./

Waveform Drawing Using Qt/QML

Contents

[Contents 2](#_Toc59453917)

[Problem Statement 4](#_Toc59453918)

[Requirement Specifications 4](#_Toc59453919)

[Methodologies 5](#_Toc59453920)

[Design 8](#_Toc59453921)

[Weekly tasks 10](#_Toc59453922)

[Week 1 10](#_Toc59453923)

[QT 10](#_Toc59453924)

[Week 2 20](#_Toc59453925)

[C++ Assignment Problem 20](#_Toc59453926)

[Waveform drawing using QT 21](#_Toc59453927)

[Week 3 22](#_Toc59453928)

[QML 22](#_Toc59453929)

[QT QML Integration 27](#_Toc59453930)

[Waveform drawing using QML 29](#_Toc59453931)

[Week 4 30](#_Toc59453932)

[Shadow Project Implementation 30](#_Toc59453933)

List of Tables

[Table 1: High level requirements 4](#_Toc59183527)

[Table 2: Low level requirements 5](#_Toc59183528)

**List of Figures**

[Figure 1: MVVM Architecture 8](#_Toc59618845)

[Figure 2: Color changer 13](#_Toc59618846)

[Figure 3: Digital clock 14](#_Toc59618847)

[Figure 4: Login form 14](#_Toc59618848)

[Figure 5: Login form 15](#_Toc59618849)

[Figure 6: Login form 15](#_Toc59618850)

[Figure 7: Login form 16](#_Toc59618851)

[Figure 8: Login form 16](#_Toc59618852)

[Figure 9: Number panel 17](#_Toc59618853)

[Figure 10: Number panel 17](#_Toc59618854)

[Figure 11: Number panel 18](#_Toc59618855)

[Figure 12: Simple Calculator 18](#_Toc59618856)

[Figure 13: Simple Calculator 19](#_Toc59618857)

[Figure 14: Simple Calculator 19](#_Toc59618858)

[Figure 15: String Operations 20](#_Toc59618859)

[Figure 16: String Operations 21](#_Toc59618860)

[Figure 17: String Operations 21](#_Toc59618861)

[Figure 18: String Operations 22](#_Toc59618862)

[Figure 19: Temperature and Volume 22](#_Toc59618863)

[Figure 20: Temperature and Volume 23](#_Toc59618864)

[Figure 21: Assignment Problem 24](#_Toc59618865)

[Figure 22: Waveform drawing using QT 25](#_Toc59618866)

[Figure 23: Snake Game 27](#_Toc59618867)

[Figure 24: Snake Game 28](#_Toc59618868)

[Figure 25: Demo Application 29](#_Toc59618869)

[Figure 26: Animation Application 30](#_Toc59618870)

[Figure 27: QT-QML Integration 31](#_Toc59618871)

[Figure 28: Waveform drawing using QML 33](#_Toc59618872)

[Figure 29: Demo UI 37](#_Toc59618873)

[Figure 30: Browse File 38](#_Toc59618874)

[Figure 31: Plotting Waveform 39](#_Toc59618875)

[Figure 32: Plotting Waveform 39](#_Toc59618876)

[Figure 33: Change Color 40](#_Toc59618877)

[Figure 34: Change Color 40](#_Toc59618878)

[Figure 35: Change Thickness 41](#_Toc59618879)

[Figure 36: Change Thickness 42](#_Toc59618880)

[Figure 37: Amplitude 43](#_Toc59618881)

[Figure 38: Zoom-In, Zoom-Out 44](#_Toc59618882)

[Figure 39: Zoom-In, Zoom-Out 44](#_Toc59618883)

[Figure 40: Localization Workflow 45](#_Toc59618884)

# **Problem Statement**

This application development is to get an understanding of C++ features and Qt/QML features. The main aim of the project is to draw multiple waveform with given multiple input file. Change waveform color, thickness, amplitude, scaling and sweep speed.

# **Requirement Specifications**

**High level Requirements**

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Type | Requirement Description | Testable |
| SW\_01 | Software | User shall be able to draw single/multiple waveforms by providing input data file. | Yes |
| SW\_02 | Software | User shall be able to control the waveform sweep speed. | Yes |
| SW\_03 | Software | User shall be able to control the waveform sweep speed. | Yes |
| SW\_04 | Software | User shall be able to change the color of the waveform. | Yes |
| SW\_05 | Software | User shall be able to change the thickness of the waveform. | Yes |
| SW\_06 | Software | User shall be able to apply localization for the textual contents. | Yes |

Table 1: High level requirements

**Low level Requirements**

|  |  |  |  |
| --- | --- | --- | --- |
| ID | Type | Requirement Description | Testable |
| SW \_01 | Software | The application shall be able to portray single/multiple waveforms, provided with input data file. | Yes |
| SW \_02 | Software | The application shall portray waveforms with the different sweep speed. | Yes |
| SW \_03 | Software | The application shall be able to zoom-in/zoom-out waveform. | Yes |
| SW \_04 | Software | The application shall support localization of textual content in different languages. | Yes |
| SW \_05 | Software | The application shall be able to generate log file for unintended exceptions. | Yes |
| SW \_06 | Software | The application shall include singleton standard design pattern. | Yes |

Table 2: Low level requirements

# **Methodologies**

* **MVVM (Model View View-Model)**

MVVM architecture offers two-way data binding between view and view-model. It also helps you to automate the propagation of modifications inside View-Model to the view. The view-model makes use of observer pattern to make changes in the view-model.

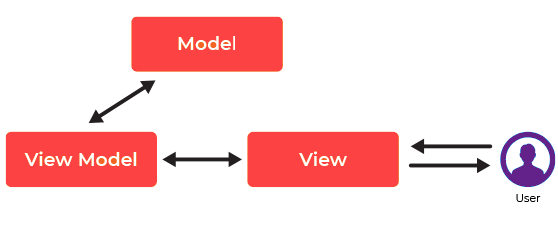


Figure 1: MVVM Architecture

**Model**

The model stores data and related logic. It represents data that is being transferred between controller components or any other related business logic.For example, a Controller object will retrieve the student info from the school database. It manipulates data and sends it back to the database or use it to render the same data.

**View**

The View stands for UI components like HTML, CSS, jQuery, etc. In MVVC pattern view is held responsible for displaying the data which is received from the Controller as an outcome. This View is also transformed Model (s) into the User Interface (UI).

**View Model**

The view model is responsible for presenting functions, commands, methods, to support the state of the View. It is also accountable to operate the model and activate the events in the View.

* **Solid Principles**

SOLID is a mnemonic device for 5 design principles of object-oriented programs (OOP) that result in readable, adaptable, and scalable code. SOLID can be applied to any OOP program.

The 5 principles of SOLID are:

Single-responsibility principle

Open-closed principle

Liskov substitution principle

Interface segregation principle

Dependency inversion principle

1. Single-responsibility principle:

* The single-responsibility principle (SRP) states that each class, module, or function in your program should only do one job. In other words, each should have full responsibility for a single functionality of the program. The class should contain only variables and methods relevant to its functionality
* Classes can work together to complete larger complex tasks, but each class must complete a function from start to finish before it passes the output to another class

1. Open-closed principle:

* The open-closed principle (OCP) calls for entities that can be widely adapted but also remain unchanged. This leads us to create duplicate entities with specialized behavior through polymorphism.
* Through polymorphism, we can extend our parent entity to suit the needs of the child entity while leaving the parent intact
* Our parent entity will serve as an abstract base class that can be reused with added specializations through inheritance. However, the original entity is locked to allow the program to be both open and closed

1. Liskov substitution principle

* The Liskov substitution principle (LSP) is a specific definition of a subtyping relation created by Barbara Liskov and Jeannette Wing. The principle says that any class must be directly replaceable by any of its subclasses without error.
* In other words, each subclass must maintain all behavior from the base class along with any new behaviors unique to the subclass. The child class must be able to process all the same requests and complete all the same tasks as its parent class.

1. Interface segregation principle

* The interface segregation principle (ISP) requires that classes only be able to perform behaviors that are useful to achieve its end functionality. In other words, classes do not include behaviors they do not use.
* This relates to our first SOLID principle in that together these two principles strip a class of all variables, methods, or behaviors that do not directly contribute to their role. Methods must contribute to the end goal in their entirety.

1. Dependency inversion principle

* The dependency inversion principle (DIP) has two parts

1. High-level modules should not depend on low-level modules. Instead, both should depend on abstractions (interfaces)
2. Abstractions should not depend on details. Details (like concrete implementations) should depend on abstractions.

# **Design**

**Structural Diagrams**

UML Structural diagrams depict the elements of a system that are independent of time and that convey the concepts of a system and how they relate to each other. The elements in these diagrams resemble the nouns in a natural language, and the relationships that connect them are structural or semantic relationships.

These static parts are represented by classes, interfaces, objects, components, and nodes. The four structural diagrams are −

* Class diagram
* Object diagram
* Component diagram
* Deployment diagram

**Behavioral Diagrams**

Any system can have two aspects, static and dynamic. So, a model is considered as complete when both the aspects are fully covered. Behavioral diagrams basically capture the dynamic aspect of a system. Dynamic aspect can be further described as the changing/moving parts of a system.

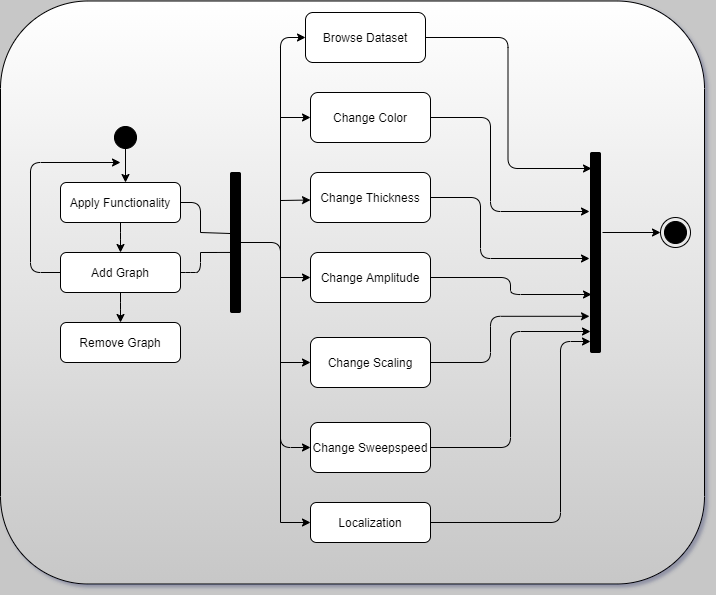
UML has the following five types of behavioral diagrams −

* Use case diagram
* Sequence diagram
* Collaboration diagram
* State chart diagram
* Activity diagram

**Activity Diagram**

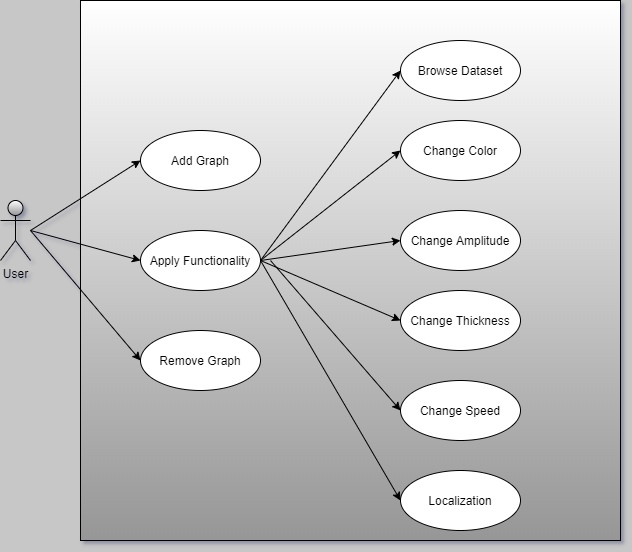
Activity diagram describes the flow of control in a system. It consists of activities and links. The flow can be sequential, concurrent, or branched. Activities are nothing but the functions of a system. Numbers of activity diagrams are prepared to capture the entire flow in a system.

Activity diagrams are used to visualize the flow of controls in a system. This is prepared to have an idea of how the system will work when executed.

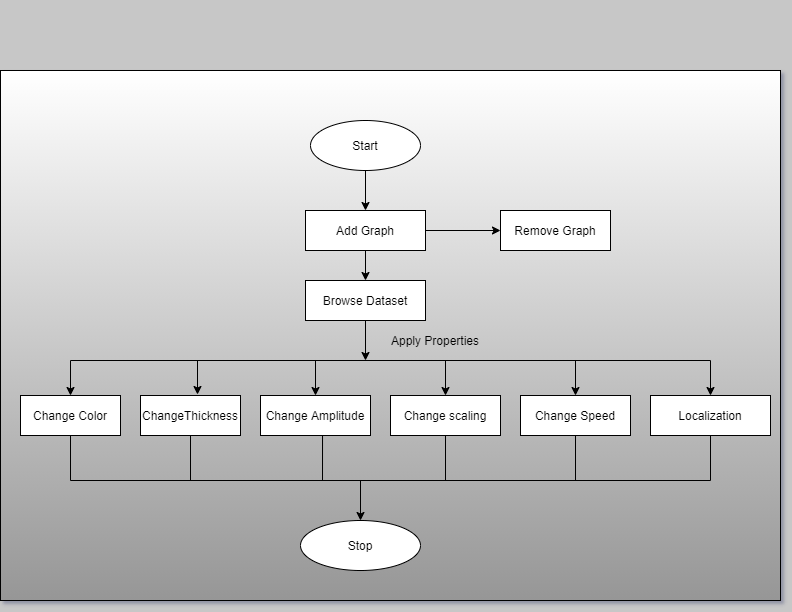


**Use Case Diagram**

Use case diagrams are a set of use cases, actors, and their relationships. They represent the use case view of a system. A use case represents a functionality of a system. Hence, use case diagram is used to describe the relationships among the functionalities and their internal/external controllers. These controllers are known as actors.



**Flow Chart**

****

# **Weekly tasks**

## **Week 1**

### QT

Qt (pronounced as "cute", not "cu-tee") is a cross-platform framework that is usually used as a graphical toolkit, although it is also very helpful in creating CLI applications. It runs on the three major desktop OSes, as well as on mobile OSes, such as Symbian, Nokia Belle, Meego Harmattan, MeeGo or BB10, and on embedded devices. Ports for Android (Necessitas) and iOS are also in development.

Qt has an impressive collection of modules, including

* **QtCore**, a base library that provides containers, thread management, event management, and much more
* **QtGui** and **QtWidgets**, a GUI toolkit for Desktop, that provides a lot of graphical components to design applications.
* **QtNetwork**, that provides a useful set of classes to deal with network communications
* **QtWebkit**, the webkit engine, that enable the use of web pages and web apps in a Qt application.
* **QtSQL**, a full featured SQL RDBM abstraction layer extensible with own drivers, support for ODBC, SQLITE, MySQL and PostgreSQL is available out of the box
* **QtXML**, support for simple XML parsing (SAX) and DOM
* **QtXmlPatterns**, support for XSLT, XPath, XQuery and Schema validation

**Applications implemented in QT**

1. **Color changer:**

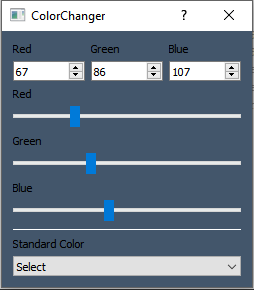


Figure 2: Color changer

Color changer is an application where we used spinbox and slider of RGB values separately and respective value’s color is reflecting on the background color. Also we can choose color from combox for background color.

1. **Digital clock**

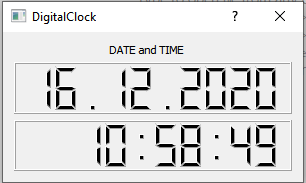


Figure 3: Digital clock

This is the Digital clock application where we used timer and LCD display to show the date and respective time.

1. **Login Form**

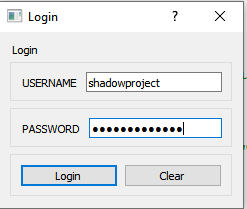


Figure 4: Login form

This is the login form application where username and password fields are there and user can login after providing right credentials.

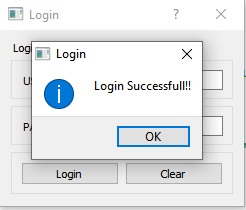


Figure 5: Login form

If entered username and password is correct then user will get this message box

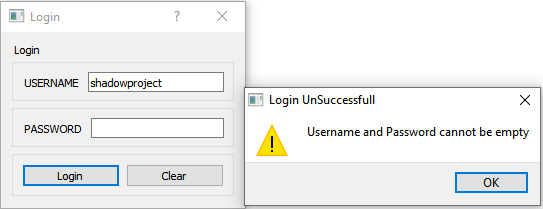


Figure 6: Login form

If any of the field is empty then user will get above message box

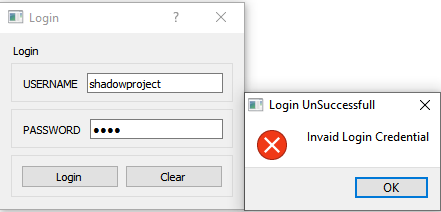


Figure 7: Login form

If username or password is incorrect then user will get above message box

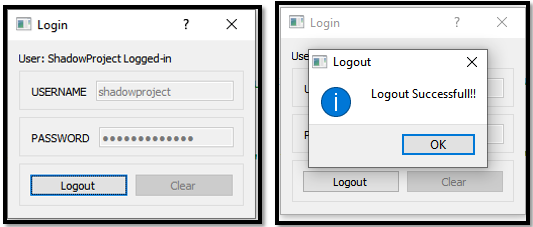


Figure 8: Login form

If login is successful then username and password field will get disable and after clicking on Logout button user will be able to enter those fields again.

1. **Number panel**

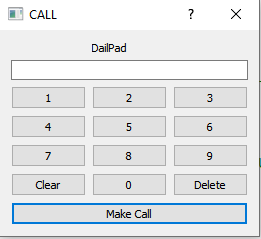


Figure 9: Number panel

This is the number panel application where we created panel to display dialled number after appending digits to it.

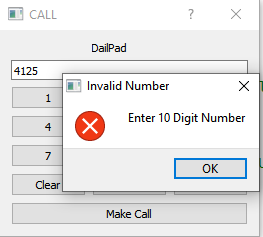


Figure 10: Number panel

If user entered number length is less than 10 then the above message box will get displayed.

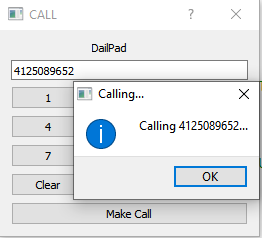


Figure 11: Number panel

If the length of the number is 10 then the above message box will get displayed.

1. **Simple calculator**

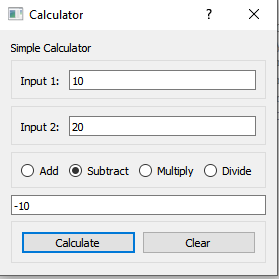


Figure 12: Simple Calculator

This is the simple calculator application where user can do add, subtract, multiply, division operations and get the respective result after clicking on calculate button. Clear button will clear input1, input2 and result field.

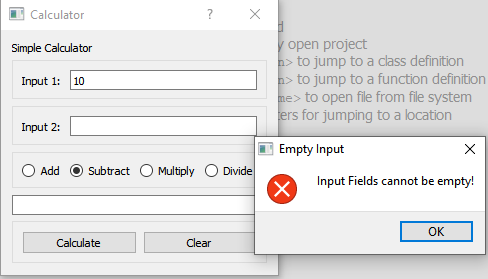


Figure 13: Simple Calculator

If one of the input field is empty then then user will get above message box

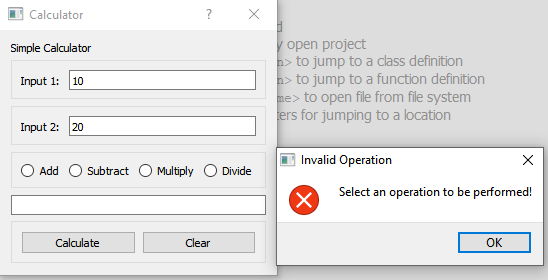


Figure 14: Simple Calculator

If user forgot to select the operation then then user will get above message box

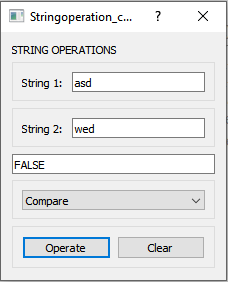


Figure 15: String Operations

This is the simple calculator application where we used combo box instead of radio button for selecting operations (add, subtract, multiply and division)

1. **String operation**

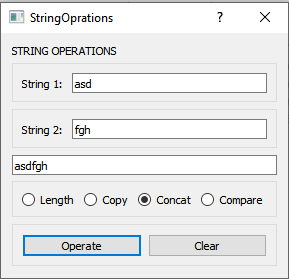


Figure 16: String Operations

This is the string operation application where user can get the length of the string, can get copy of the string, can get concatenated result of the string, can compare the string.

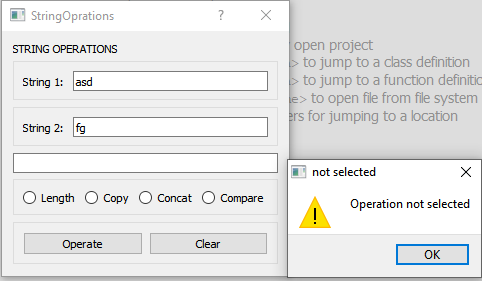


Figure 17: String Operations

If user forgot to select the operation then he will get above message box

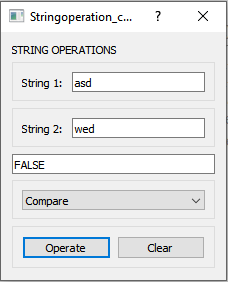


Figure 18: String Operations

This is the string operation application where user can get the length of the string, can get copy of the string, can get concatenated result of the string, can compare the string. We used combo box instead of radio buttons in this application.

1. **Temperature and Volume**

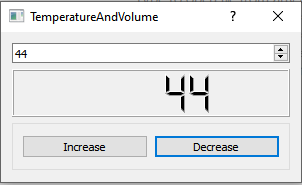


Figure 19: Temperature and Volume

This is the application which shows temperature in LCD display. User can increase or decrease the temperature using combo box or buttons.

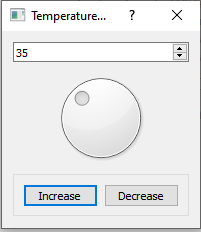


Figure 20: Temperature and Volume

This is the application where we used QDial to increase or decrease the temperature. Respective value will get update into the combo box.

## **Week 2**

### C++ Assignment Problem

Write a program that creates a list of bunny objects.

* Each bunny object must have

gender: Male, Female (random at creation 50%-50%)

color: white, brown, black, spotted

age : 0-10 (years old)

Name : randomly chosen at creation from a list of bunny names.

radioactive\_mutant\_vampire, bunny: true/false (decided at time of bunny creation 2% chance of true)

* At program initialization 5 bunnies must be created and given random colors.
* Each turn afterwards the bunnies age 1 year.
* So long as there is at least one male age 2 or older, for each female bunny in the list age 2 or older;
* a new bunny is created each turn. (i.e. if there was 1 adult male and 3 adult female bunnies, three new bunnies would be born each turn)
* New bunnies born should be the same color as their mother.
* If a bunny becomes older than 10 years old, it dies.
* If a radioactive mutant vampire bunny is born then each turn it will change exactly one non radioactive bunny into a radioactive vampire bunny.
* (if there are two radioactive mutant vampire bunnies two bunnies will be changed each turn and so on...)
* Radioactive vampire bunnies are excluded from regular breeding and do not count as adult bunnies.
* Radioactive vampire bunnies do not die until they reach age 50.
* The program should print a list of all the bunnies in the colony each turn along w/ all the bunnies details, sorted by age.
* The program should also output each turns events such as

"Bunny Thumper was born!

Bunny Fufu was born!

Radioactive Mutant Vampire Bunny Darth Maul was born!

Bunny Julius Caesar died!

* The program should write all screen output to a file.
* When all the bunnies have died the program terminates.
* If the bunny population exceeds 1000 a food shortage must occur killing exactly half of the bunnies (randomly chosen)

? Modify the program to run in real time, with each turn lasting 2 seconds, and a one second pause between each announcement.

?? Allow the user to hit the 'k' key to initiate a mass rabit cull! which causes half of all the rabits to be killed (randomly chosen).

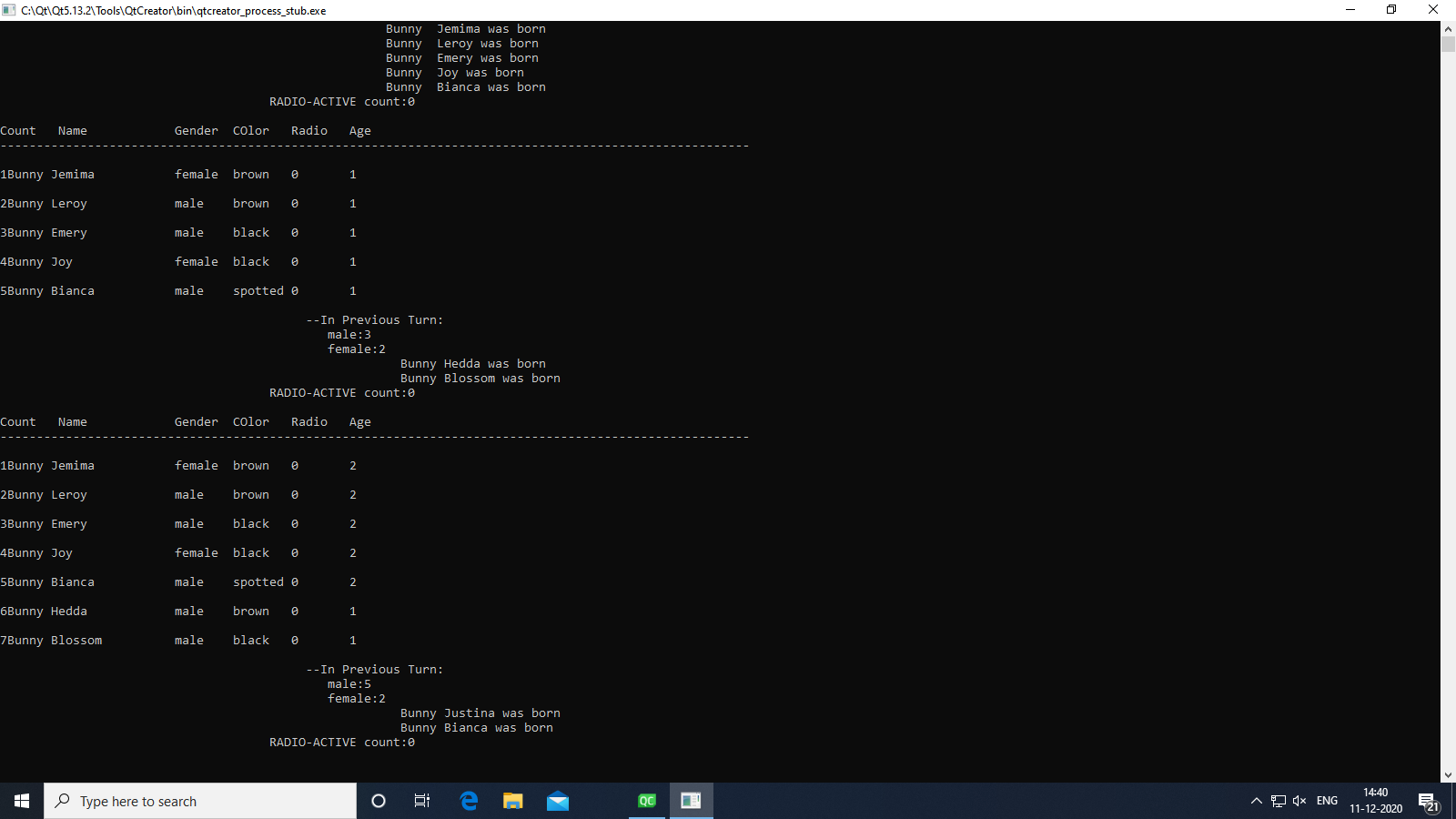


Figure 21: Assignment Problem

### Waveform drawing using QT

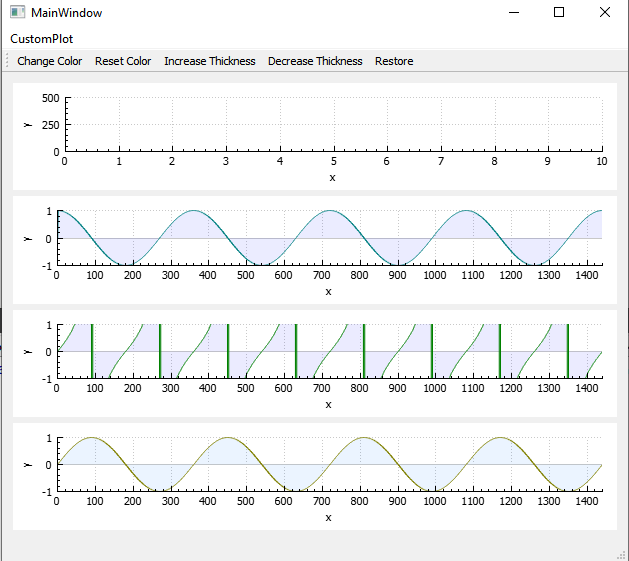


Figure 22: Waveform drawing using QT

This application has implemented in QT which contains waveforms. The method which is used for waveform plotting is qcustom plot. This application contains various functionality for waveform like change color of the waveform, reset color of the waveform, Increase or decrease thickness of the waveform, Restore the changes. It also contains zoom in and zoom out functionality.

## 

## **Week 3**

### **QML**

QML (Qt Modeling Language) is a [user interface mark-up language](https://en.wikipedia.org/wiki/User_interface_markup_language). It is a [declarative](https://en.wikipedia.org/wiki/Declarative_programming) language (similar to [CSS](https://en.wikipedia.org/wiki/CSS) and [JSON](https://en.wikipedia.org/wiki/JSON)) for designing user interface–centric applications. Inline [JavaScript](https://en.wikipedia.org/wiki/JavaScript) code handles imperative aspects. It is associated with [Qt Quick](https://en.wikipedia.org/wiki/Qt_Quick), the UI creation kit originally developed by [Nokia](https://en.wikipedia.org/wiki/Nokia) within the [Qt](https://en.wikipedia.org/wiki/Qt_(framework)) framework. Qt Quick is used for mobile applications where touch input, fluid animations and user experience are crucial. QML is also used with Qt3D to describe a 3D scene and a frame graph rendering methodology. A QML document describes a hierarchical object tree. QML modulesshipped with Qt include primitive graphical building blocks (e.g., Rectangle, Image), modeling components (e.g., FolderListModel, XmlListModel), behavioral components (e.g., TapHandler, DragHandler, State, Transition, Animation), and more complex controls (e.g., Button, Slider, Drawer, Menu). These elements can be combined to build components ranging in complexity from simple buttons and sliders, to complete internet-enabled programs.

**Applications implemented in QML**

* 1. **Snake game**

Figure 23: Snake Game

Snake game is the application in which we have created game application. Above figure shows how to play this game and what are the rules are there to play this game.

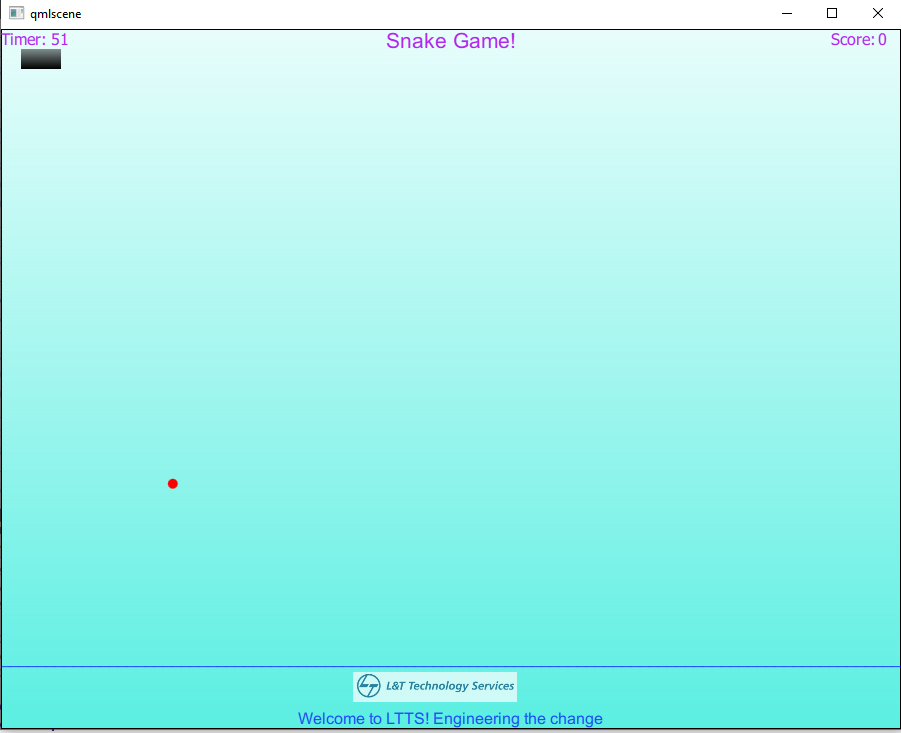


Figure 24: Snake Game

The above figure is the UI of the game where we can see the snake, dot, timer and score on both the end.

* 1. **QML Application**

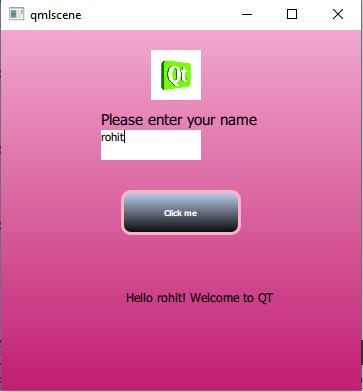


Figure 25: Demo Application

This application taking name as a input and printing message with that input on the screen.

* 1. **Animation**

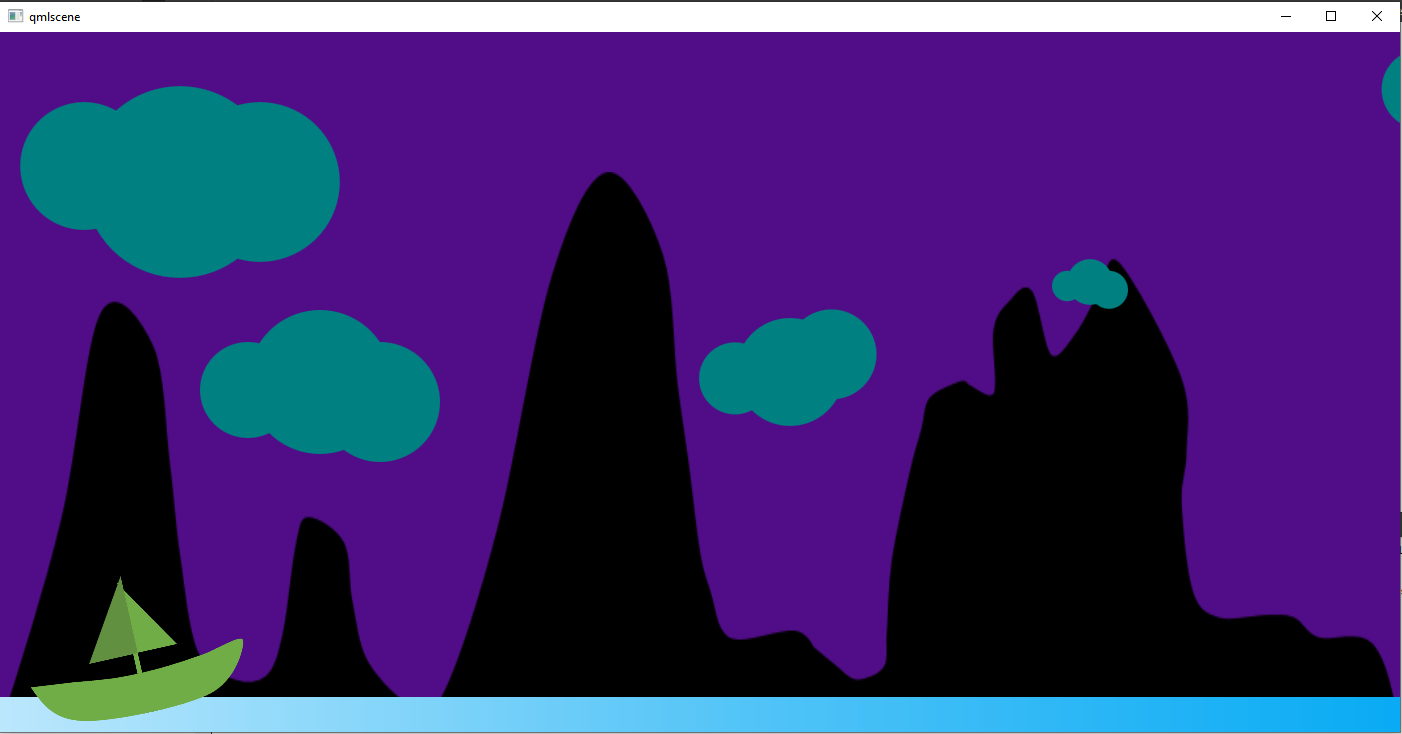


Figure 26: Animation Application

This is the application in which we used different types of animation effects on the objects present in the image.

### QT QML Integration

QML is designed to be easily extensible through C++ code. The classes in the [Qt QML](https://doc.qt.io/qt-5/qtqml-index.html) module enable QML objects to be loaded and manipulated from C++, and the nature of QML engine's integration with Qt's [meta object system](https://doc.qt.io/qt-5/metaobjects.html) enables C++ functionality to be invoked directly from QML. This allows the development of hybrid applications which are implemented with a mixture of QML, JavaScript and C++ code.

Integrating QML and C++ provides a variety of opportunities, including the ability to:

* Separate the user interface code from the application logic code, by implementing the former with QML and JavaScript within [QML documents](https://doc.qt.io/qt-5/qtqml-documents-topic.html), and the latter with C++
* Use and invoke some C++ functionality from QML (for example, to invoke your application logic, use a data model implemented in C++, or call some functions in a third-party C++ library)
* Access functionality in the [Qt QML](https://doc.qt.io/qt-5/qtqml-index.html) or [Qt Quick](https://doc.qt.io/qt-5/qtquick-index.html) C++ API (for example, to dynamically generate images using [QQuickImageProvider](https://doc.qt.io/qt-5/qquickimageprovider.html))
* Implement your own [QML object types](https://doc.qt.io/qt-5/qtqml-typesystem-objecttypes.html) from C++ — whether for use within your own specific application, or for distribution to others

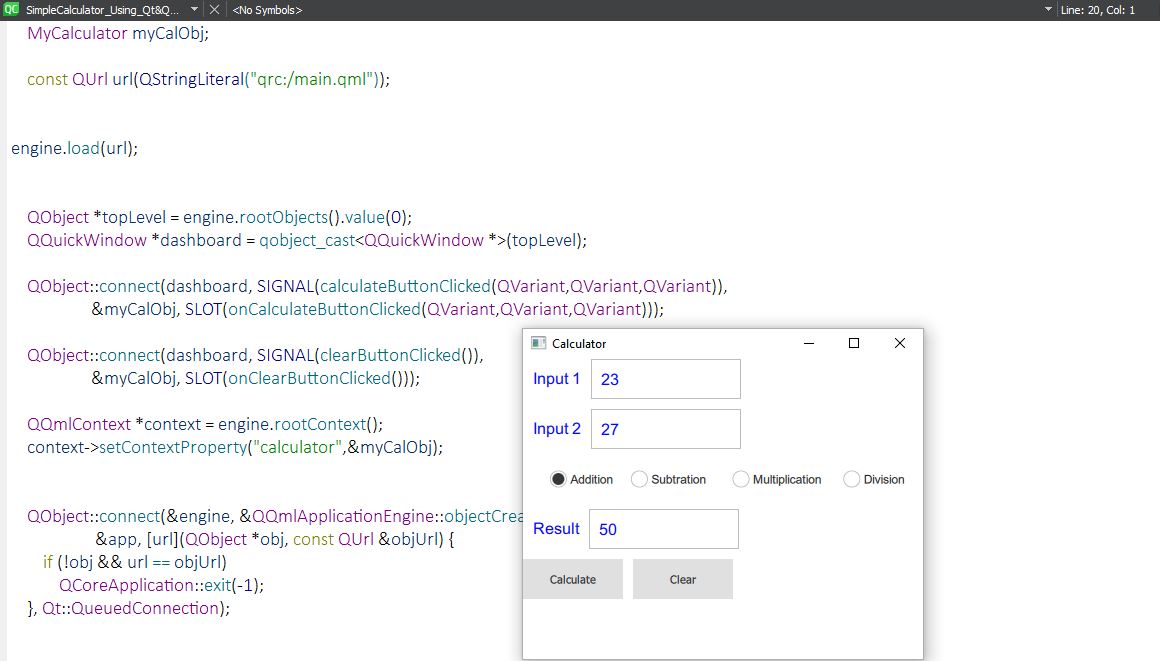


Figure 27: QT-QML Integration

This is the application contain integration of both QT and QML. You can see here myCalObj is a QT object and dashboard is QML object. Signal is there on QML side and its respective slot is there on QT side.

### Waveform drawing using QML

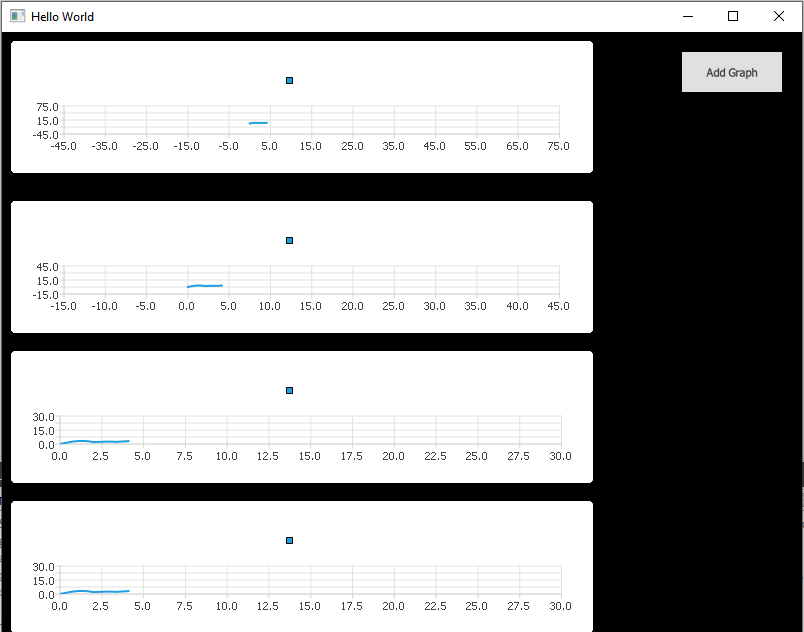


Figure 28: Waveform drawing using QML

This application has implemented in QML which contains waveforms. The method which is used for waveform plotting is qtcharts. User can add graphs after clicking on Add Graph button. It also contains zoom in and zoom out functionality.

## **Week 4**

### Shadow Project Implementation

**Window:**

* The Window object creates a new top-level window for a Qt Quick scene. It automatically sets up the window for use with QtQuick 2.x graphical types.
* To use this type, you will need to import the module with the following line

import QtQuick.Window 2.2

**MouseArea:**

* A [MouseArea](file:///C:\Users\99002660\AppData\Local\Microsoft\Windows\INetCache\Content.Outlook\G5W6GZ6A\qml-qtquick-mousearea.html) is an invisible item that is typically used in conjunction with a visible item to provide mouse handling for that item. By effectively acting as a proxy, the logic for mouse handling can be contained within a [MouseArea](file:///C:\Users\99002660\AppData\Local\Microsoft\Windows\INetCache\Content.Outlook\G5W6GZ6A\qml-qtquick-mousearea.html) item.
* The [enabled](file:///C:\Users\99002660\AppData\Local\Microsoft\Windows\INetCache\Content.Outlook\G5W6GZ6A\qml-qtquick-mousearea.html#enabled-prop) property is used to enable and disable mouse handling for the proxied item. When disabled, the mouse area becomes transparent to mouse events.
* [MouseArea](file:///C:\Users\99002660\AppData\Local\Microsoft\Windows\INetCache\Content.Outlook\G5W6GZ6A\qml-qtquick-mousearea.html) is an invisible Item, but it has a visible property. When set to false, the mouse area becomes transparent to mouse events.

**Grid Layout:**

* Grid is a type that positions its child items in grid formation.
* A Grid creates a grid of cells that is large enough to hold all its child items, and places these items in the cells from left to right and top to bottom. Each item is positioned at the top-left corner of its cell with position (0, 0).
* A Grid defaults to four columns, and creates as many rows as are necessary to fit all its child items. The number of rows and columns can be constrained by setting the [rows](https://www3.sra.co.jp/qt/relation/doc/qtquick/qml-qtquick-grid.html#rows-prop) and [columns](https://www3.sra.co.jp/qt/relation/doc/qtquick/qml-qtquick-grid.html#columns-prop) properties.

**Button:**

* Button presents a push-button control that can be pushed or clicked by the user. Buttons are normally used to perform an action, or to answer a question. Typical buttons are OK, Apply, Cancel, Close, Yes, No, and Help.
* A button emits the signal [clicked ()](file:///C:\Users\99002660\AppData\Local\Microsoft\Windows\INetCache\Content.Outlook\G5W6GZ6A\qml-qtquick-controls2-abstractbutton.html#clicked-signal) when it is activated by the user. Connect to this signal to perform the button's action. Buttons also provide the signals [canceled()](file:///C:\Users\99002660\AppData\Local\Microsoft\Windows\INetCache\Content.Outlook\G5W6GZ6A\qml-qtquick-controls2-abstractbutton.html#canceled-signal), [doubleClicked()](file:///C:\Users\99002660\AppData\Local\Microsoft\Windows\INetCache\Content.Outlook\G5W6GZ6A\qml-qtquick-controls2-abstractbutton.html#doubleClicked-signal), [pressed()](file:///C:\Users\99002660\AppData\Local\Microsoft\Windows\INetCache\Content.Outlook\G5W6GZ6A\qml-qtquick-controls2-abstractbutton.html#pressed-signal), [released()](file:///C:\Users\99002660\AppData\Local\Microsoft\Windows\INetCache\Content.Outlook\G5W6GZ6A\qml-qtquick-controls2-abstractbutton.html#released-signal) and [pressAndHold()](file:///C:\Users\99002660\AppData\Local\Microsoft\Windows\INetCache\Content.Outlook\G5W6GZ6A\qml-qtquick-controls2-abstractbutton.html#pressAndHold-signal) for long presses.

**Text:**

* Text items can display both plain and rich text. For example, red text with a specific font and size can be defined

**Dial:**

* The Dial is like a traditional dial knob that is found on devices such as stereos or industrial equipment. It allows the user to specify a value within a range.
* The value of the dial is set with the [value](file:///C:\Users\99002660\AppData\Local\Microsoft\Windows\INetCache\Content.Outlook\G5W6GZ6A\qml-qtquick-controls2-dial.html#value-prop) property. The range is set with the [from](file:///C:\Users\99002660\AppData\Local\Microsoft\Windows\INetCache\Content.Outlook\G5W6GZ6A\qml-qtquick-controls2-dial.html#from-prop) and [to](file:///C:\Users\99002660\AppData\Local\Microsoft\Windows\INetCache\Content.Outlook\G5W6GZ6A\qml-qtquick-controls2-dial.html#to-prop) properties. To enable or disable wrapping, use the [wrap](file:///C:\Users\99002660\AppData\Local\Microsoft\Windows\INetCache\Content.Outlook\G5W6GZ6A\qml-qtquick-controls2-dial.html#wrap-prop) property.
* onMoved ():This signal is emitted when the dial has been interactively moved by the user by either touch, mouse, or keys.

**ComboBox:**

* [ComboBox](file:///C:\Users\99002660\AppData\Local\Microsoft\Windows\INetCache\Content.Outlook\G5W6GZ6A\qml-qtquick-controls2-combobox.html) is a combined button and popup list. It provides a means of presenting a list of options to the user in a way that takes up the minimum amount of screen space.
* [ComboBox](file:///C:\Users\99002660\AppData\Local\Microsoft\Windows\INetCache\Content.Outlook\G5W6GZ6A\qml-qtquick-controls2-combobox.html) is populated with a data model. The data model is commonly a JavaScript array, a [ListModel](file:///C:\Users\99002660\AppData\Local\Microsoft\Windows\INetCache\Content.Outlook\qtqml\qml-qtqml-models-listmodel.html) or an integer, but other types of [data models](file:///C:\Users\99002660\AppData\Local\Microsoft\Windows\INetCache\Content.Outlook\qtquick\qtquick-modelviewsdata-modelview.html#qml-data-models) are also supported.

**GradientStop:**

* The position and color properties describe the color used at a given position in a gradient, as represented by a gradient stop.

**ValueAxis:**

* The [ValueAxis](file:///C:\Users\99002660\AppData\Local\Microsoft\Windows\INetCache\Content.Outlook\G5W6GZ6A\qml-qtcharts-valueaxis.html) type can be set up to show an axis line with tick marks, grid lines, and shades. The values on the axis are drawn at the positions of tick marks.
* Max: real

The maximum value on the axis. When setting this property, the minimum value is adjusted if necessary, to ensure that the range remains valid.

|  |
| --- |
| * **min**: [real](file:///C:\Users\99002660\AppData\Local\Microsoft\Windows\INetCache\Content.Outlook\qtqml\qml-real.html) |

The minimum value on the axis. When setting this property, the maximum value is adjusted if necessary, to ensure that the range remains valid.

|  |
| --- |
| * **tickCount**: [int](file:///C:\Users\99002660\AppData\Local\Microsoft\Windows\INetCache\Content.Outlook\qtqml\qml-int.html) |

The number of tick marks on the axis. This indicates how many grid lines are drawn on the chart. The default value is 5, and the number cannot be less than 2.

**FileDialog:**

* The [FileDialog](file:///C:\Users\99002660\AppData\Local\Microsoft\Windows\INetCache\Content.Outlook\G5W6GZ6A\qml-qt-labs-platform-filedialog.html) type provides a QML API for native platform file dialogs.

**Animation:**

* **Sequential Animation:** The Sequential Animation allow multiple animations to be run together. Animations defined in a Sequential Animation are run one after the other
* **Parallel Animation:** Animations defined in a [Parallel Animation](https://doc.qt.io/qt-5/qml-qtquick-parallelanimation.html) are run at the same time.
* **Pause Animation:** When used in a sequential animation, pause animation is a step when nothing happens, for a specified duration. This property holds the duration of the pause in milliseconds
* **Color Animation :** [ColorAnimation](file:///C:\Users\99002660\AppData\Local\Microsoft\Windows\INetCache\Content.Outlook\G5W6GZ6A\qtquick-animation-example.html#coloranimation) is a specialized [PropertyAnimation](file:///C:\Users\99002660\AppData\Local\Microsoft\Windows\INetCache\Content.Outlook\G5W6GZ6A\qtquick-animation-example.html#propertyanimation) that defines an animation to be applied when a color value changes.
* **Number Animation:** [NumberAnimation](file:///C:\Users\99002660\AppData\Local\Microsoft\Windows\INetCache\Content.Outlook\G5W6GZ6A\qml-qtquick-numberanimation.html) is a specialized [PropertyAnimation](file:///C:\Users\99002660\AppData\Local\Microsoft\Windows\INetCache\Content.Outlook\G5W6GZ6A\qtquick-animation-example.html#propertyanimation) that defines an animation to be applied when a numerical value changes.



Figure 29: Demo UI

#### Browse File

Clicking on the browse button user shall be able to upload dataset to plot waveform

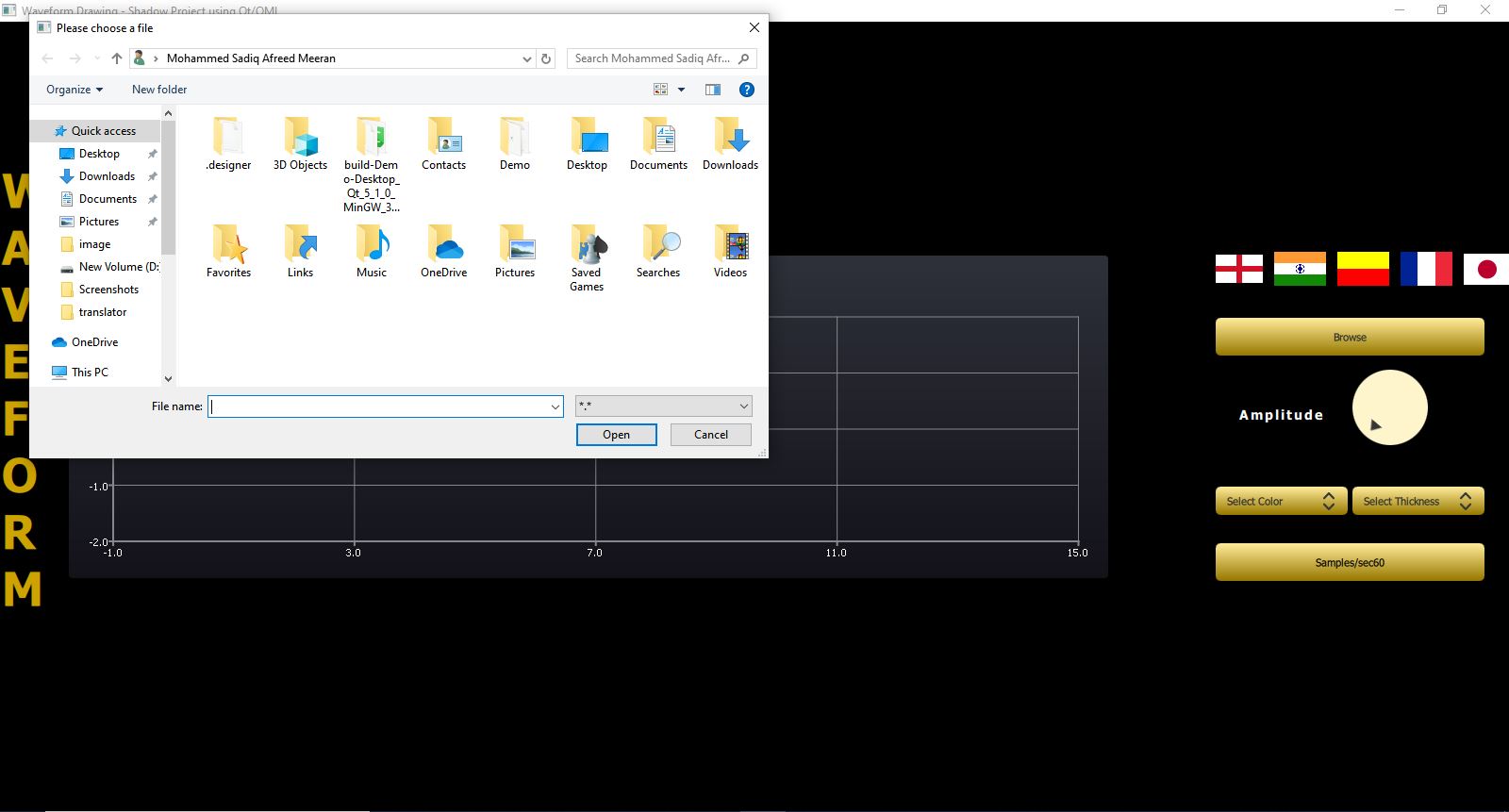


Figure 30: Browse File

#### Plot Waveform

User can plot waveform by uploading dataset into the program. Single or multiple waves can be drawn at the same time. Different type of dataset can be chosen

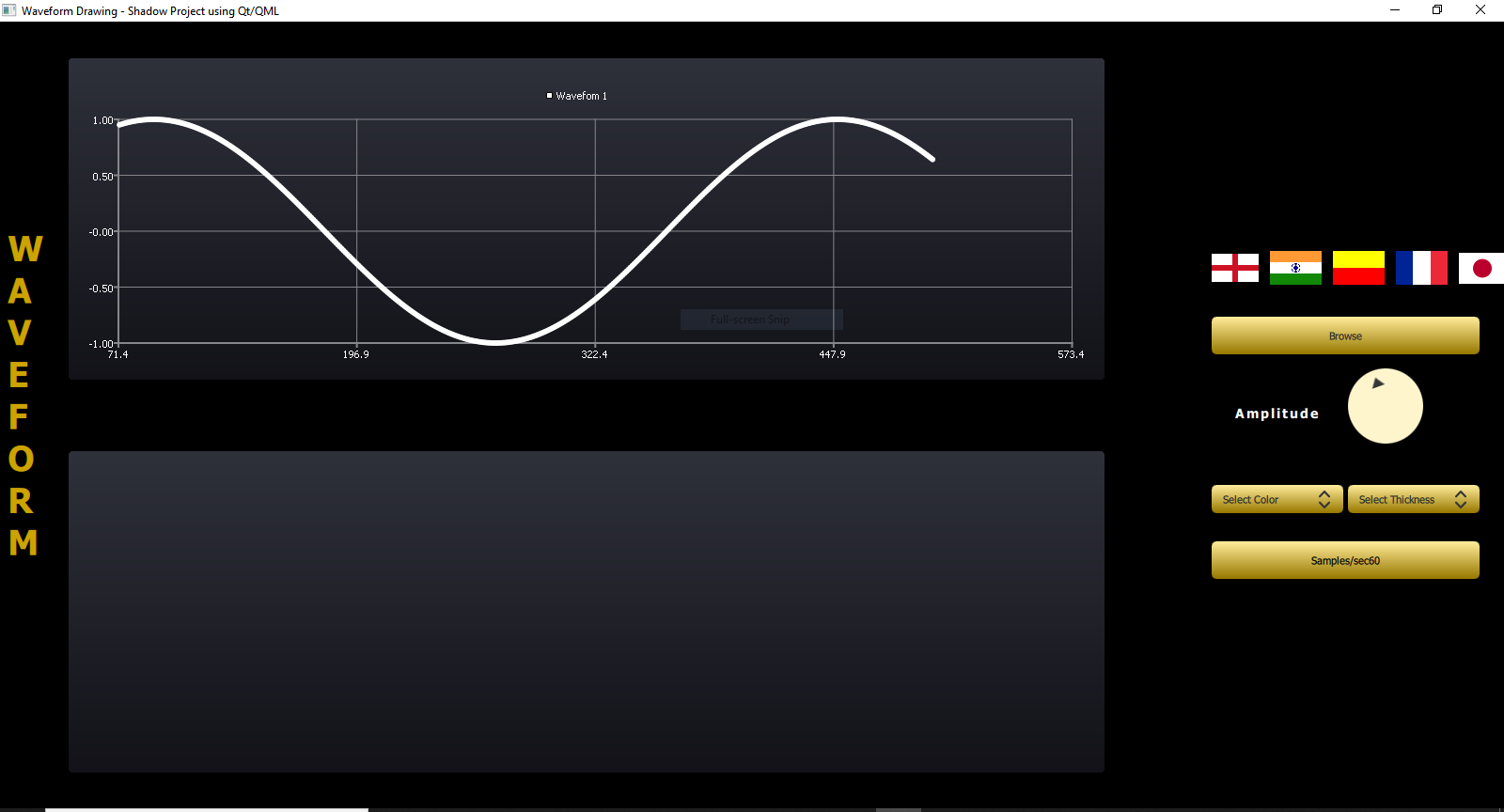


Figure 31: Plotting Waveform

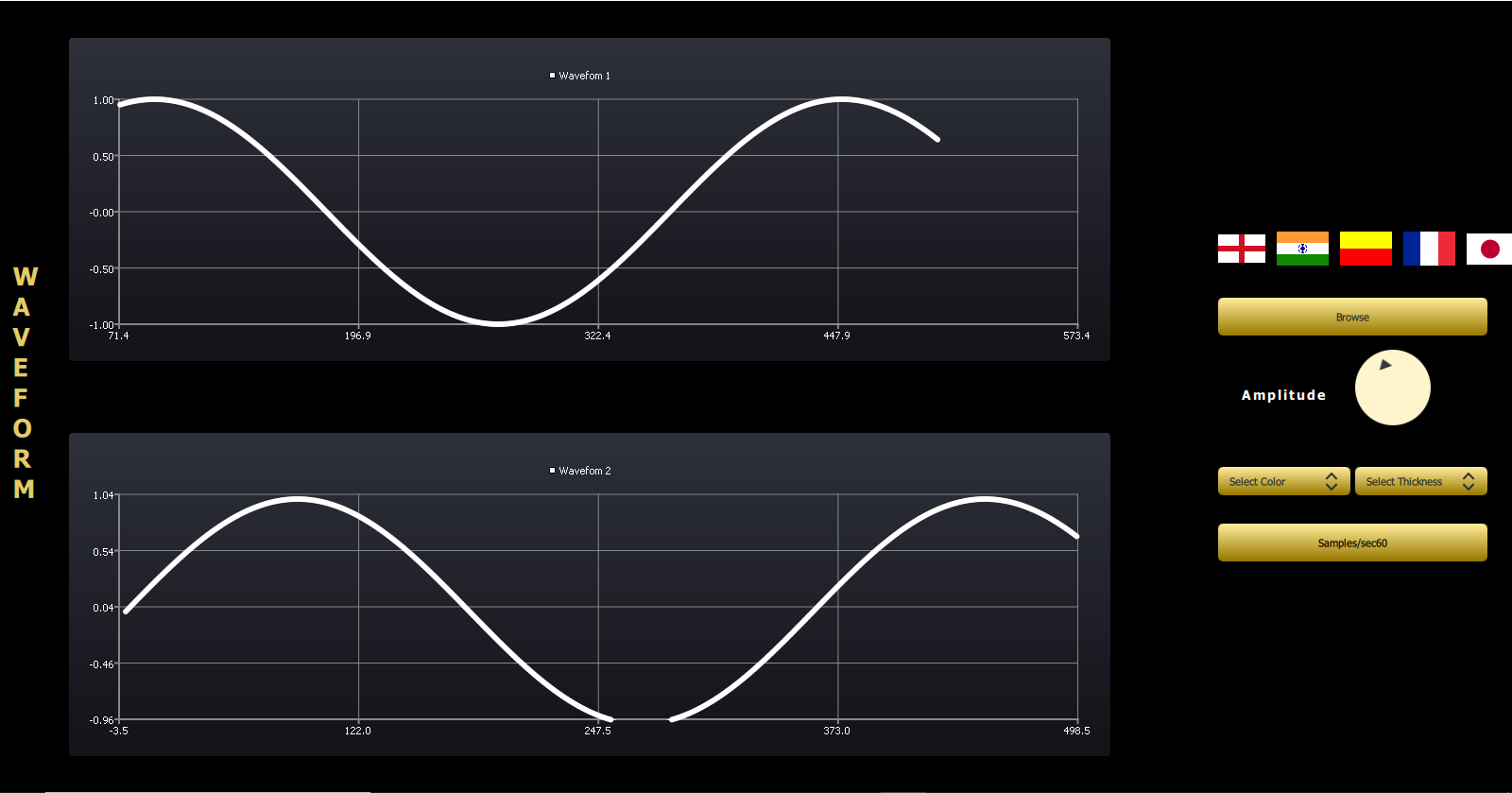


Figure 32: Plotting Waveform

#### Change Color

Clicking on the change color ComboBox user shall be able to change color of one graph or multiple graph at the same time.

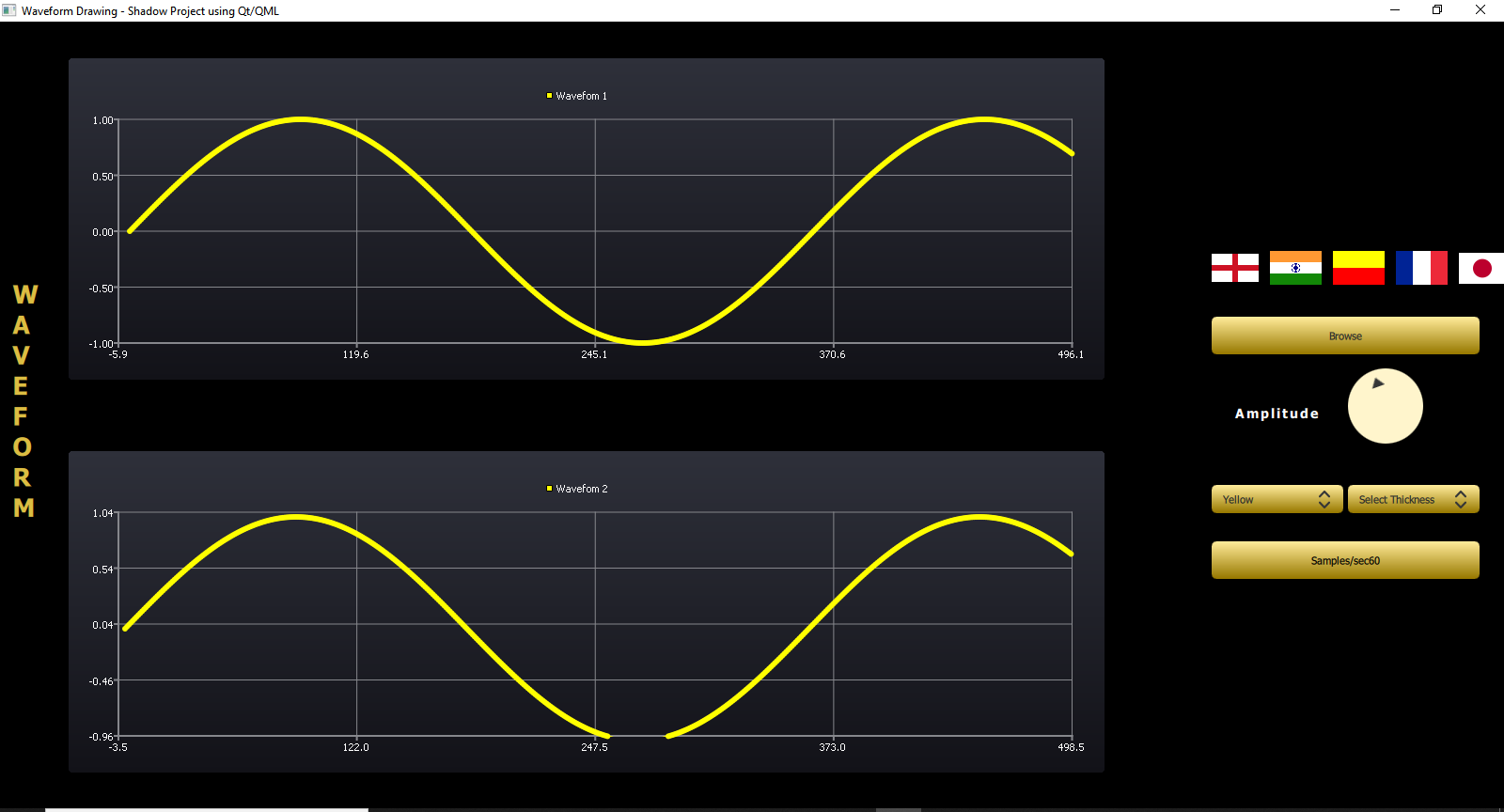


Figure 33: Change Color

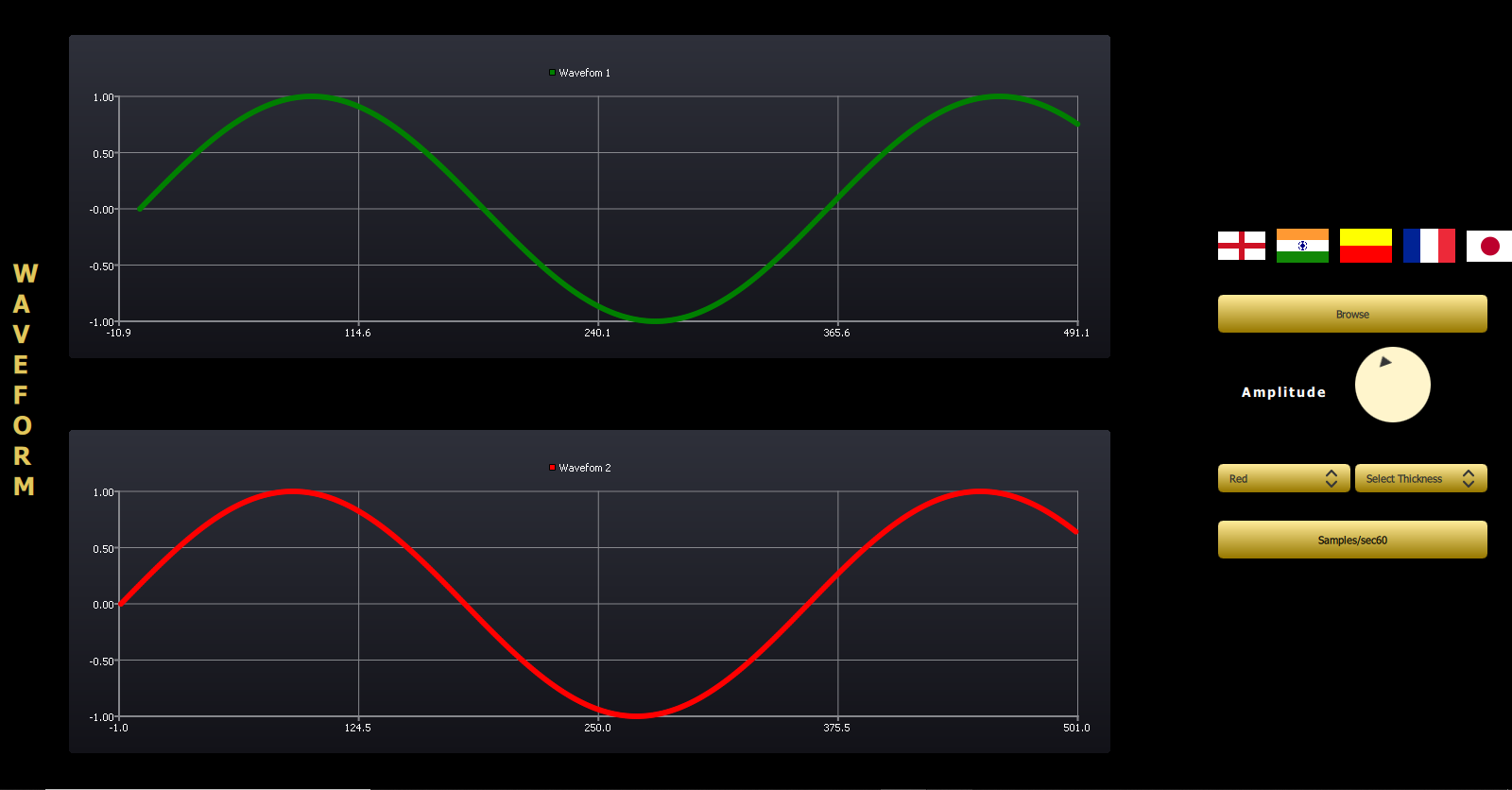


Figure 34: Change Color

#### Change Thickness

Clicking on the change thickness ComboBox user shall be able to change thickness of one graph or multiple graph at the same time.

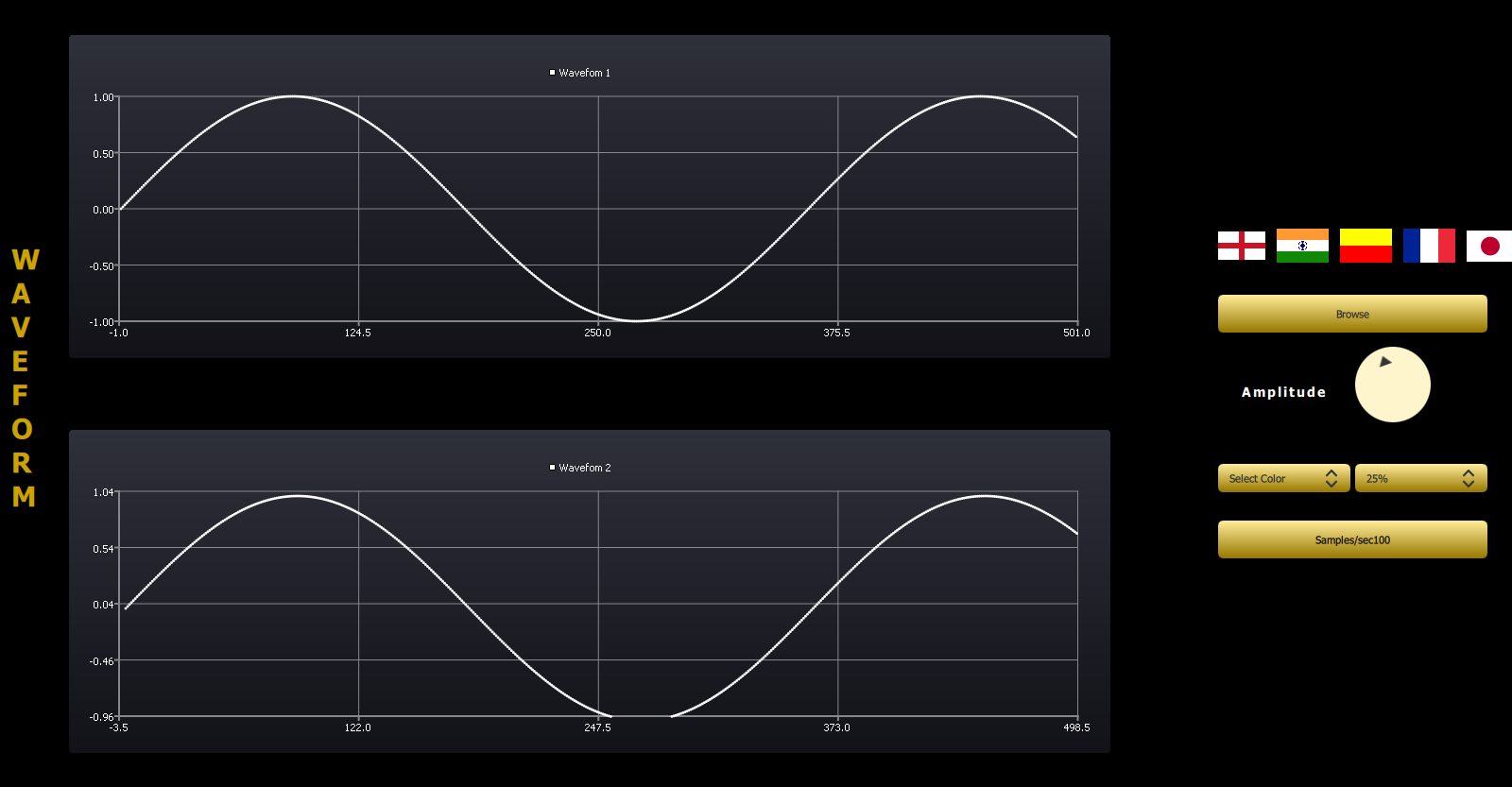


Figure 35: Change Thickness

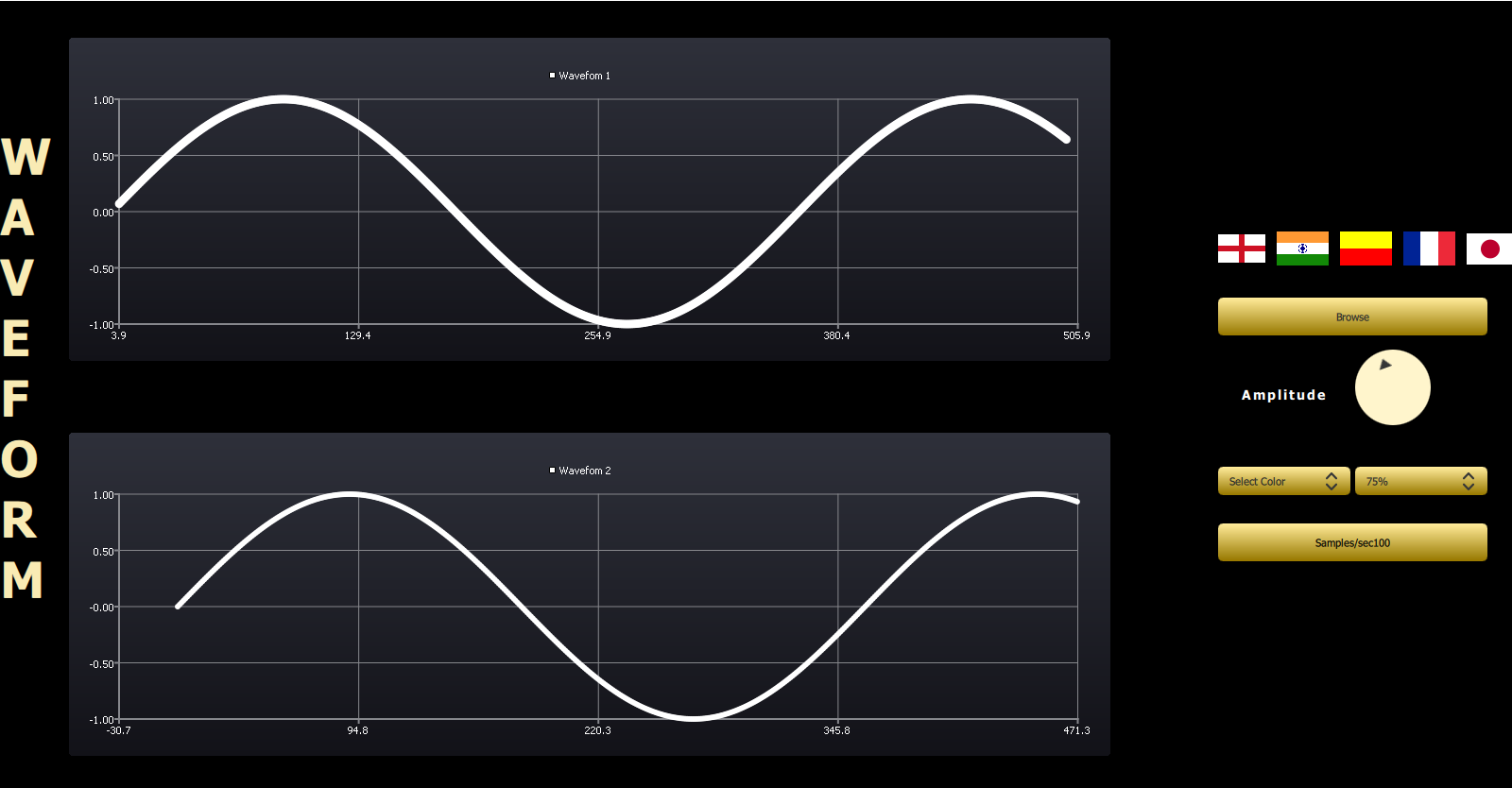


Figure 36: Change Thickness

#### Amplitude

User shall be able to change amplitude of the waveform by moving the dial clockwise or anticlockwise

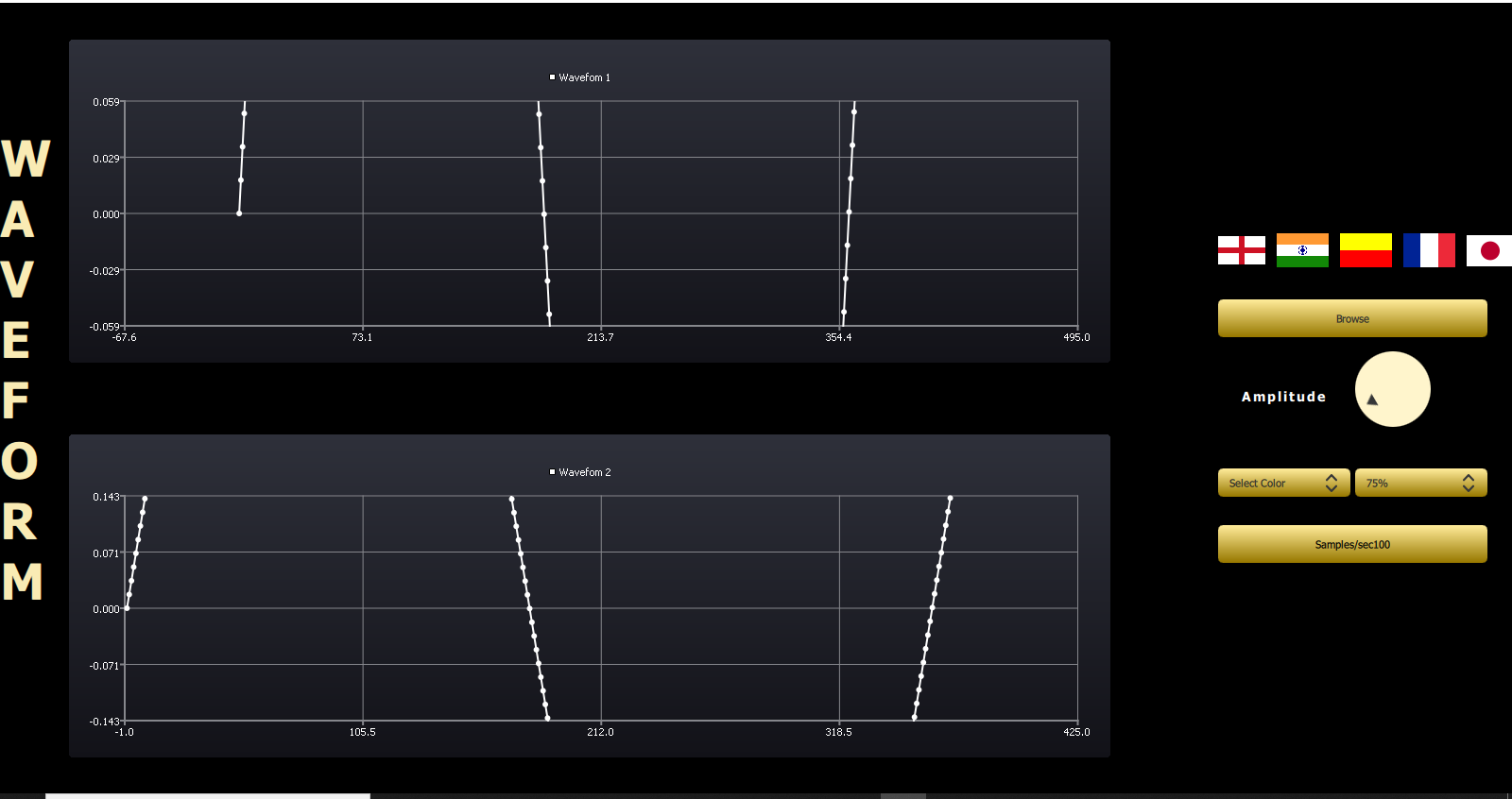


Figure 37: Amplitude

#### Zoom-In, Zoom-out

Scrolling mouse up and down user shall be able to zoom-in and zoom-out of the waveform

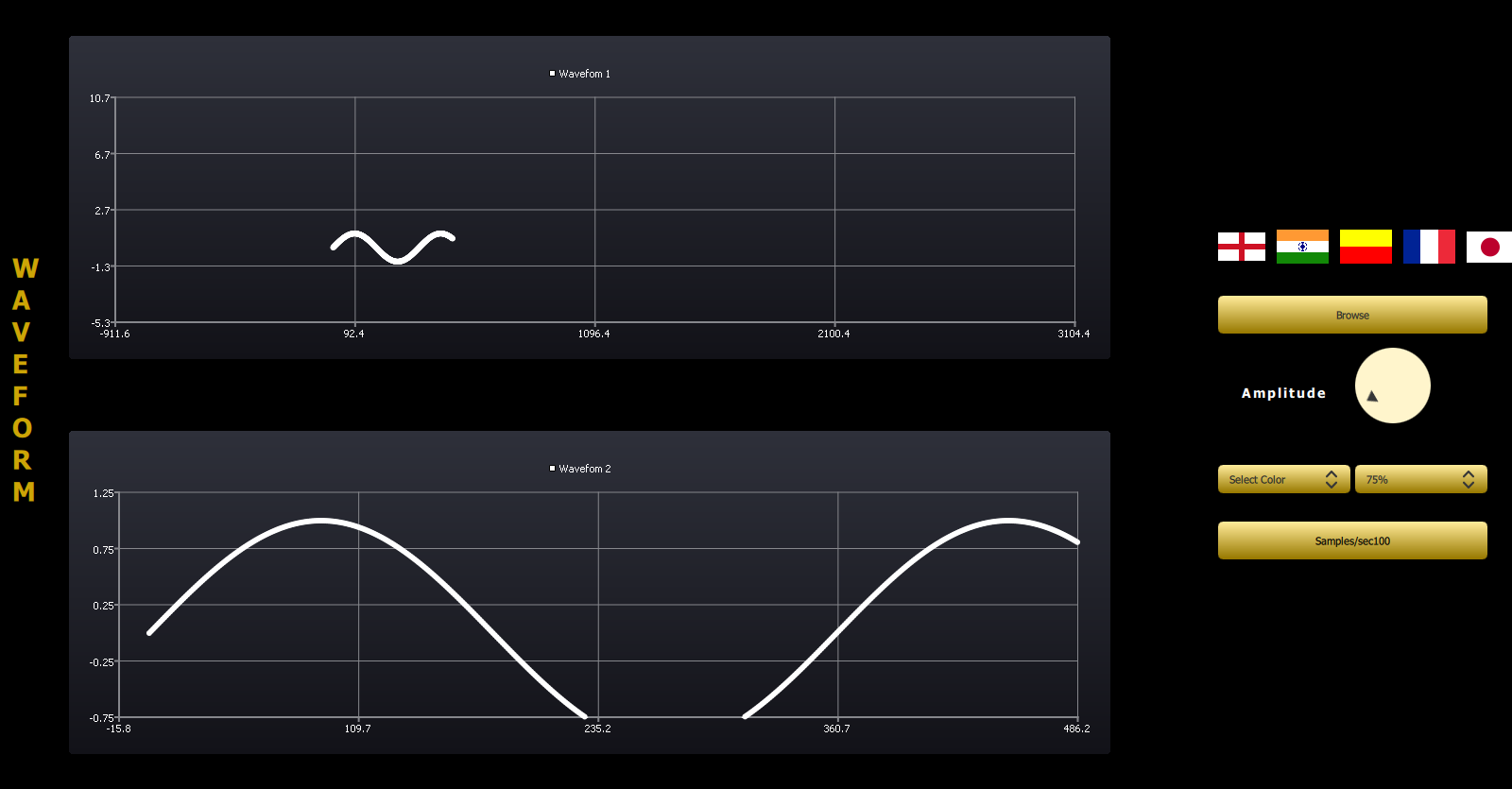


Figure 38: Zoom-In, Zoom-Out

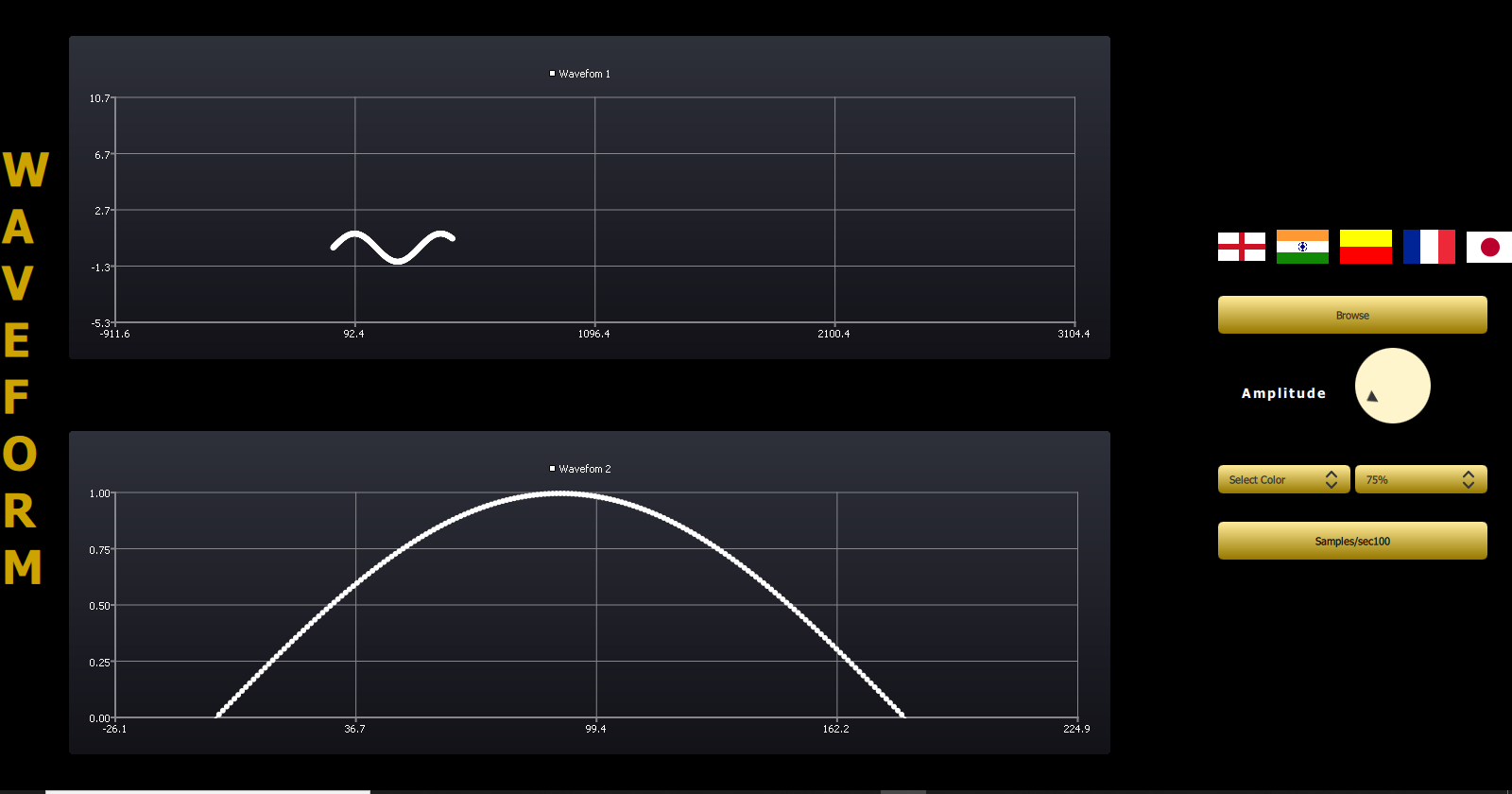
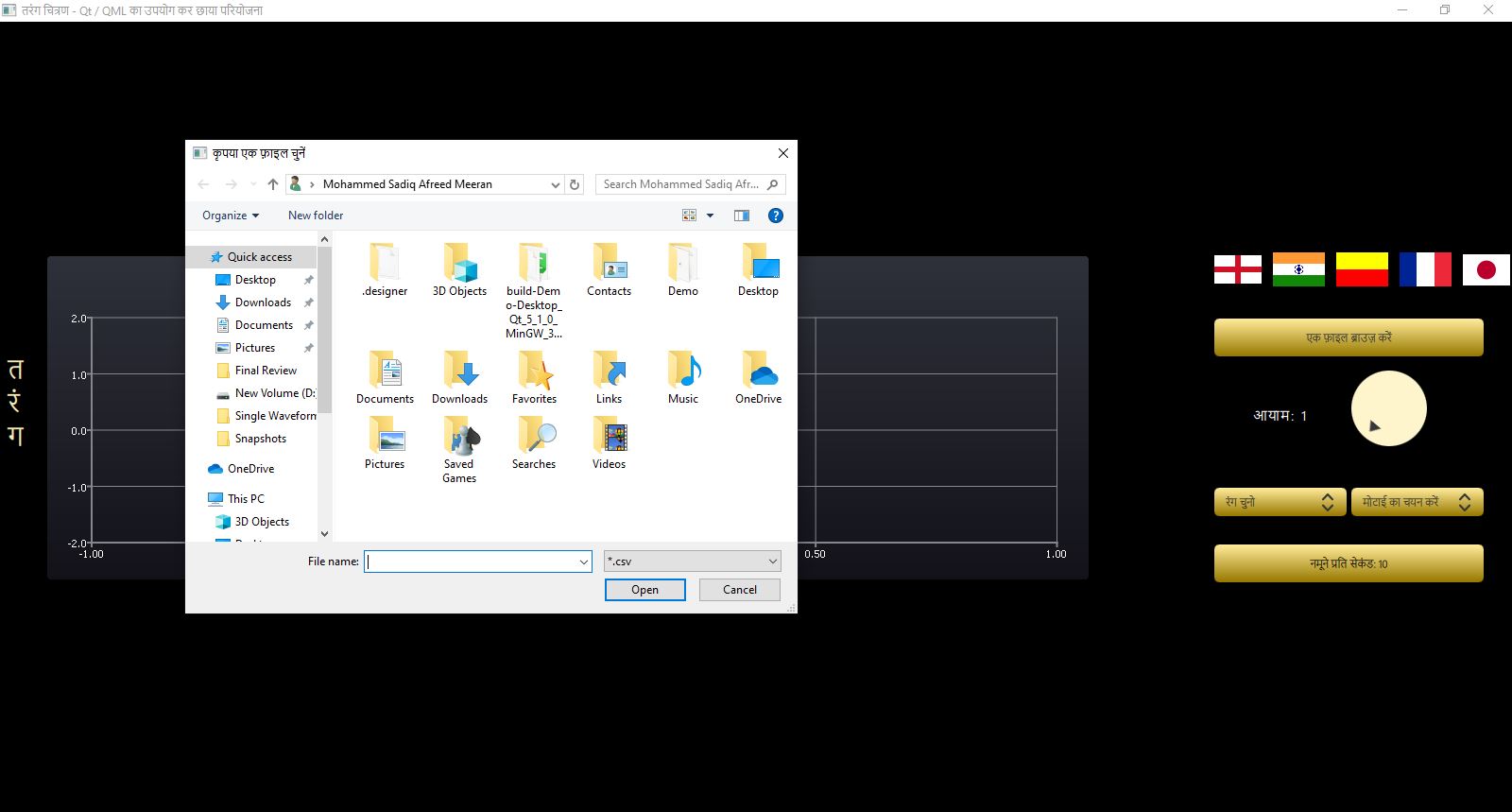
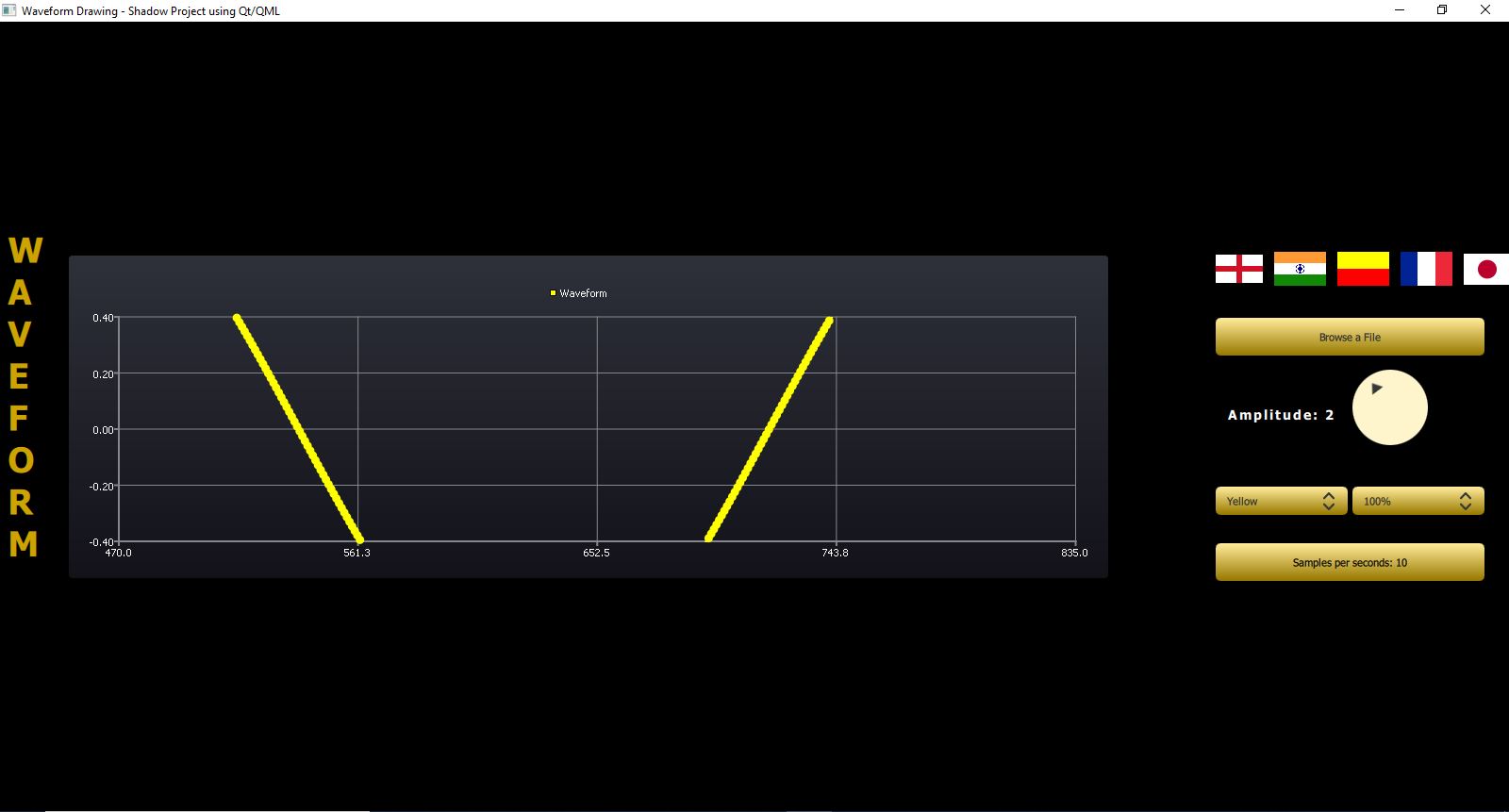


Figure 39: Zoom-In, Zoom-Out

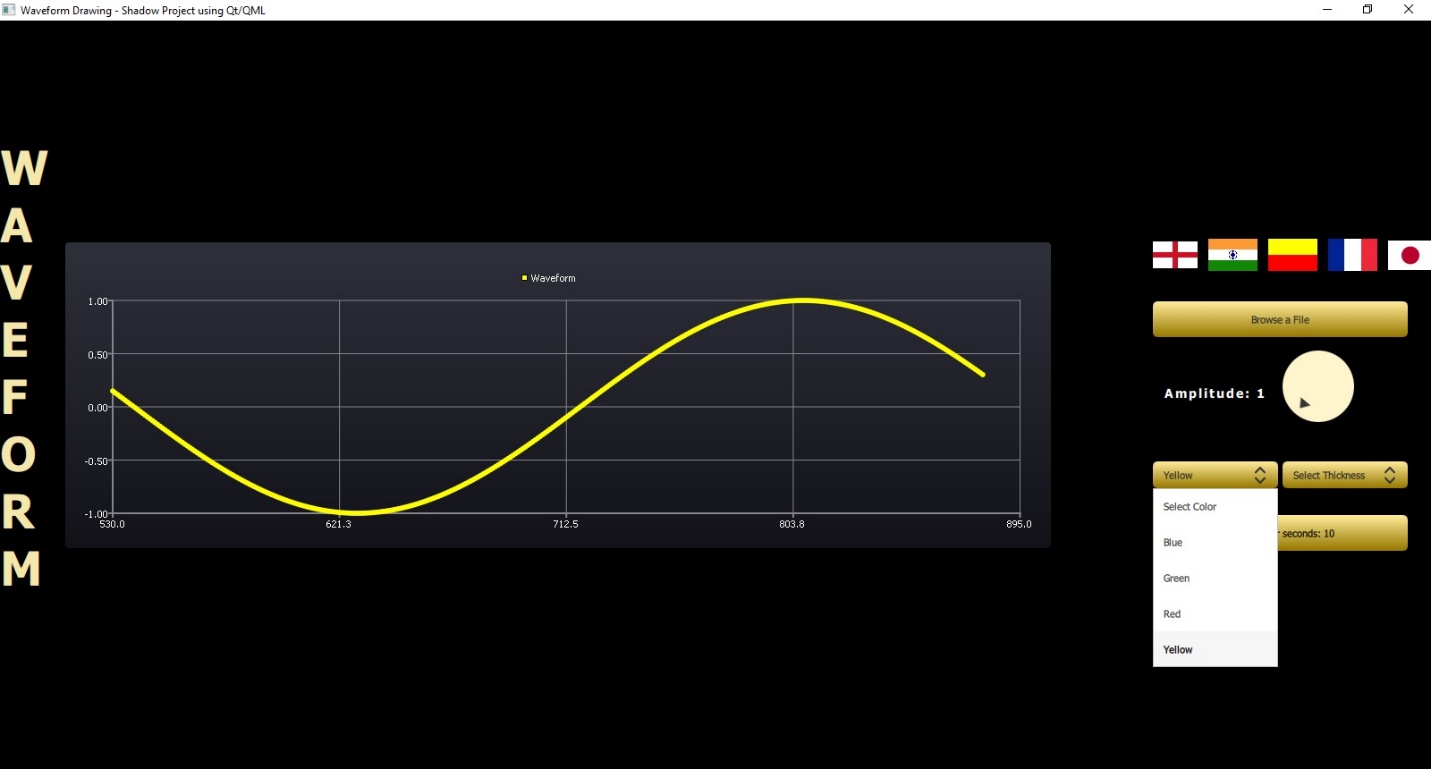
#### Browse File



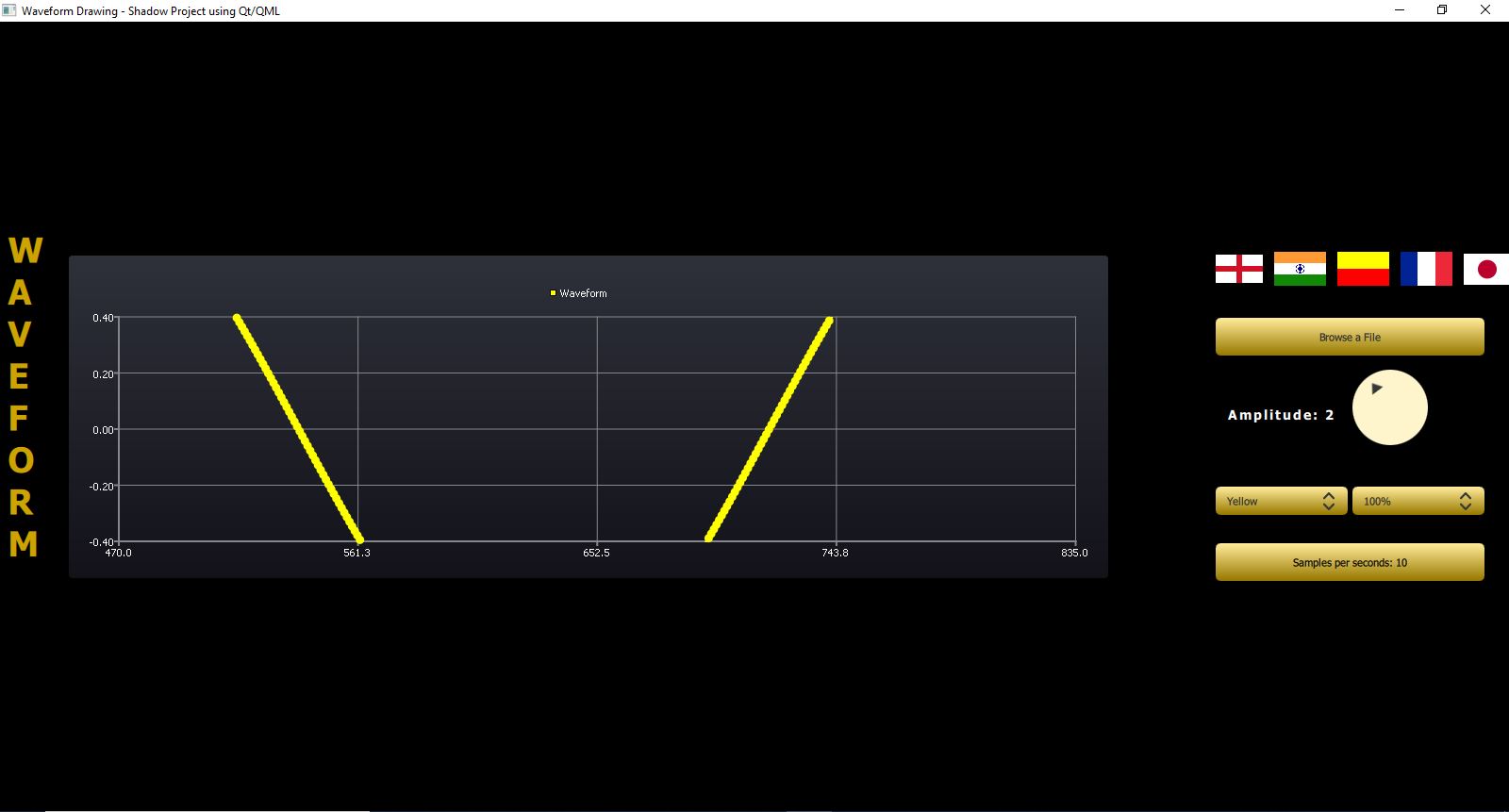
#### Amplitude



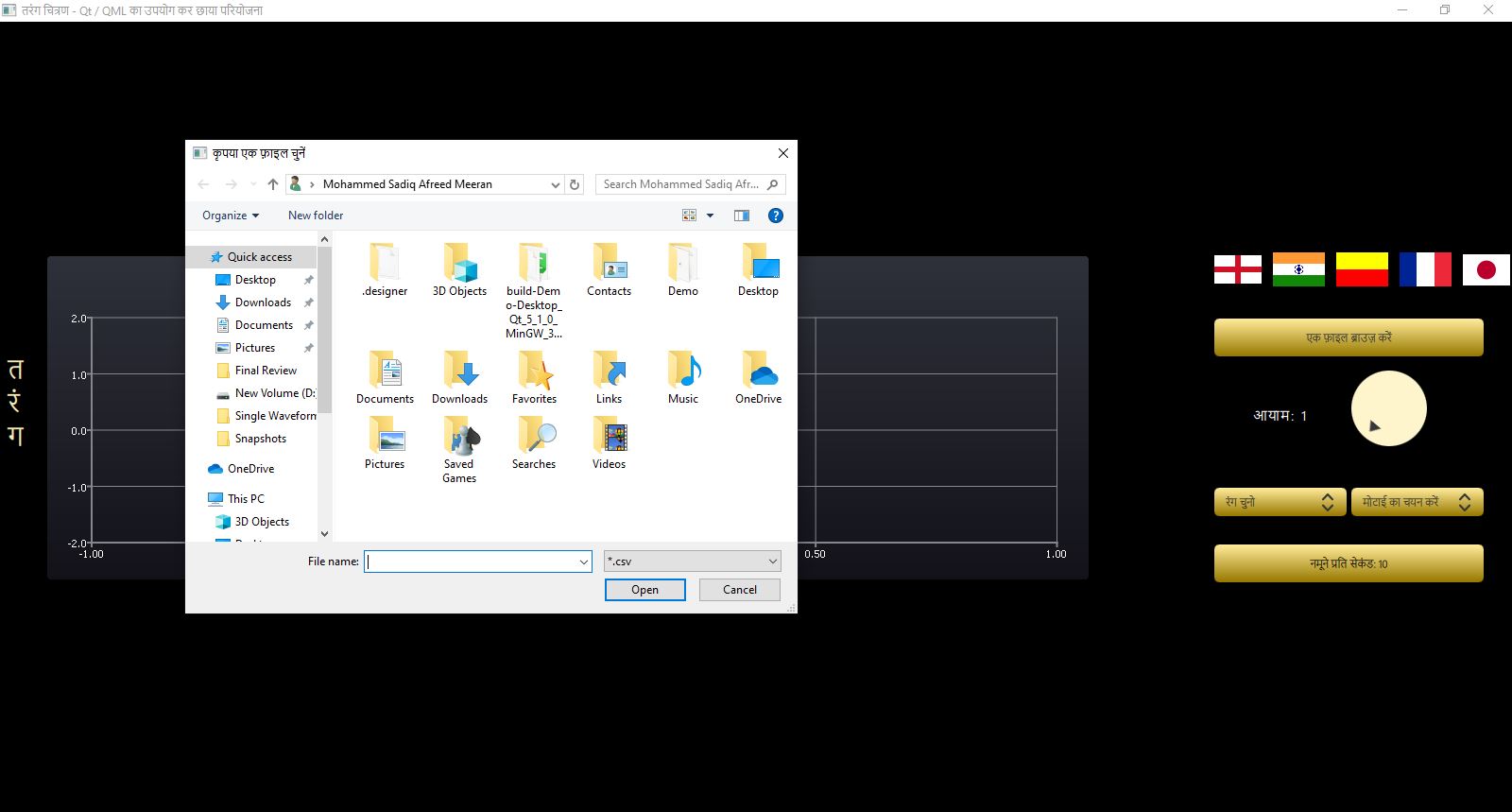
Color Change



#### Thickness



Localization



#### Localization

Localization is the process of adapting software for a specific region or language by adding locale-specific components and translating text.

Clicking on the flag images user can translate text respective languages like Kannada, Hindi, English, French and Japanese

**Qt Linguist:**

*Qt Linguist* is a tool for adding translations to Qt applications.

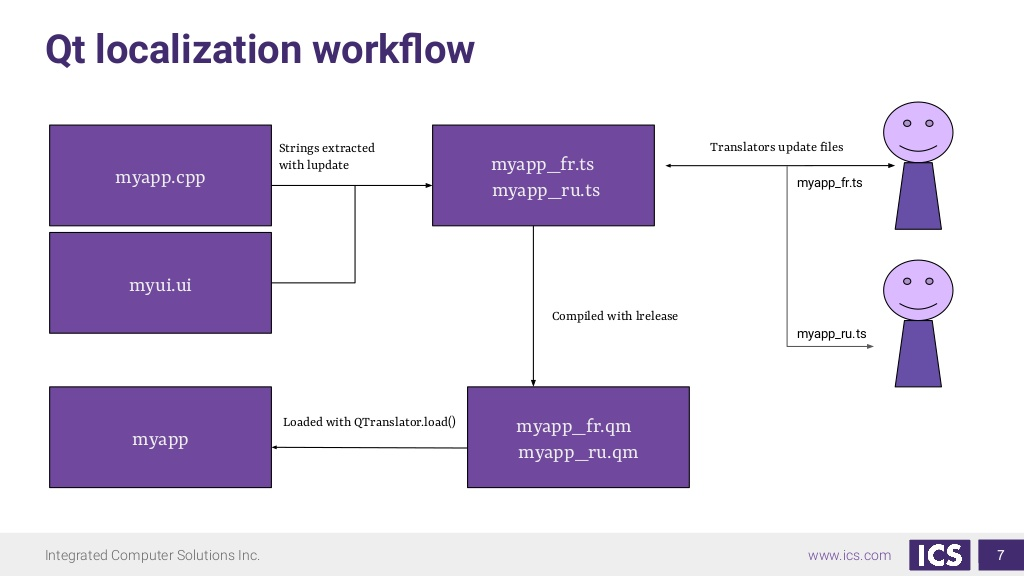


Figure 40: Localization Workflow

**Steps :**

* List the .ts files in your Qt project file.pro using

TRANSLATIONS += myapp\_Kan.ts\

Myapp\_fre.ts

* Run lupdate to extract/merge strings when UI strings change.
* Lrelease is used to compile .ts files to. qm files

# Future Scope

* Add/Remove waveforms dynamically
* Synchronization between different waveforms