

OENG1207 – Digital Fundamentals

INDIVIDUAL PROJECT REPORT

Unit Converter

TEACHER

Dr. Alexandru C Fechete

STUDENT

Nguyen Quoc Hoang
s3697305

PRACTICAL SESSION

Tuesday, 13:00 – 15:00

SUBMISSION DUE DATE

September 02, 2020

I. Background and Design

a. Program requirements and purpose

This Unit Converter is the final individual project of the course Digital Fundamentals taught by Dr. Alexandru C Fechete. The program's main functionality is to allow the user to choose what unit type and system they want to convert from, type in a number and the program will perform the conversion to its corresponding unit. The program consists of a window with other thoughtful and neatly designed graphical elements that will deliver an intuitive experience to the user when using the program.

b. Initial GUI layout design

The program has received an upgrade from a Command Line Interface to a Graphical User Interface to be more user-friendly since the majority of users are more familiar with buttons and pop-up windows rather than command lines.

Initially, I tried out multiple sketches on paper for the program's wireframe to see what design would make sense and what components it should have:

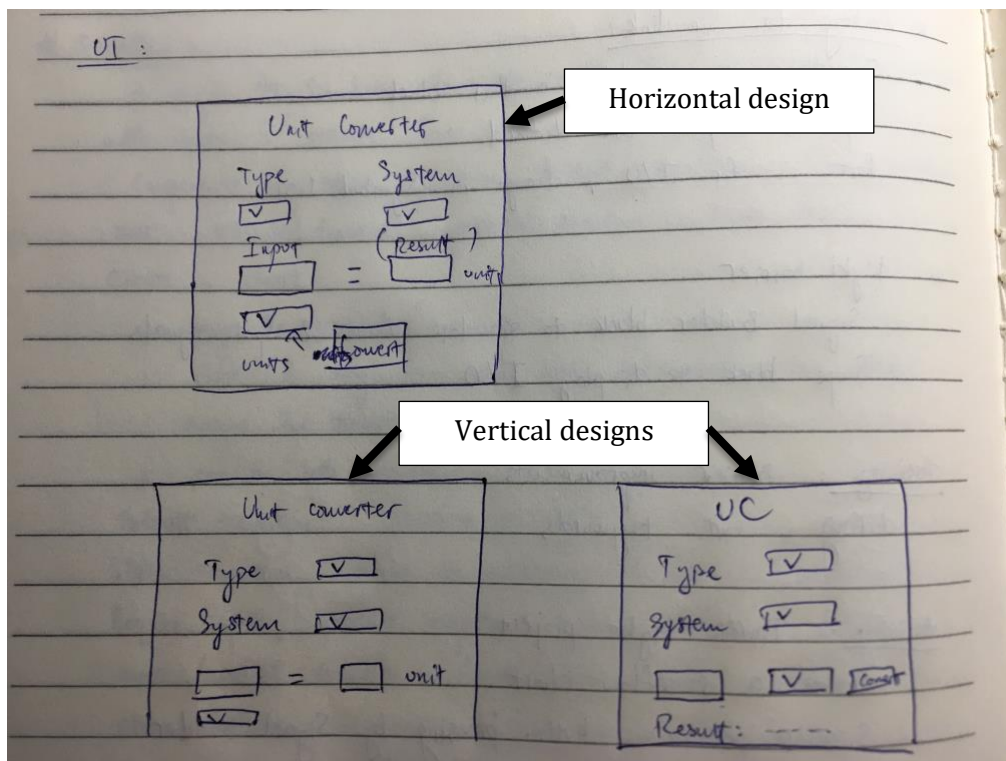


Figure 1: Initial UI sketches

After sketching out the design, I decided that the program should basically have 2 list boxes to allow the user to choose the unit types and systems; a dropdown list to choose the unit; 1 numeric field to get the input and 1 value field to display the output along with a button to perform the conversion.

c. Program structure and Algorithm design

Steps to use the program:

1. Open up the Unit Converter program
2. Choose Unit Type (Temperature, Length and Distance and Mass)
3. Choose System (Metric or Imperial)
4. Choose the available units they want to convert from in the dropdown list
5. Type in a number in the “Input Value” field
6. Click on “Convert!” button
7. The result will be displayed in the “Converted Value” box with the associated unit below it

Algorithm design:

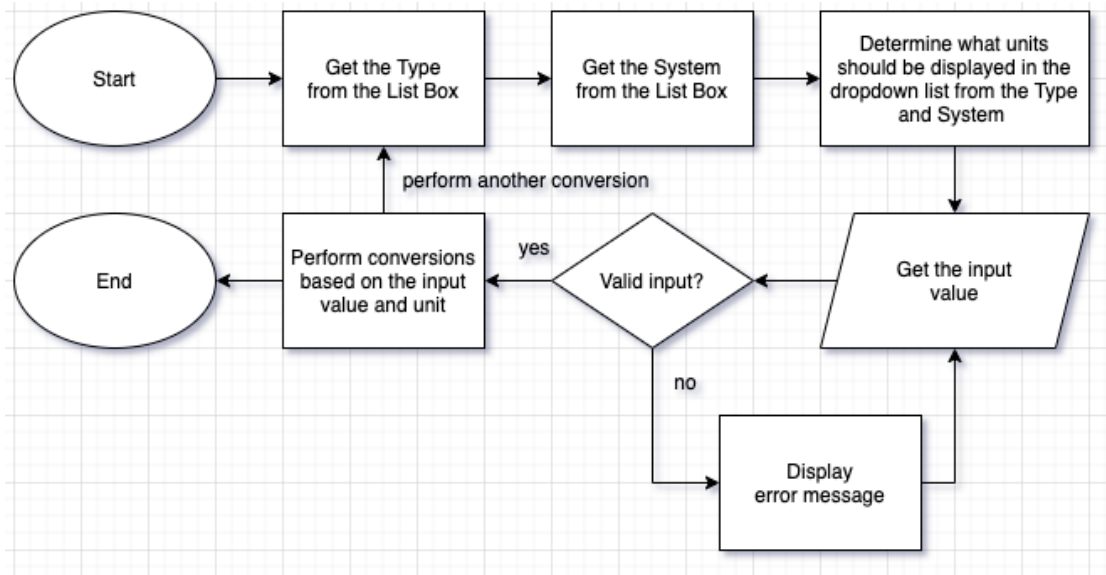


Figure 2: Algorithm design

The unit drop down list items will be decided and changed accordingly based on the type and system list boxes that the user chooses. If the user changes either a type or a system, the unit items will also be updated. Then the input will be validated before

it can be converted to the associated unit, otherwise a red error message will show up, telling the user what has been input wrong.

Program structure:

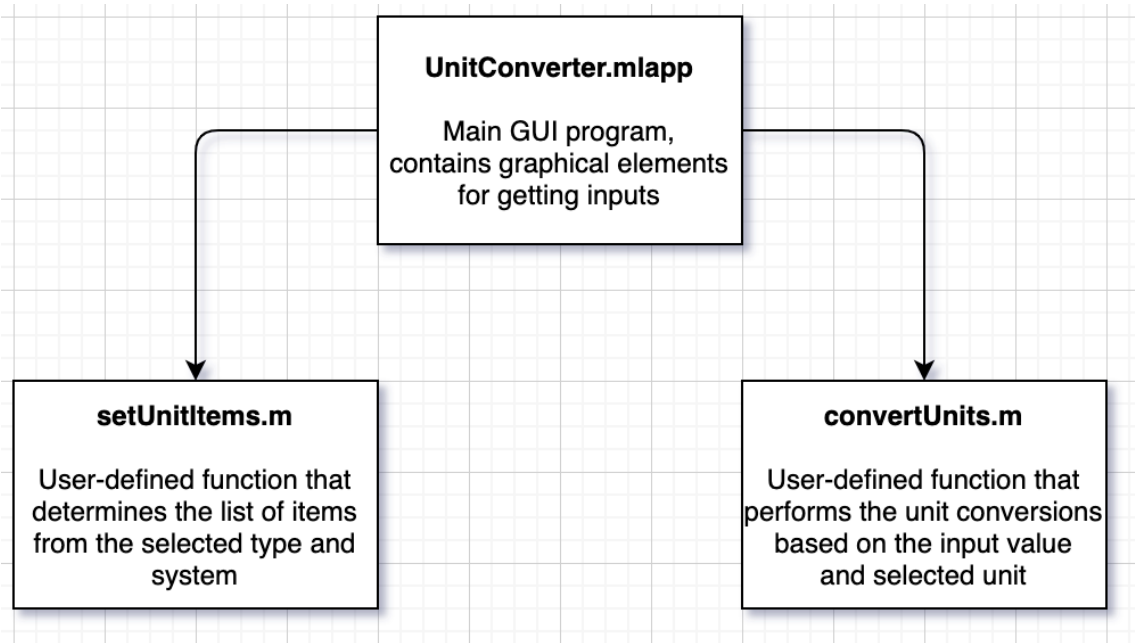


Figure 3: Program structure

The structure for Unit Converter program consists of 3 main files. The **UnitConverter.mlapp** file contains the main program with a pop-up window with graphical elements for getting the inputs from the user. This file is going to call **setUnitItems.m** file, which is a user-defined function each time the user chooses a different type and system and it will update the list of units accordingly. The main file also calls another user-defined function file: **convertUnits.m** to perform the unit conversions based on the input number and unit that the user has selected. This file will return the converted value with its corresponding unit along with a possible error message if the input is invalid.

II. Solution and Testing

a. Behavior table

Table 1 - Testing for Unit Converter program

| Type | Input Value | Output Value | Verified Answer |
|---|-------------|---|-----------------|
| Temperature | | | |
| $^{\circ}F \rightarrow ^{\circ}C$ | 123 | 50.5556 | 50.556 |
| $^{\circ}C \rightarrow ^{\circ}F$ | 300 | 572 | 572 |
| All Conversations | | | Correct |
| Length & Distance | | | |
| cm \rightarrow inch | 40 | 15.748 | 15.748 |
| inch \rightarrow cm | 50 | 127 | 127 |
| m \rightarrow feet | 12 | 39.3701 | 39.37 |
| feet \rightarrow m | 21 | 6.4008 | 6.4008 |
| km \rightarrow miles | 4 | 2.4855 | 2.48548 |
| miles \rightarrow km | 19 | 30.5775 | 30.578 |
| All Conversations | | | Correct |
| Mass | | | |
| grams \rightarrow ounces | 23 | 0.8113 | 0.811301 |
| ounces \rightarrow grams | 5 | 141.7475 | 141.748 |
| kg \rightarrow pounds | 74 | 163.1393 | 163.142 |
| pounds \rightarrow kg | 180 | 81.648 | 81.6466 |
| tonne (met) \rightarrow ton (imp) | 69 | 67.9134 | 67.9103 |
| ton (imp) \rightarrow tonne (met) | 96 | 97.536 | 94.5405 |
| All Conversation | | | Correct |
| <u>Erroneous inputs</u> | | | |
| Negative mass | -20 kg | "Error: Mass must be a positive number" | |
| Negative length | -6.5 ft | "Error: Length must be a positive number" | |
| Wrong format | 26 mikez | "Value must be numeric" | |
| The verified answers are from online: https://www.unitconverters.net/ | | | |

b. Program's graphical user-interface

1. Starting up the program, you will be welcomed by a clean and user-friendly design:

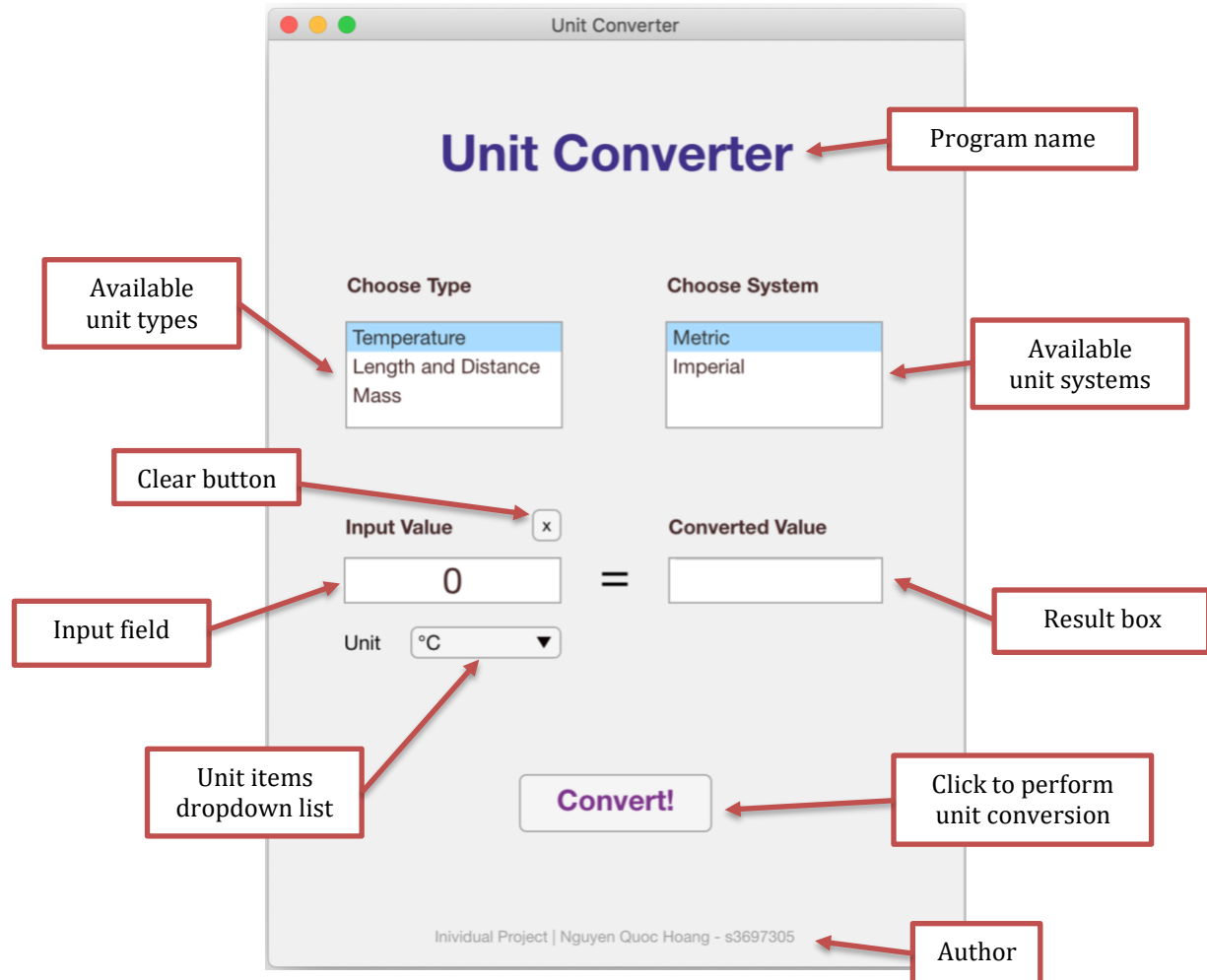
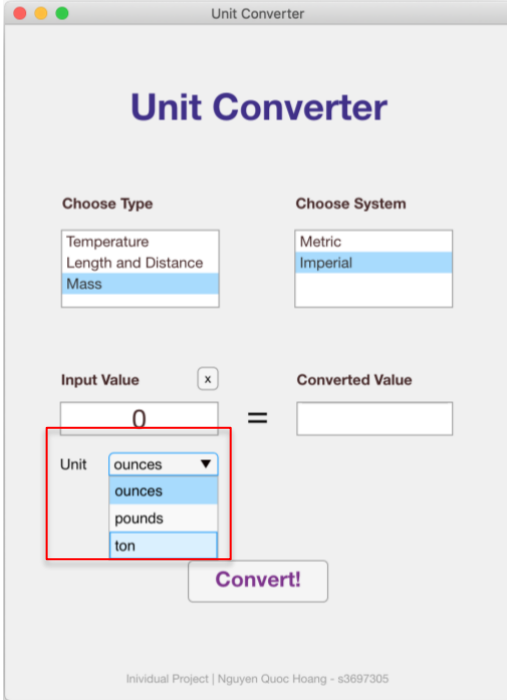


Figure 4: Program design

2. To convert a unit, select the its type and system. Select the unit you want to convert from from the dropdown list:

An example shown here for the imperial units for mass are:
ounces, pounds and ton



Unit Converter

Choose Type

- Temperature
- Length and Distance
- Mass

Choose System

- Metric
- Imperial

Input Value

Unit

- ounces
- ounces
- pounds
- ton

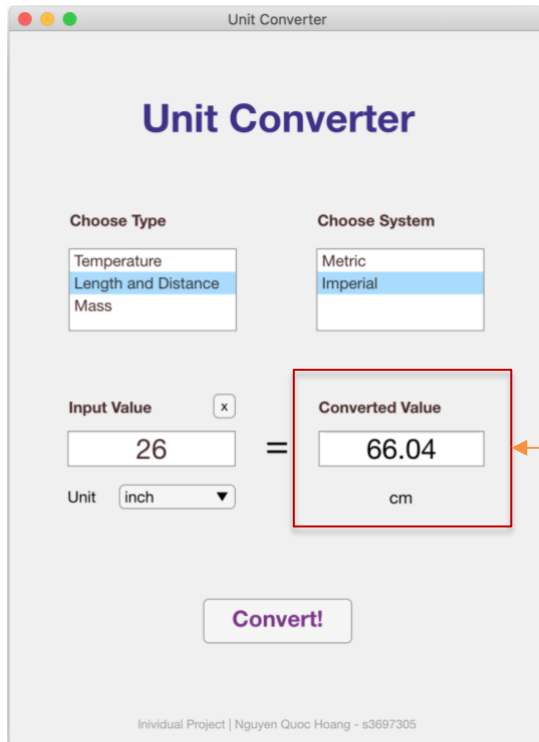
Converted Value

Convert!

Individual Project | Nguyen Quoc Hoang - s3697305

Figure 5: Unit dropdown items

3. Enter the number you want to convert in the input value field and click on **Convert!** button to perform conversion:



Unit Converter

Choose Type

- Temperature
- Length and Distance
- Mass

Choose System

- Metric
- Imperial

Input Value

Unit

Converted Value

cm

Convert!

Individual Project | Nguyen Quoc Hoang - s3697305

A sample conversion of 26 inch.
The associated metric unit for inch
is centimeter, therefore we got
66.04 cm for the result.

Figure 6: A successful conversion

4. If the input is invalid, the program will display an error message in red with the error code “err”:

The screenshot shows the 'Unit Converter' application window. The 'Choose Type' dropdown is set to 'Mass', and the 'Choose System' dropdown is set to 'Metric'. The 'Input Value' field contains '-20', and the 'Unit' dropdown is set to 'kg'. The 'Converted Value' field displays 'err'. A red error message, 'Error: Mass must be a positive number', is shown in a green-bordered box below the input field. A 'Convert!' button is visible at the bottom. Two callout boxes with arrows point to the 'err' in the converted value field (labeled 'Error code') and the red error message (labeled 'Error message'). A small 'x' button is next to the input field. The footer text reads 'Individual Project | Nguyen Quoc Hoang - s3697305'.

Figure 7: Invalid input

5. If you want to convert another number, the clear button (x) will help you clean up everything and get ready for a new input:

The screenshot shows the 'Unit Converter' application window after clearing. The 'Choose Type' dropdown is set to 'Temperature', and the 'Choose System' dropdown is set to 'Metric'. The 'Input Value' field contains '0', and the 'Unit' dropdown is set to '°C'. The 'Converted Value' field is empty. A 'Convert!' button is visible at the bottom. A small 'x' button is next to the input field. The footer text reads 'Individual Project | Nguyen Quoc Hoang - s3697305'.

Figure 8: All fields have been cleared up

III. Conclusion and Future Work

a. Program performance

After verifying the output values from my Unit Converter to another source from the internet and running through multiple test cases, the program is currently solid enough that it can handle all the errors pretty well. It can point out what might have been typed in wrong and tell the user where to correct it. More importantly, the program never crashes. In conclusion, not only the program has fulfilled all the requirements of the task well, but also with an aesthetic design along with additional functionalities such as the clear button delivers a wonderful user experience.

b. Future work

At the current state, the program has fulfilled all the requirements and operates well without crashing, there is little to no weaknesses to the core of the program. However, the units are still limited to only 3 types: temperature, length and distance and mass so more types of unit could be added in the future. Another extra feature is to make this program available to users in the form of a standalone desktop or mobile app, or potentially a web app. Currently it is still a MATLAB app, which requires user to have MATLAB installed in order to run the program. Packaging and distributing the program to different platform makes it more accessible to the majority of users.