Circuit Designer Update

Kenneth Shipley, Joseph Spear, Jason Rivas

GUI:

All of the development for the GUI has been done using the PyQt library, and thus all GUI items are related to how PyQt is being utilized.

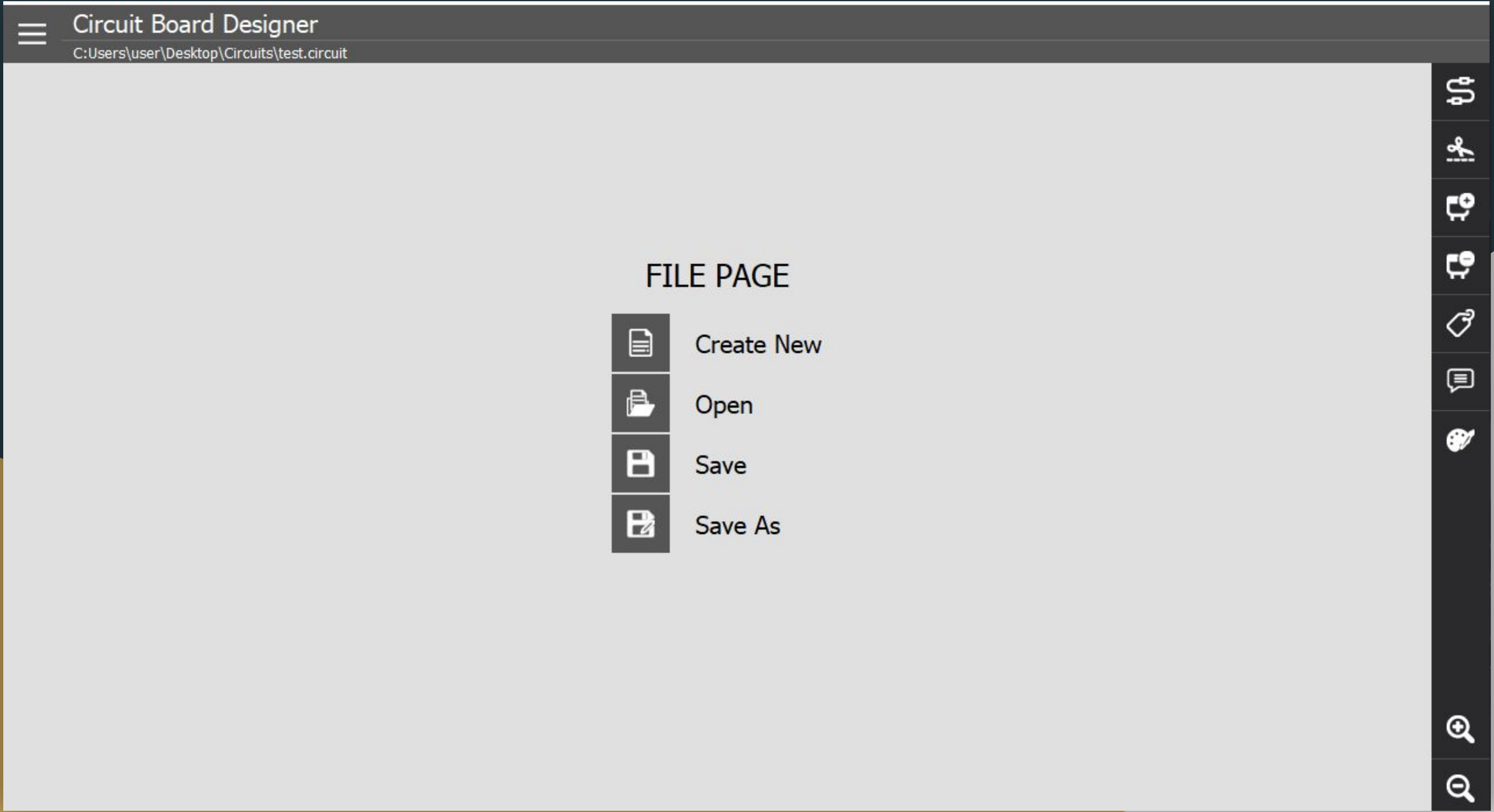
1. Completed Items
2. Partially completed items
3. Items that still need to be implemente

GUI:

1. Completed Items

The initial portion of the GUI has been completed, where the window holding all of the buttons and elements will be located. This includes a “Design Page”, a “Convert Page”, and a “File Page”. Each of these pages are made of frames, and will include the various first-glance GUI elements.

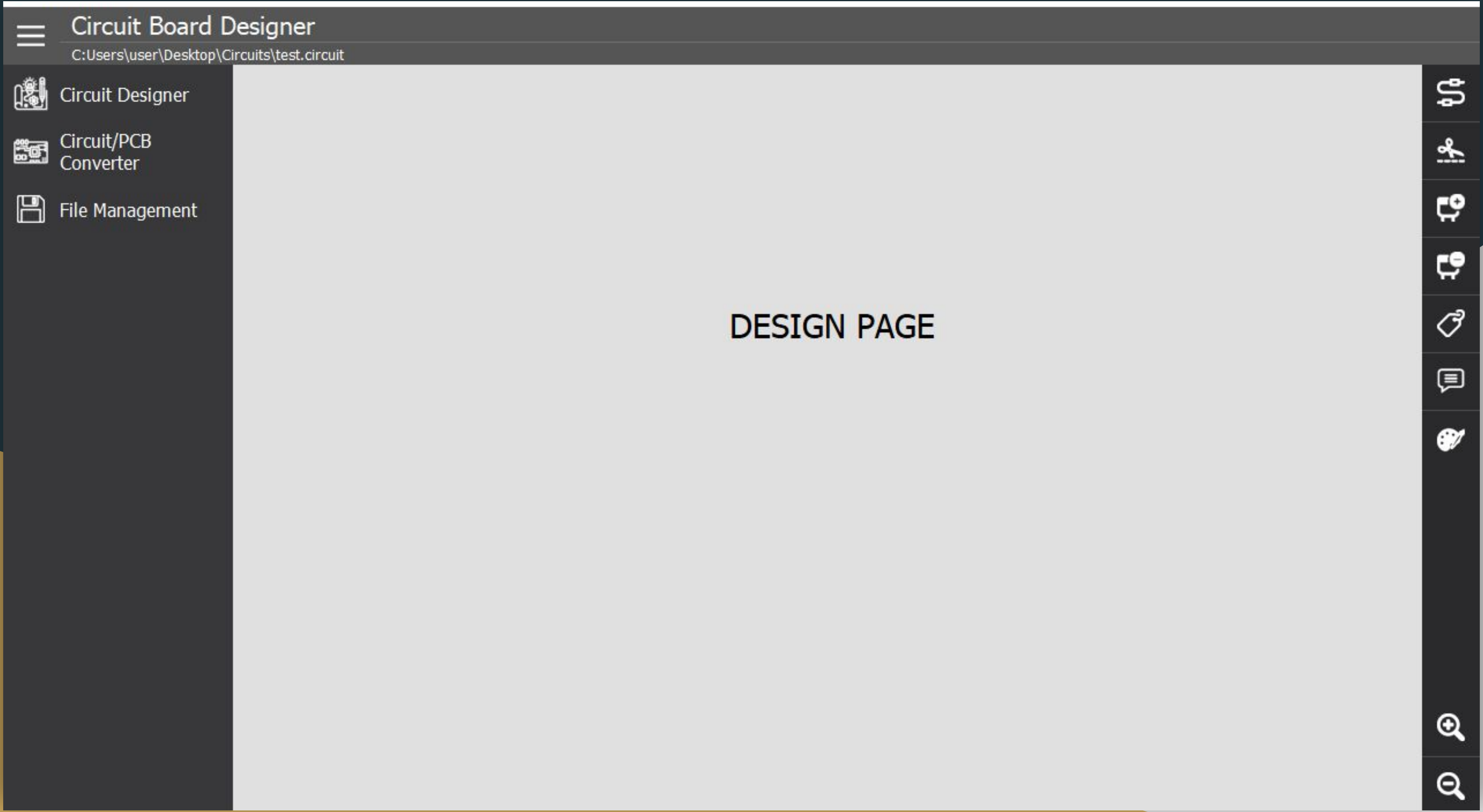
The following images have transitions between each one once the connected button is clicked.



Circuit Board Designer

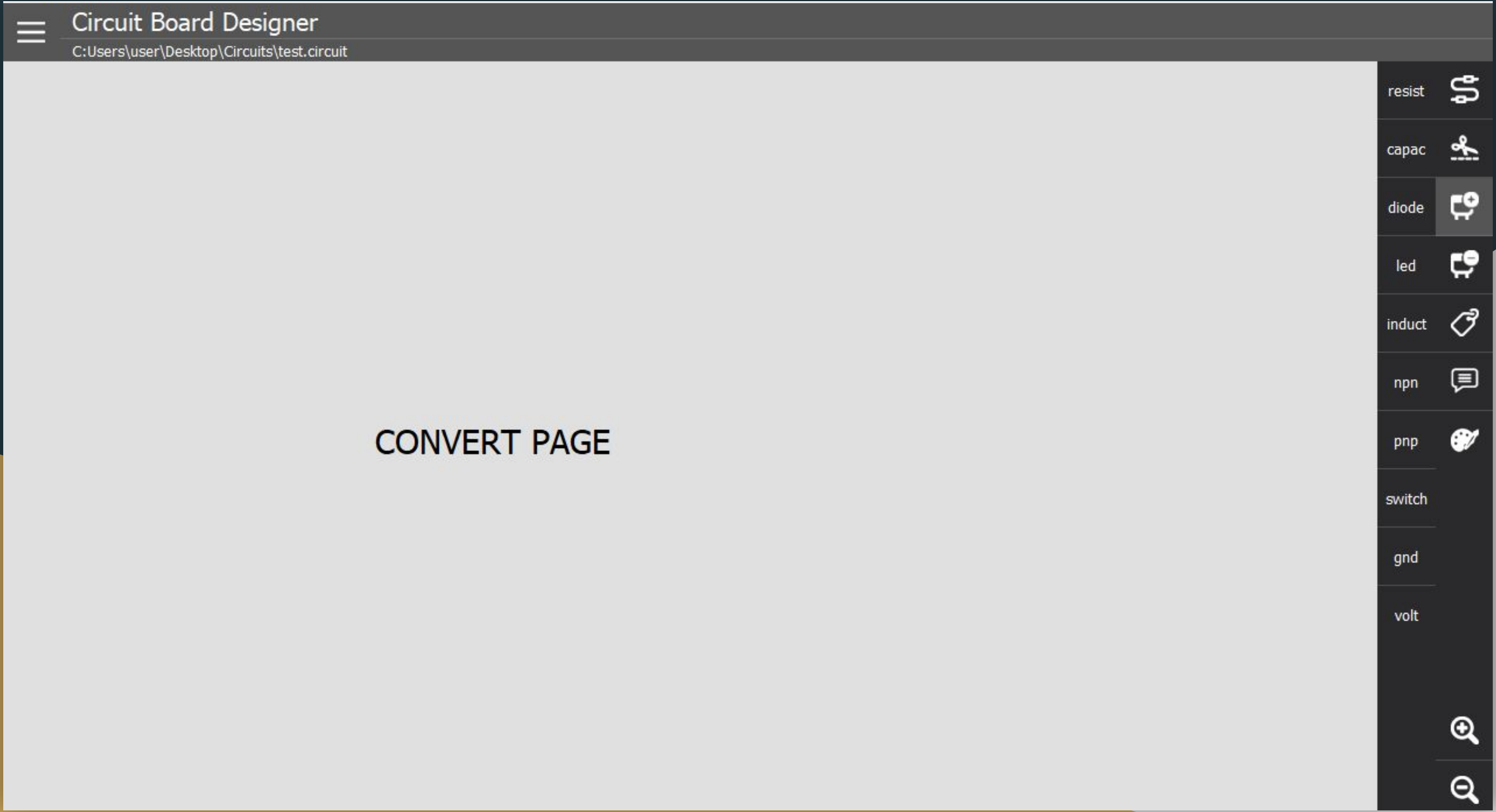
C:Users\user\Desktop\Circuits\test.circuit

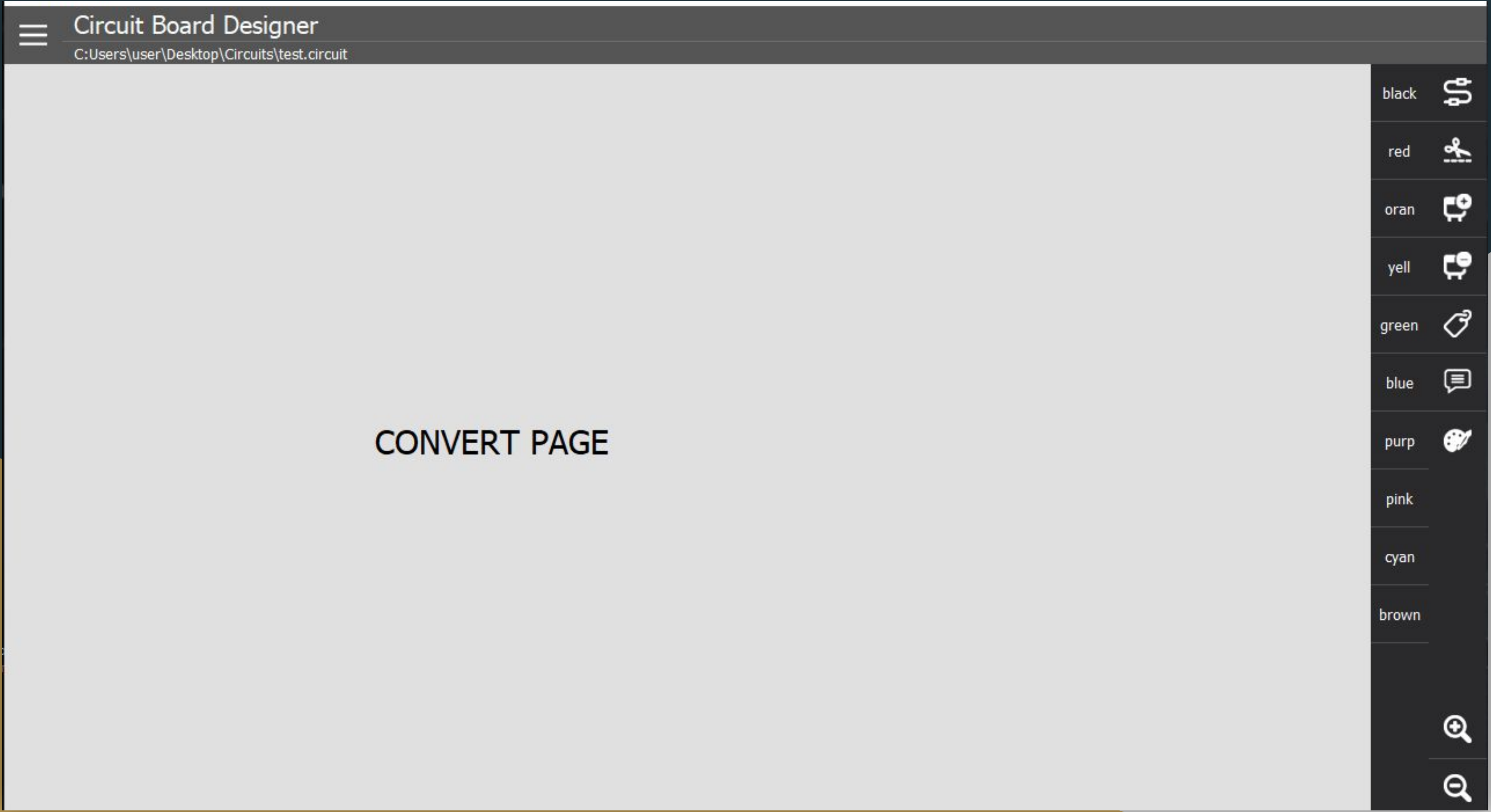
FILE PAGE



Circuit Board Designer

C:Users\user\Desktop\Circuits\test.circuit


Circuit Board Designer
C:Users\user\Desktop\Circuits\test.circuit
CONVERT PAGE
resist
capac
diode
led
induct
npn
pnp
switch
gnd
volt


Circuit Board Designer
C:Users\user\Desktop\Circuits\test.circuit
CONVERT PAGE
black
red
oran
yell
green
blue
purp
pink
cyan
brown

GUI:

2. Partially Completed Items

The partially completed items include;

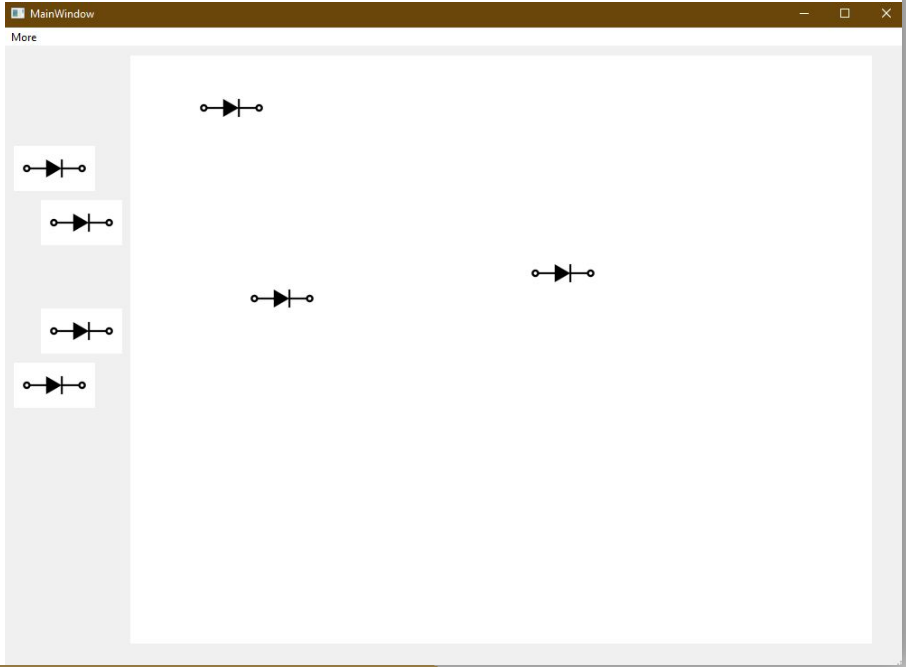
- Drag and Drop
- Coordinate acquisition
- Canvas workspace

These three items are all solutions to various needs of the GUI application, but need to integrated together.

Drag and Drop Functionality

This drag and drop functionality is done by creating a mouseClickedEvent and mouseDropEvent which helps to tell the object when to move and when not to move.

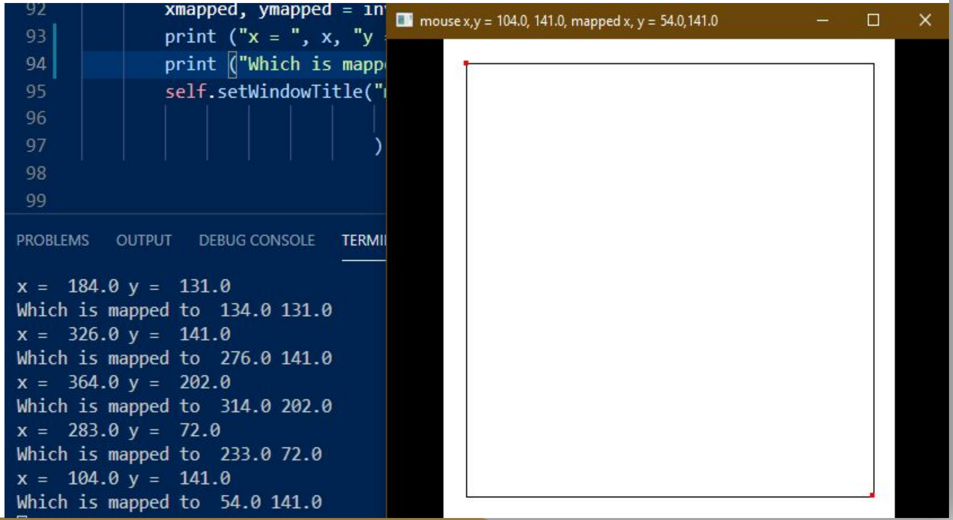
This test on the right is just an example of some diodes than started on the left and have been dragged to the right freely.



Click Coordinate Functionality

The click coordinate functionality creates a window on screen, and localizes the window coordinates relative to the screen coordinates to help translate screen location to window location.

The image on the left provides a set of examples where clicking on the white box will send the clicked coordinates to the command line output.

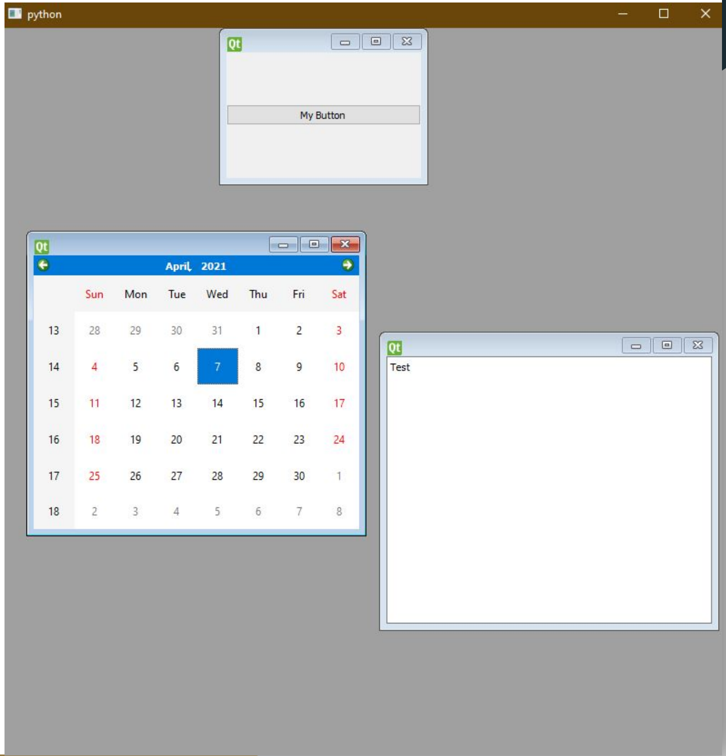

```
            xmapped, ymapped = in
            print ("x = ", x, "y
            print ("Which is mapp
            self.setWindowTitle("
                                )
```

```
x =  184.0 y =  131.0
Which is mapped to  134.0 131.0
x =  326.0 y =  141.0
Which is mapped to  276.0 141.0
x =  364.0 y =  202.0
Which is mapped to  314.0 202.0
x =  283.0 y =  72.0
Which is mapped to  233.0 72.0
x =  104.0 y =  141.0
Which is mapped to  54.0 141.0
```

Canvas Workspace Functionality

The Canvas Workspace is implemented similar to the Drag and Drop example, however, it is much more versatile and can be used to integrate the main windows seen in the completed section with the drag and drop functionalities that are seen in this section.

The example to the right show various widget which operate and can be moved around in the main window.

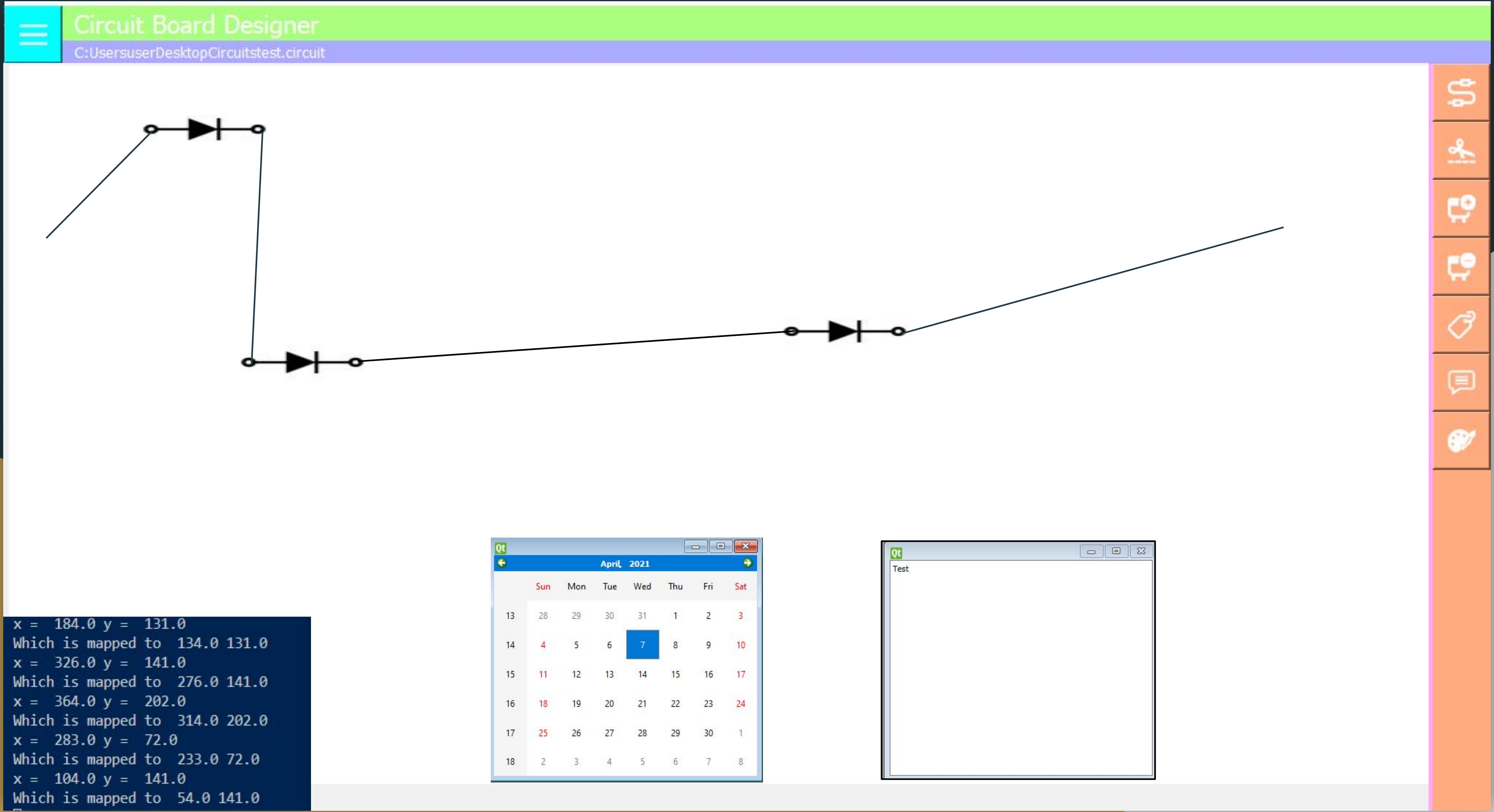


	Sun	Mon	Tue	Wed	Thu	Fri	Sat
13	28	29	30	31	1	2	3
14	4	5	6	7	8	9	10
15	11	12	13	14	15	16	17
16	18	19	20	21	22	23	24
17	25	26	27	28	29	30	1
18	2	3	4	5	6	7	8

GUI:

3. Items that still need to be implemented

The primary piece of implementation that needs to be done is integration. All of the solutions in the partially completed items section need to be implemented using one streamlined method. This then needs to be integrated with the main window functionality seen in the completed section.



Circuit Board Designer

C:UsersuserDesktopCircuitstest.circuit

```
x =  184.0 y =  131.0
Which is mapped to  134.0 131.0
x =  326.0 y =  141.0
Which is mapped to  276.0 141.0
x =  364.0 y =  202.0
Which is mapped to  314.0 202.0
x =  283.0 y =  72.0
Which is mapped to  233.0 72.0
x =  104.0 y =  141.0
Which is mapped to  54.0 141.0
```

April, 2021

	Sun	Mon	Tue	Wed	Thu	Fri	Sat
13	28	29	30	31	1	2	3
14	4	5	6	7	8	9	10
15	11	12	13	14	15	16	17
16	18	19	20	21	22	23	24
17	25	26	27	28	29	30	1
18	2	3	4	5	6	7	8

Test

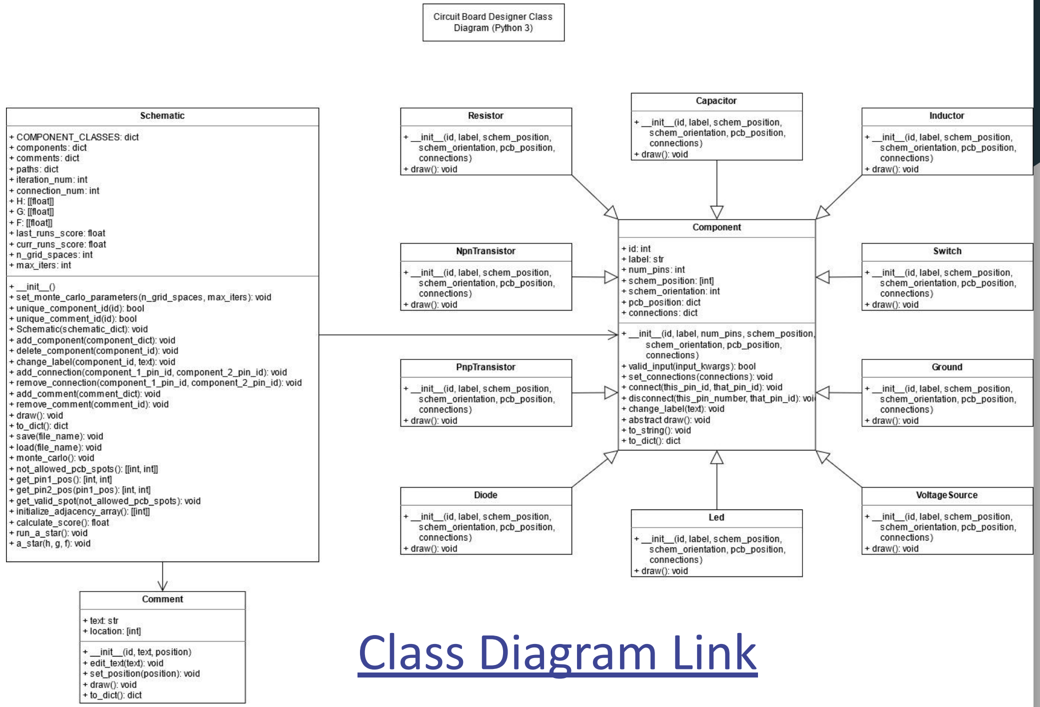
GUI:

Other smaller things are still need to be implemented are:

- Sending click coordinates to the Monte Python section to process the schematic.
- Finalized svg drawings of all of the components
- Ability to change colors of lines

Classes:

1. Completed Items
2. Partially completed items
3. Items that still need to be Implemented


Circuit Board Designer Class Diagram (Python 3)
Schematic
+ COMPONENT_CLASSES: dict
+ components: dict
+ comments: dict
+ paths: dict
+ iteration_num: int
+ connection_num: int
+ H: [[float]]
+ G: [[float]]
+ F: [[float]]
+ last_runs_score: float
+ curr_runs_score: float
+ n_grid_spaces: int
+ max_iters: int
+ __init__()
+ set_monte_carlo_parameters(n_grid_spaces, max_iters): void
+ unique_component_id(id): bool
+ unique_comment_id(id): bool
+ Schematic(schematic_dict): void
+ add_component(component_dict): void
+ delete_component(component_id): void
+ change_label(component_id, text): void
+ add_connection(component_1_pin_id, component_2_pin_id): void
+ remove_connection(component_1_pin_id, component_2_pin_id): void
+ add_comment(comment_dict): void
+ remove_comment(comment_id): void
+ draw(): void
+ to_dict(): dict
+ save(file_name): void
+ load(file_name): void
+ monte_carlo(): void
+ not_allowed_pcb_spots(): [[int, int]]
+ get_pin1_pos(): [int, int]
+ get_pin2_pos(pin1_pos): [int, int]
+ get_valid_spot(not_allowed_pcb_spots): void
+ initialize_adjacency_array(): [[int]]
+ calculate_score(): float
+ run_a_star(): void
+ a_star(h, g, f): void
Comment
+ text: str
+ location: [int]
+ __init__(id, text, position)
+ edit_text(text): void
+ set_position(position): void
+ draw(): void
+ to_dict(): dict
Component
+ id: int
+ label: str
+ num_pins: int
+ schem_position: [int]
+ schem_orientation: int
+ pcb_position: dict
+ connections: dict
+ __init__(id, label, num_pins, schem_position, schem_orientation, pcb_position, connections)
+ valid_input(input_kwargs): bool
+ set_connections(connections): void
+ connect(this_pin_id, that_pin_id): void
+ disconnect(this_pin_number, that_pin_id): voi
+ change_label(text): void
+ abstract draw(): void
+ to_string(): void
+ to_dict(): dict
Resistor
+ __init__(id, label, schem_position, schem_orientation, pcb_position, connections)
+ draw(): void
Capacitor
+ __init__(id, label, schem_position, schem_orientation, pcb_position, connections)
+ draw(): void
Inductor
+ __init__(id, label, schem_position, schem_orientation, pcb_position, connections)
+ draw(): void
NpnTransistor
+ __init__(id, label, schem_position, schem_orientation, pcb_position, connections)
+ draw(): void
Switch
+ __init__(id, label, schem_position, schem_orientation, pcb_position, connections)
+ draw(): void
PnpTransistor
+ __init__(id, label, schem_position, schem_orientation, pcb_position, connections)
+ draw(): void
Ground
+ __init__(id, label, schem_position, schem_orientation, pcb_position, connections)
+ draw(): void
Diode
+ __init__(id, label, schem_position, schem_orientation, pcb_position, connections)
+ draw(): void
Led
+ __init__(id, label, schem_position, schem_orientation, pcb_position, connections)
+ draw(): void
Voltage Source
+ __init__(id, label, schem_position, schem_orientation, pcb_position, connections)
+ draw(): void

Class Diagram Link

Classes:

1. Completed items

- Adding and deleting components to a schematic instance.
- Connecting and disconnecting component pins.
- Adding and removing comments.
- Editing component labels.

Classes:

2. Partially completed items

- Save and Load work for the Schematic class. However,
  - save() just throws an error if the file already exists.
  - load() just throws an error if the file doesn’t exist.
- Component svg images.
- The randomizing pin positions function can sometimes get stuck.

Classes:

3. Items that still need to be implemented

- Tying functions to the GUI.
  - Create New, Open, Save, Save As -> **Schematic.save(), Schematic.load(), and new Schematic()**
  - Create/Snip wire -> **Schematic.add_connection() and Schematic.remove_connection()**
  - Add/Delete Component -> **Schematic.add_component() and Schematic.remove_component()**
  - Add/Edit/Remove Comment -> **Schematic.add_comment(), Schematic.edit_comment(), and Schematic.remove_comment()**
  - Edit Label -> **Schematic.edit_label()**
  - Drag and Drop Component -> **Schematic.draw()**
- Schematic.draw()
- Clicking on a components' pin in the schematic view.
  - Implementing a basic svg editor using an xml library
- A* and the auxiliary functions it uses.

Issues to the project:

GUI Issues:

Can’t integrate the coordinates and the drag and drop functionality together.

Class Issues:

Communication with the GUI.

Converter Issues:

Unclear how to specifically implement the mapping from grid space to a physical pcb.

Potential Fix for GUI:

There is are classes in QT called qGraphicsView and qGraphicsScene that does most everything we need for the UI. It has a built-in canvas, allows objects to be easily added to a scene, and easy movement where every object has its own position.