programmers](https://en.wikipedia.org/wiki/Programmer) *write once, run anywhere* ([WORA](https://en.wikipedia.org/wiki/Write_once,_run_anywhere)), meaning that [compiled](https://en.wikipedia.org/wiki/Compiler) Java code can run on all platforms that support Java without the need to recompile. Java applications are typically compiled to [bytecode](https://en.wikipedia.org/wiki/Java_bytecode) that can run on any [Java virtual machine](https://en.wikipedia.org/wiki/Java_virtual_machine) (JVM) regardless of the underlying [computer architecture](https://en.wikipedia.org/wiki/Computer_architecture). The [syntax](https://en.wikipedia.org/wiki/Syntax_(programming_languages)) of Java is similar to [C](https://en.wikipedia.org/wiki/C_(programming_language)) and [C++](https://en.wikipedia.org/wiki/C%2B%2B), but has fewer [low-level](https://en.wikipedia.org/wiki/Low-level_programming_language) facilities than either of them. The Java runtime provides dynamic capabilities (such as [reflection](https://en.wikipedia.org/wiki/Reflective_programming) and runtime code modification) that are typically not available in traditional compiled languages.

The syntax of Java is largely influenced by [C++](https://en.wikipedia.org/wiki/C%2B%2B) and [C](https://en.wikipedia.org/wiki/C_(programming_language)). Unlike C++, which combines the syntax for structured, generic, and object-oriented programming, Java was built almost exclusively as an object-oriented language. All code is written inside classes, and every data item is an object, with the exception of the primitive data types, (i.e. integers, floating-point numbers, [boolean values](https://en.wikipedia.org/wiki/Boolean_data_type" \o "Boolean data type), and characters), which are not objects for performance reasons.

## The Graphics library

The project will utilize Java’s **Graphics** library, which contains the Graphics class. It is the abstract base class for all graphics contexts that allow an application to draw onto components that are realized on various devices, as well as onto off-screen images.

A Graphics object encapsulates state information needed for the basic rendering operations that Java supports. This state information includes the following properties:

* The Component object on which to draw.
* A translation origin for rendering and clipping coordinates.
* The current clip.
* The current color.
* The current font.
* The current logical pixel operation function (XOR or Paint).
* The current XOR alternation color

Coordinates are infinitely thin and lie between the pixels of the output device. Operations that draw the outline of a figure operate by traversing an infinitely thin path between pixels with a pixel-sized pen that hangs down and to the right of the anchor point on the path. Operations that fill a figure operate by filling the interior of that infinitely thin path. Operations that render horizontal text render the ascending portion of character glyphs entirely above the baseline coordinate. All coordinates that appear as arguments to the methods of this Graphics object are considered relative to the translation origin of this Graphics object prior to the invocation of the method.

From this library the program will use the **drawLine** and **drawArc** methods, which do what you expect them to do.

* [drawLine](https://docs.oracle.com/javase/7/docs/api/java/awt/Graphics.html#drawLine(int,%20int,%20int,%20int))(int x1, int y1, int x2, int y2) - Draws a line, using the current color, between the points (x1, y1) and (x2, y2) in this graphics context's coordinate system.
* [drawArc](https://docs.oracle.com/javase/7/docs/api/java/awt/Graphics.html#drawArc(int,%20int,%20int,%20int,%20int,%20int))(int x, int y, int width, int height, int startAngle, int arcAngle) - Draws the outline of a circular or elliptical arc covering the specified rectangle.

These two methods will be a crucial part of the program, as they are the main commands the user can write in G-code. As mentioned before, the G01 command draws a line, and G02 or G03 commands draw arcs.

## MVC Architecture

To make the program more comprehensive and the code easy to understand, it will be build using the MVC architecture. **Model–view–controller** (**MVC**) is a software [architectural pattern](https://en.wikipedia.org/wiki/Architectural_pattern) commonly used for developing [user interfaces](https://en.wikipedia.org/wiki/User_interface) that divide the related program logic into three interconnected elements. This is done to separate internal representations of information from the ways information is presented to and accepted from the user.

Traditionally used for desktop [graphical user interfaces](https://en.wikipedia.org/wiki/Graphical_user_interface) (GUIs), this pattern also became popular for designing [web applications](https://en.wikipedia.org/wiki/Web_application). Popular programming languages have MVC frameworks that facilitate the implementation of the pattern. The role of each package is the following:

* **Model -** The central component of the pattern. It is the application's dynamic data structure, independent of the user interface. It directly manages the data, logic and rules of the application.
* **View –** This package contains any representation of information such as a window, button, textbox, chart, diagram or table. Multiple views of the same information are possible, like a bar chart for management and a tabular view for accountants.
* **Controller** - Accepts input and converts it to commands for the model or view.

In addition to dividing the application into these components, the model–view–controller design defines the interactions between them. The model is responsible for managing the data of the application. It receives user input from the controller. The view renders presentation of the model in a particular format. The controller responds to the user input and performs interactions on the data model objects. The controller receives the input, optionally validates it and then passes the input to the model.

## Program design

The user interface of the program will look somewhat, if not exactly, like the next concept:

Diagram

Description automatically generated

Figure 3.1. GUI concept of the CNC application

The components of the GUI will be:

* **Application Window** – The window of the application that contains all the other components. Can be minimized, maximized and closed any time. However, maximizing is not recommended, as the other components have fixed size and may not scale accordingly.
* **Load File Button** – Pressing this button will bring up the File Explorer for you to select the desired **text file** you wish the application to interpret. The file explorer should let you only select a **.txt** file. Upon successful loading a message “*file\_name successfully loaded!”* will be displayed.
* **Start/Stop Button** **–** Button to start the simulation once the selected file is loaded. Pressing the button without a loaded file will result in an error message “*File not leaded!”* being displayed. When the simulation starts, the text on the button should change to *“Stop”* and pressing it will stop the simulation. The text on the button will change back to *“Start”* again afterwards.
* **G-code Scroll Pane –** This pane will contain the G-code from the file so that the user can easily access and see what the code is.
* **Simulation area –** In this box you will be able to visualize the animated tracing of the given G-code commands, with a slight pause between the commands, and possibly displaying which is the current command and translating the command into human understandable language.

# **Implementation**

# **Testing**

# **Conclusions**

# **Bibliography**

<https://en.wikipedia.org/wiki/Numerical_control>

<https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller>

<https://docs.oracle.com/javase/7/docs/api/java/awt/Graphics.html>

[Suk-Hwan Suh](https://www.google.ro/search?tbo=p&tbm=bks&q=inauthor:%22Suk-Hwan+Suh%22), [Seong Kyoon Kang](https://www.google.ro/search?tbo=p&tbm=bks&q=inauthor:%22Seong+Kyoon+Kang%22), [Dae-Hyuk Chung](https://www.google.ro/search?tbo=p&tbm=bks&q=inauthor:%22Dae-Hyuk+Chung%22), [Ian Stroud](https://www.google.ro/search?tbo=p&tbm=bks&q=inauthor:%22Ian+Stroud%22), **Theory and Design of CNC Systems**, 2008, Springer Science & Business Media, [Theory and Design of CNC Systems - Suk-Hwan Suh, Seong Kyoon Kang, Dae-Hyuk Chung, Ian Stroud - Google Books](https://books.google.ro/books?hl=en&lr=&id=c_-3TxZlnpMC&oi=fnd&pg=PR19&dq=cnc&ots=PZh_4YOSpI&sig=1zTNdyA-FJzyBw6ExTSvM6sGavs&redir_esc=y#v=onepage&q&f=false)