ISTANBUL TECHNICAL UNIVERSITY

FACULTY OF ELECTRICAL AND ELECTRONICS

ELECTRONICS AND COMMUNICATION ENGINEERING

BLG252E TERM PROJECT REPORT

ALP KÖKSAL

040190760

NEVZAT KUTAY KOÇ

040180538

MEHMET FURKAN BURTGİL

040180520

GROUP NAME: TEAM

INTRODUCTION

The purpose of this project is to implement a simple logic simulator while using object-oriented programming methods. We have a space on the left-hand side where all the logic elements are, a simulation area where start and stop buttons are located and a workbench. User can select a logic element on the left side by simply clicking on it, and after dragging, droping it to the workbench by clicking again. He can select and drop logic gates to the workbench as much as he wants. There are multiple things that user can do. He can select a logic element located in the workbench and delete it. He can also select all the logic elements in the workbench by pressing the key A, and he can delete all the objects at once. All the selected objects can be unselected by pressing the esc key. If delete is pressed after esc is pressed, nothing happens since all the objects have been unselected. User can also connect two logic gates by clicking on pins of logic elements. when a pin is clicked, it gets a green circle around it and any other object's pin can be selected to connect them with wire. When an object is deleted, all of its connections also get deleted.

WORK DISTRIBUTION AMONG GROUP MEMBERS

Alp Köksal: I have done the project myself. Thus, everything written in this project report has been implemented by me. Nevertheless, below is the summary of what I have done for the project.

- Creating program logic
- Implementing all the classes and all of their methods
- Implementing the main function
- Testing to check if the program crashes

Nevzat Kutay Koç: -

Mehmed Furkan Burtgil: -

IMPLEMENTATION

CLASSES

1) Object Class

First of all, a class named Object is implemented for all the drawable objects that can be shown in the screen. All the features that might be common among other classes that derived from this class are implemented. Every instance of that class has a unique id, type and coordinate info. When working in a project like this one, it is very easy to work with the objects having an id when we want to select a specific element and do something with it. In this project, we need the information of what object gets selected when user interacts with program. Type is another variable that refers to type of an object such as and gate or xor gate. We also need the information of coordinates of a specific element and that is why we need an array for storing coordinates. All objects have one or multiple textures and a sprite so that is why we need those variables. We also need a variable that stores a boolean value that refers to whether an object is selected or not. A few methods are implemented under the object class such as a method for setting the coordinates for given parameters or a method for getting the coordinates of the object. The constructor of the class takes 3 arguments which are a pointer to window object and two strings indicated the type of the object. Third parameter is optional and has a default value of empty string. This parameter is for objects with multiple textures such as led element. For example, the arguments of the constructor might be pointer to a window object and “AND”. When it is called with these arguments, the constructor loads the image of AND gate under assets folder to texture and sprite uses it. If led element is created, default texture is a led image while it is off. It can be changed to on using a method called changeSprite which is declared as a virtual method. LED class itself overrides this method and changes the image of led to on if it is off and to off if it is on.

2) Node and ObjectList Classes

Linked List

In order to store all the objects dynamically, we need a linked list. First a node class is implemented. It includes a pointer to another node, and a pointer to object. In Figure 1, visual representation of a node can be seen.

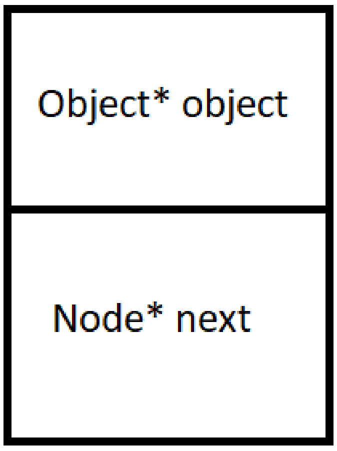

Figure 1. A Node of Linked List

ObjectList

After the node class is implemented, we can now implement the linked list. Since it stores object instances, the name of the class is called objectlist. Everything related to objectlist is defined under a header file named objectlist.h and implemented under a .cpp file named objectlist.cpp. Objectlist is actually not a linked list.

Objectlist is a class that includes two different linked lists as well as some other attributes. Head pointer of first linked list is called head. This pointer points to the head of the linked list where all the logic elements on the workbench are stored. This is the main link list. All the objects in the workbench are related to this one. If user selects and gate from left side and put it into workbench, an andgate object is created and added to the linked list. If user deletes something, it gets deleted from the linked list.

Another linked list is used for showing default objects on the left side that user can choose and drop to workbench. This linked list has 9 logic elements by default and we don't add or delete anything from the linked list. The purpose of it is to show default objects. Pointer that points to the head node of the linked list is called default_objects_head. A visual representation for link lists can be seen in Figure 2.

We also have an attribute named current. It is a pointer to an instance of the object class and it indicates the object that is selected from the left side but it is not dropped to workbench yet. Whatever object is selected from the left side, that logic element is created and if user drops it to workbench it is added to first linked list whose head pointer is named as head. There is also an attribute called carry that is a variable that stores boolean value and indicates whether selected element is dropped to workbench or not. Its value is false if current which is the selected element is dropped and it is true if the element is selected but not dropped to the workbench yet. There are attributes for both start and stop buttons.

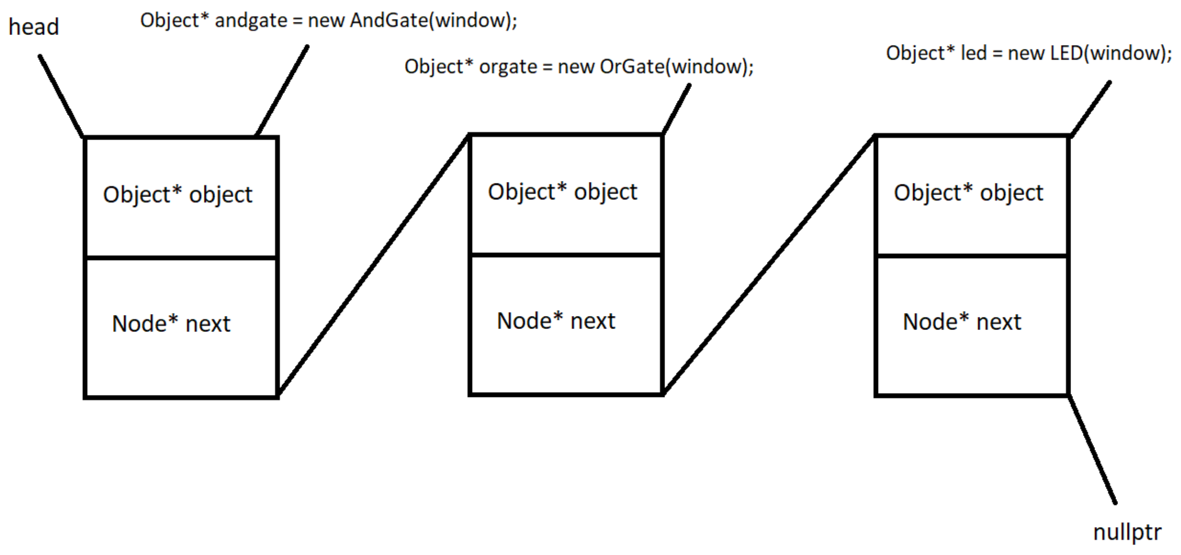
head
Object* andgate = new AndGate(window);
Object* orgate = new OrGate(window);
Object* led = new LED(window);
Object* object
Node* next
Object* object
Node* next
Object* object
Node* next
nullptr

Figure 2. An example of a linked list that might be used in project.

ObjectList Methods

There are many methods declared in this class. It would be pointless to explain what each method does so major methods will be explained here.

Note: Since there are two linked lists here, from now on the term linked list is going to be referring to the linked list whose head pointer is named as head. The one that is used to show default logic element is going to be called default linked list.

Since we are dealing with linked lists, there are some methods related to it such as adding and deleting nodes from a link list. Constructor of Object List initializes the linked list by setting head to nullptr. Purpose of methods can be understood by their names.

To show default elements to the left side, a method named set_default_objects is called first. All the default objects such as logic gates, leds and start stop buttons are created. In the while loop, show_default_objects is called.

A method named length is used to get linked list length. It takes a default parameter and it is a boolean value and false by default. It gives the length of the linked list. If argument is true, then it gives the length of default linked list.

There are also methods to check whether user clicks on the left side, simulation area or workbench. is_left method checks if user clicks on the left side where default objects are located. is_workbench method checks if user clicks on the workbench and is_simulation method checks if user clicks on simulation area. There could have been one method to check all of these but it separated into three methods.

There are methods related to current which is a pointer to an object and refers to the object that is selected from the left but is not dropped to the workbench. The methods related to current do different things such as creating whatever logic element user has selected and assigning it to current or setting coordinates of current.

is_available method checks if where user wants to drop the selected logic element is available or not. Logic elements can not be placed in the same place and there must be some distance.

3) LogicElement Class

A class name LogicELement is implemented for describing a general logic element. A logic element is a drawable object, thus this class is inherited from object class and the type of the inheritance is public inheritance. So, logic element class has everything that object class has with more properties. This class includes all the common attributes of logic elements. This class has 2 attributes. First one is an array of pointers to Pin class. the other one is the number of pins the element has. The class has only one method and a constructor. The parameters are the same as parameters of object class. The parameters are given to object constructor using member initializer. The only method of the class is called handle_pin and as the name refers, when user selects logic element from the left side and put it to workbench, this function gets called and it sets current positions of all pins.

When user selects an object from the left-hand side, and drops it into workbench, a method named add_current is called. It is a method from ObjectList class. It simply takes coordinates of current and sends them to another method called handlePins in Object class through current. handlePins is a virtual method and declared in Object class. However, current is a pointer to an object class but it points to a Logic gate class such as a class for and gate or led (LED is not a logic gate but that term is used due to possible confusion with LogicElement class which is a different class). handlePins method of that logic gate's class is called and it also called another method which is handle_pins that is declared in LogicElement class. It sets all the coordinates of that logic element's pins.

For example, if user selects an xor gate and drops it into workbench, current is assigned to a new created xor gate and when user drops it, the handlePins method of relevant logic element class is called for setting all the pins' coordinates. That method calls handle_pins method in LogicElement class.

4) Logic Elements Class (AndGate, OrGate etc. – different from LogicElement class)

Different classes for each logic element have been implemented. All classes inherit from LogicElement class. Each class except LED has one constructor and one method which is called HandlePins. LED class has one attribute for storing the information of whether it is on or off. It also has one more method which is changeSprite and it simply changes the sprite by setting the other texture.

5) Pin Class

There is a class for describing pins. Two important attributes of pin class which are parameters for its constructor are pin type and index. Pin type indicates whether it is a pin for input or output. Index indicates whether it is the first pin of logic gate or second etc. Index start from 0 to the total number of pins the logic element has -1. 0 corresponds to upper left side and the biggest index corresponds to lower right side. Since the class has pointers as attributes, a destructor is implemented.

There are many static members in the pin class. There is another heading that explains how connections between logic elements work and it all uses these static members. Since they are all explained inside the heading, it would be pointless to mention about them in here.

Hierarchy between classes can be seen in Figure 3.

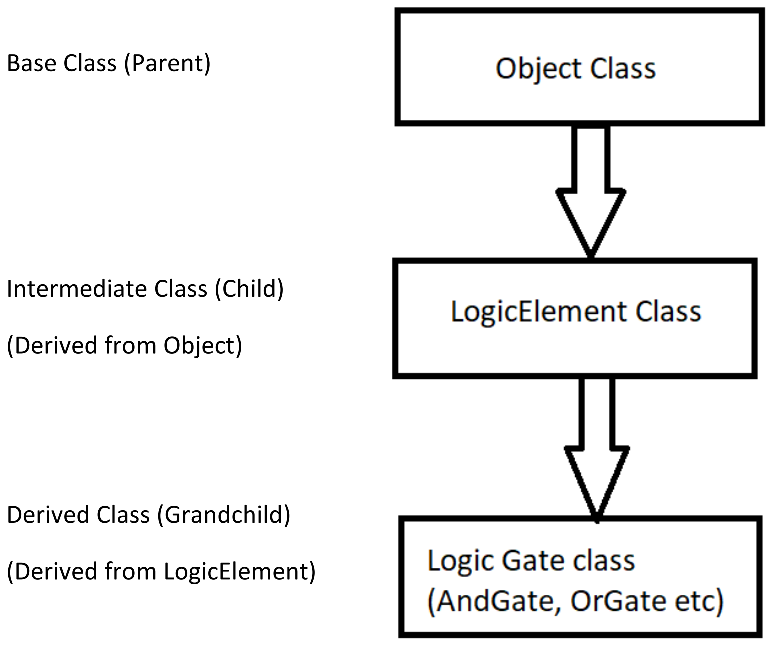

Figure 3. Hierarchy between implemented classes

HOW TO DRAW LOGIC ELEMENTS, WIRES AND MORE TO THE SCREEN

1) How to draw logic elements and start-stop buttons

Object class has a window attribute which is a pointer to a window object. It also has its own draw method which uses the draw method of window object.

show_default_objects method in ObjectList is called after window.clear is called. It simply iterates over the linked list that includes all the default objects and calls their draw method. As stated, draw method simply calls window.draw and it draws the object to the screen. That is how all the default objects on the left side is drawn.

Then show_everything method gets called. It also belongs to ObjectList class. It draws all the objects on the workbench and the simulation area. It iterates over the linked list which stores all the objects that is in the workbench and calls each object's setPosition method. This method is implemented in Object class and simply uses setPosition method of sprite object. Same thing is done for both start and stop buttons since they are also inherited from object class. their positions get set and draw methods are called as other objects. That is how logic elements in workbench and start and stop button in simulation are drawn.

Then selected_shape method is called. What this method does is it simply iterates over the linked list and checks if objects are selected or not. If true, then four rectangle shapes are created and placed in such a way that it surrounds the object. That is how rectangles around objects are drawn.

2) How do connections between logic elements work

Most of the static members of Pin class are used for connections. An array named wires_array is used to keep track of each connection. A linked list should have been used for that specific purpose but a 2D static array is used due to simplicity. It can be used for 50 connections but the size can be changed easily. Each element of the array consists of 6 sub elements: pin coordinates of the logic element that is connected from, pin coordinates of the logic element that is connected to and their ids.

When a pin is clicked and no other pins are selected so far, the coordinates of that pin is stored in the static attributes of pin class as well as id of the logic element that the pin belongs to. After the pin is selected, if another logic element's pin gets selected, the wire is drawn.

Here is the detailed explanation of how connections are implemented

When user clicks on anywhere in screen, a method in ObjectList called selectedObject gets called. It checks if any logic element in workbench is selected by looping thorough the linked list. If any logic element is selected, a static method of pin class named clicked_pin_index gets called. It checks if any of the selected object's pin is selected. If a pin is selected, then it checks whether it is the first pin that has been selected or not by checking static attribute is_pin_from_selected. It is a boolean value indicating whether a pin is selected but no other pins are selected. Then the same process is repeated if user clicks on a pin. But this time other condition is true since we have already a pin selected. It means that, user has already selected a pin and he selects a pin one more time. Thus, we can now connect the two pins. Coordinates of selected pins are stored in static attributes and the method named wires_array_add is called. As its name refers, it adds the info to the array. So, we have coordinates of the pin that is to be connected from, coordinates of the pin that is to be connected to and ids of two logic elements.

Here is the explanation of how to draw wires and circle shape for indicating selected pin

To draw wires and a shape indicating that the first chosen pin is selected a method named selected_shape in ObjectList is called. It is also used to draw rectangle shape to indicate selected objects.

It simply gets the coordinates of the selected pin and create a circle shape having a color of green and draw method of window using ObjectList's own window attribute that is a pointer to a window object.

Then it iterates over the wires_array and for each coordinates pair it creates and draws a wire. To create a line, sf::Vertex class has been used.

When esc is pressed, as rectangles around objects getting deleted, the green circle is also deleted.

When delete is pressed, all the wires of deleted objects are also getting deleted by using the id info in wires_array.

PROGRAM EXPLANATION – STEP BY STEP EXPLANATION OF WHAT HAPPENS INSIDE MAIN FUNCTION

1) Create a window with decided height and width values
2) Set the framelimit, it can be changed but 60 is enough
3) Create an instance of ObjectList and call set_default_objects to set default objects that are shown in the left side,
4) A while loop is created and it will stay opened until user clicks on the close button
5) Create an event object to check all the events while user interacts with window
6) If user clicks on close button, window gets closed and program is terminated
7) Check for key press events
8) If user presses delete, all the selected objects in the workbench get deleted meaning that they will get deleted from the linked list and they will no longer be seen to the screen
9) If user presses escape, all the selected objects and pins (pin class does not inherit from object class) on workbench get unselected
10) If user presses A, All the objects in the workbench will get selected.
11) Check for mouse press events
12) Take the information of where user is clicked and store the coordinates in the variables named x and y
13) Call selectedObject method from ObjectList class. This method iterates over the linked list and checks whether or not any object is selected. If user selects an object, that object's selected attribute is set to true, if it is already selected, then the object's selected attribute is set to false. An important point here is if user selects a pin, the object that the pin belongs to does not get selected. User should be click on the object but not on the pin in order to select that object. If a pin is selected, it gets a green circle around it and if user clicks on another object's pin then a wire is drawn between two objects to connect the objects.
14) Check if user drags object and click on workbench that is available to drop the dragged object. If true, add that object to the linked list meaning that the object is now dropped to the workbench. then set carry to false meaning that there is no object that is selected from the left and is being dragged but not dropped to workbench yet.
15) Check if user clicks on simulation and no object is being carried. If true, call the method called simulation_clicked. this method belongs to objectlist class and checks if start button or stop button is clicked. If one of them is clicked, it starts or stops simulation.
16) Check if left area is clicked meaning that user selects and object. If true, call set_current method. The method takes x and y coordinates as arguments and decides which element is selected, create the logic element with same type and assign it to current. Current is a pointer to an object which is used to store current logic element's info. In example, if user clicks on the left side and selects an and gate, an and gate is created using createObject method and a pointer to an object is returned by this method and whatever is returned gets assigned to current. is_left method sets carry to true if user clicks on left side and selects an object.

17) Check for mouse move events
18) Check if carry is true which means that user selects an object from the left side but does not drop it to the workbench yet, if carry is true and current is not a nullptr set current's coordinates everytime user moves his mouse. this way we can see selected object is dragged and follows the cursor.
19) Clear background and set its color to black
20) Call show_default_objects method from objectlist. this method shows all the default objects located on the left side
21) Call show_everything method from objectlist. This method draws all objects from the linked list, current and start and stop buttons to screen.
22) Call selected_shape method from the objectlist. This method draws red rectangles around the selected objects and a green circle around selected pin.
23) Call display method from window object to display everything drawn

Flowchart of the program can be seen in Figure 4.

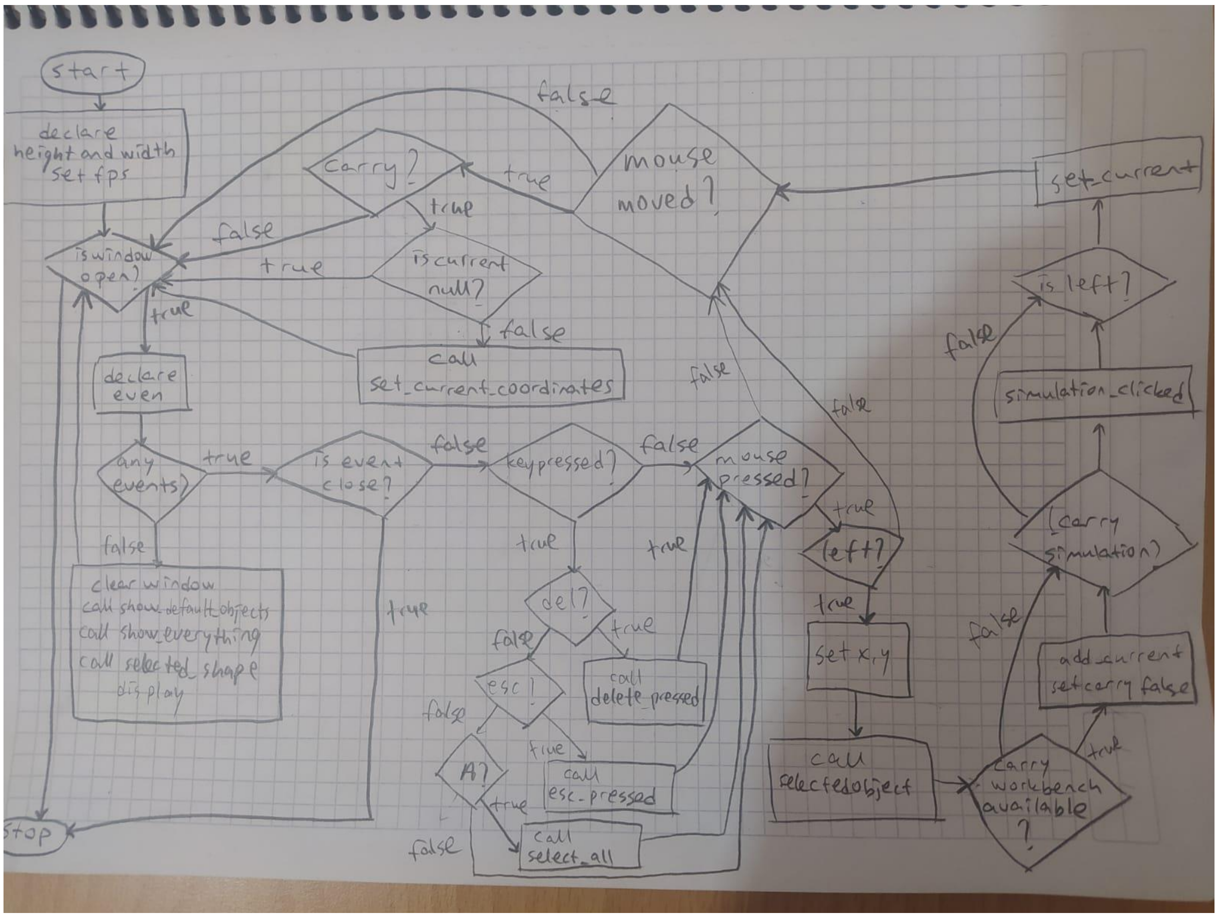

Figure 4. Flowchart of the program

A screenshot of the project can be seen in Figure 5.

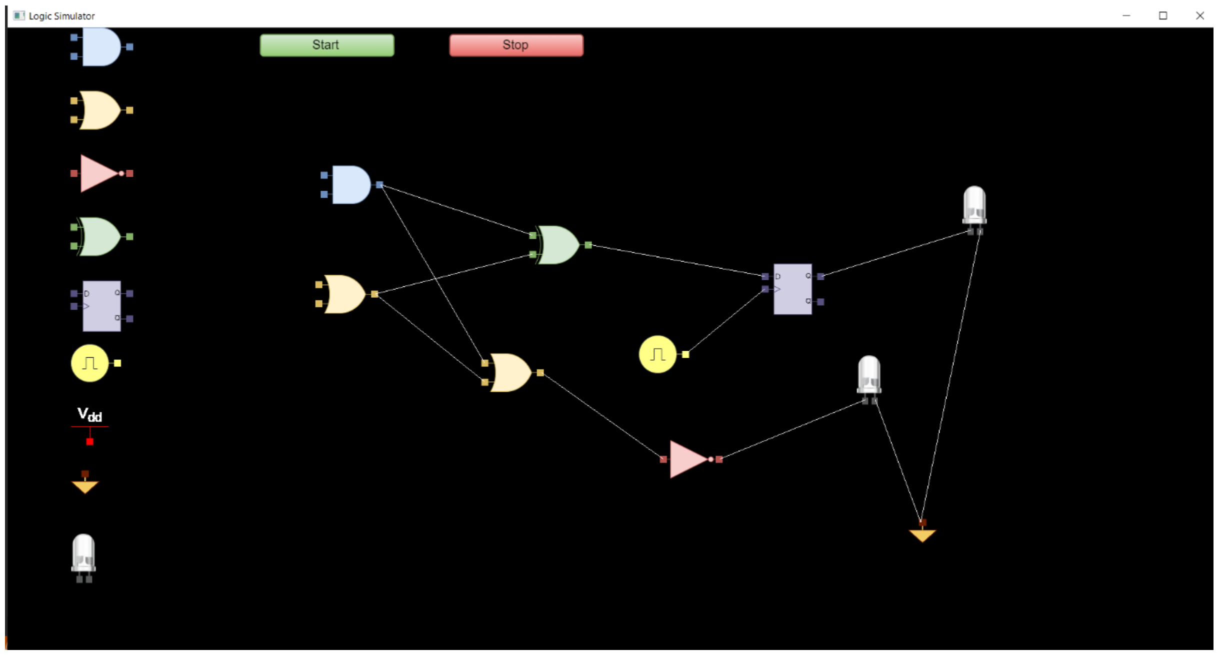

Figure 5. Logic Simulator

DISCUSSION

I did not face any problems during project and it was easier than I thought it was going to be.

One problem might be inheritance. It was the hardest part for me to do since I had never worked on a project using OOP before. The OOP concepts can be hard to grasp at first glance, especially when during the first challenging project.

Another problem is that my teammates had a hard time trying to understand the project so they could not do anything about the part where I asked them to implement; thus, I had to finish the project all by myself.

But it was really fun and educational, I really enjoyed it.