**RECTANGLE TOOL**

**About the tool:**

This tool creates a rectangle feature with length on the horizontal axis and width on the vertical axis, with a click and drag. The rectangle is also resizable by dragging the extent of the second point.

**How to use:**

To create rectangle feature with one click:

* Load Quick Digitize plugin.
* Select Polygon layer to which feature is to be added.
* Enable Toggle Edit.
* Click on the Rectangle Tool button.
* Left click on the desired point using the custom cursor, drag till rectangle of required size is formed and release to form the feature.
* To rotate feature, use the Rotate Feature button or the in-built Rotate feature option in QGIS.
* To add attributes to the feature, use the Add Attributes button.

**How it works:**

The entire working of the tool is contained in two Python files:

**rect.py:** Contains the logic of making the rectangle feature.

**recttool.py:** `Links the QGIS canvas to the tool’s working logic.

Finally they are imported to the main program **quick\_digitize.py**.

**rect.py**

**Class:**

**RectByExtentTool(QgsMapTool):**

**Methods:**

**\_\_init\_\_(self, canvas):**Initiates the variables and creates the cursor.

**canvasPressEvent(self,event):** Records the coordinates of the clicked point on the screen and converts it into map coordinates.

**canvasMoveEvent(self,event):** Tracks the movement of the cursor on the screen and sets that to be the second point. Accordingly, it calculates the two other points that form the rectangle and displays the rectangle area in-real-time.

**canvasReleaseEvent(self,event):** Checks if all rectangle has been formed and when the cursor is released, sets the polygon as feature in the layer and refreshes all the parameters for the next iteration.

**recttool.py**

**Class:**

**RectTool**

**Methods:**

**\_\_init\_\_(self, iface, toolBar):** Initializes variables and connects the signals of the dialog box to the mentioned slots.

**rectbyextenttool(self):** Connects to the tool when the button is triggered.

**deactivate(self):** Disconnects from the tool.

**createFeature(self, geom):** Turns on, On the fly transformation of the CRS, creates and adds the feature to the layer.

**ADD FIELDS TOOL**

**About the tool:**

This tool adds field names to the attribute table of the layer from a .csv file selected by the user.

**How to use:**

* Load Quick Digitize plugin.
* Select layer to whose attribute table the fields are to be added.
* Enable Toggle Edit
* Click on the Add Fields button.
* A dialog box will pop up which asks for the user to enter the input file.
* Select the appropriate .csv file and click OK.
* Open the attribute table to view the added fields.

**How it works:**

The entire working of the tool is contained in three Python files:

**ui\_addfield.py:** Contains the code for the UI of the dialog box

**addfieldsgui.py:** Links the UI components to the working logic of the dialog box

**addfieldstool.py:** Reads the field names from the user input file and adds to attribute table

Finally they are imported to the main program **quick\_digitize.py**.

**ui\_addfield.py**

**Class name:**

Ui\_AddField(object)

**Methods:**

**setupUi(self, Dialog)**: Adds objects with specified parameters to dialog box

**retranslateUi(self, Dialog)**: Renames the labels

**addfieldsgui.py**

**Class name:**

AddFieldsGui(QDialog, QWidget, Ui\_AddField)

**Methods:**

**\_\_init\_\_(self, parent, flags):** Initialises variables and connects the signals of the dialog box to the mentioned slots.

**initGui(self):** Refreshes the settings

**select\_input\_file(self):** opens the browse window for user to select the file and accepts the file name.

**close\_1(self):** Emits signal to unset tool

**addfieldstool.py**

**Class name:**

AddFieldsTool

**Methods:**

**\_\_init\_\_(self, iface, toolBar):** Initialises variables and connects the signals of the dialog box to the mentioned slots.

**toggle(self):** Enables the plugin when layer is editable.

**showDialog(self**): Calls the methods of the AddFieldsGui class through an object

**add\_field(self):** Reads the .csv file and adds the fields to the attribute table

**close\_func(self):** Calls close\_1(self) from AddFieldsGui class

**Note:**

To change any of the field names, Go to the respective \*\*\*\_labels.csv and change the 2nd column and you control the placement of the labels in the setupUi\_\* method in the Ui\_AddAttribute class and the which label which row of the column is displayed in retranslateUi\_\* method of Ui\_AddAttribute class.

**ADD ATTRIBUTE TOOL**

**About the tool:**

This tool adds attributes of a selected feature to the attribute table. The dialog box allows the user to enter parameters such as the Geo- Spatial Data, Geometry type, Class, Sub-Class etc. from drop down boxes and other specifics from line edits.

**How to use:**

* Load Quick Digitize plugin.
* Select layer to whose attribute table the fields are to be added.
* Enable Toggle Edit
* Click on the Select Feature button.
* Select the feature whose attributes are to be added.
* Click on the Add Attributes button.
* The dialog box that pops up will contain the following fields: Geo- Spatial Data, Geometry type, Class, Sub-Class, Code.
* Select the options as required. Code will be assigned automatically.
* According to the Sub-Class selected the other set of fields will be shown. Fill all the fields.
* Press OK when done to add the attributes to the table.

**Note:**

1. Make sure that the fields have been added to the attribute table before adding the attributes. Look at **ADD FIELDS TOOL**.
2. Make sure that the geometry of the layer selected and the geometry entered in the dialog box is the same.
3. Reload the plugin each time you select a layer to add feature attributes to. Look at **PLUGIN RELOADER.**

**How it works:**

The entire working of the tool is contained in three Python files:

**ui\_addattribute.py:** Contains the code for the UI of the dialog box

**addattributegui.py:** Links the UI components to the working logic of the dialog box

**addattributetool.py:** Reads the field names from the user input file and adds to attribute table.

Finally they are imported to the main program **quick\_digitize.py**.

**ui\_addattribute.py**

**Class Name:**

**Ui\_AddAttribute(object)**

**Methods:**

**setupUi\_1(self, Dialog)**: Adds objects with specified parameters to dialog box

**retranslateUi\_1(self, Dialog)**: Renames the labels

**setupUi\_2(self, Dialog)**: Adds objects with specified parameters to dialog box

**retranslateUi\_2(self, Dialog)**: Renames the labels

**addattributegui.py**

**Class Name:**

**AddAttributeGui(QDialog, QWidget, Ui\_AddAttribute)**

**Methods:**

**\_\_init\_\_(self, parent, flags):** Initializes variables and connects the signals of the dialog box to the mentioned slots.

**initGui(self):** Refreshes the settings and add items to Geo-Spatial and Geometry combo boxes.

**accept(self):** Contains the signals that are emitted when the various combo boxes are triggered and the slots that they connect to.

**append\_1(self):** Appends the data entered in the different fields of the dialog box into a single list which will be added to the attribute table.

**close\_1(self):** Emits signal to unset tool

**creation(self):** Creates the fields and corresponding labels by calling the setup function from Ui\_AddAttribute.

**deletion(self):** Deletes the fields and corresponding labels by calling the setup function from Ui\_AddAttribute.

**PyQt Slots:**

**assign\_value\_1(self, text):** Assigns value to the field, Geometry.

**assign\_value\_2(self, text):** Assigns value to field, Sub-Class.

**assign\_value\_3(self):** Assigns value to field, Class.

**assign\_value\_4(self, text):** Assigns value to field, Geo-Spatial Data.

**add\_attribute\_1(self, string):** Gets the Geometry value to be added to table.

**add\_attribute\_2(self, string):** Gets the Sub-Class value to be added to table.

**add\_attribute\_3(self, string):** Gets the Class value to be added to table.

**add\_attribute\_4(self, string):** Gets Geo-Spatial Data the value to be added to table.

**addattributetool.py**

**Class Name:**

**AddAttributeTool**

**Methods:**

**\_\_init\_\_(self, iface, toolBar):** Initializes variables and connects the signals of the dialog box to the mentioned slots.

**toggle(self):** Enables the plugin when layer is editable.

**newattribute(self):** Adds the attribute list imported from AddAttributesGui to the attribute table of the selected layer, for the selected feature.

**close\_func(self):** Calls close\_1(self) from AddAttributesGui class.

**NOTE:**

1. To change the labels go the respective \*\*\*\*\_labels.csv file and change the first column.
2. To add more lineEdits go to Ui\_AddAttribute.py and add to setupUi and add the corresponding label names in retranslateUi. Add the corresponding functionality in AddAttributeGui.py if required.
3. Be careful about resizing the dialog to the largest size after making any changes.
4. To add more Geo-Spatial Data Categories add to the self.geodat array in the definition of initGui() and add the corresponding data file name as ‘ ‘\*\*\*\*\_class\_subclass.csv’ to self.filelist array.
5. To add a different set of fields and labels for a different Geo-Spatial Data set, go to Ui\_AddAttribute and add definitions of methods setupUi\_n and delUi\_n (where ‘n’ is any integer greater than 2). Read labels from files named \*\*\*\*\_labels.csv, named accordingly. Make the necessary changes in conditions in the class AddAttributesGui.

**ADD POINT LAYER TOOL**

**About the tool:**

This tool adds a \*.shp (shapefile) of the the point geometry.

**How to use:**

* Load Quick Digitize plugin.
* Select the Add Point Layer.
* In the dialog box specify the name of the layer to be loaded in the QGIS layer panel and click on the push button to get the file browser dialog, and select the path and the name of the file as you want to save it.

**Note:**

A warning message will be shown which is not of concern as we will always be working on WGS 84 for a point layer.

**How it works:**

The entire working of the tool is contained in two Python files:

**createlayergui.py** : Contains the UI for the dialog box appearing after clicking the button.

**createpointlayertool.py:** Contains the working logic and the saves the point layer.

**createlayergui.py**

**Class Name**

**CreateLayerGui(QDialog, QWidget, Ui\_CreateLayer)**

**Methods:**

**\_\_init\_\_(self, parent, flags): ):** Initializes variables and connects the signals of the dialog box to the mentioned slots.

**initGui(self):** Refreshes the settings.

**Point(self):** Adds and sets the name of the layer in the QGIS layer panel and the geometry type to point.

**close\_1(self):** Emits signal to unset tool

**createpointlayertool.py**

**Class Name**

**CreatePointLayerTool**

**Methods:**

**\_\_init\_\_(self, iface, toolBar):**  Initializes variables and connects the signals of the dialog box to the mentioned slots.

**showDialog(self):** Links the dialog box UI to the functionalities of the toolbar.

**select\_output\_file(self):**Saves the file to the selected path from the file browser dialog box.

**close\_func(self):**Calls close\_1 method in class CreateLayerGui from createlayergui.py

**ADD LINE LAYER TOOL**

**About the tool:**

This tool adds a \*.shp (shapefile) of the the line geometry.

**How to use:**

* Load Quick Digitize plugin.
* Select the Add Point Layer.
* In the dialog box specify the name of the layer to be loaded in the QGIS layer panel and click on the push button to get the file browser dialog, and select the path and the name of the file as you want to save it.

**Note:**

A warning message will be shown saying the CRS is defaulted to WGS 84 , change it to any projected CRS(WGS- 84/Pseudo Mercator is optimal) while using spline tool to draw lines.

**How it works:**

The entire working of the tool is contained in two Python files:

**createlinelayergui.py** : Contains the UI for the dialog box appearing after clicking the button.

**createlinelayertool.py:** Contains the working logic and the saves the point layer.

**createlinelayergui.py**

**Class Name**

**CreateLineLayerGui(QDialog, QWidget, Ui\_CreateLayer)**

**Methods:**

**\_\_init\_\_(self, parent, flags): ):** Initializes variables and connects the signals of the dialog box to the mentioned slots.

**initGui(self):** Refreshes the settings.

**Line(self):** Adds and sets the name of the layer in the QGIS layer panel and the geometry type to line.

**close\_1(self):** Emits signal to unset tool

**createlinelayertool.py**

**Class Name**

**CreateLineLayerTool**

**Methods:**

**\_\_init\_\_(self, iface, toolBar):**  Initializes variables and connects the signals of the dialog box to the mentioned slots.

**showDialog(self):** Links the dialog box UI to the functionalities of the toolbar.

**select\_output\_file(self):**Saves the file to the selected path from the file browser dialog box.

**close\_func(self):**Calls close\_1 method in class CreateLineLayerGui from createlinelayergui.py

**ADD POLYGON LAYER TOOL**

**About the tool:**

This tool adds a \*.shp (shapefile) of the the polygon geometry.

**How to use:**

* Load Quick Digitize plugin.
* Select the Add Point Layer.
* In the dialog box specify the name of the layer to be loaded in the QGIS layer panel and click on the push button to get the file browser dialog, and select the path and the name of the file as you want to save it.

**Note:**

A warning message will be shown which is not of concern as we will always be working on WGS 84 for a polygon layer.

**How it works:**

The entire working of the tool is contained in two Python files:

**createpolygonlayergui.py** : Contains the UI for the dialog box appearing after clicking the button.

**createpolygonlayertool.py:** Contains the working logic and the saves the point layer.

**createpolygonlayergui.py**

**Class Name**

**CreatePolygonLayerGui(QDialog, QWidget, Ui\_CreateLayer)**

**Methods:**

**\_\_init\_\_(self, parent, flags): ):** Initializes variables and connects the signals of the dialog box to the mentioned slots.

**initGui(self):** Refreshes the settings.

**Polygon(self):** Adds and sets the name of the layer in the QGIS layer panel and the geometry type to polygon.

**close\_1(self):** Emits signal to unset tool

**createpolygonlayertool.py**

**Class Name**

**CreatePolygonLayerTool**

**Methods:**

**\_\_init\_\_(self, iface, toolBar):**  Initializes variables and connects the signals of the dialog box to the mentioned slots.

**showDialog(self):** Links the dialog box UI to the functionalities of the toolbar.

**select\_output\_file(self):**Saves the file to the selected path from the file browser dialog box.

**close\_func(self):**Calls close\_1 method in class CreatePolygonLayerGui from createpolygonlayergui.py