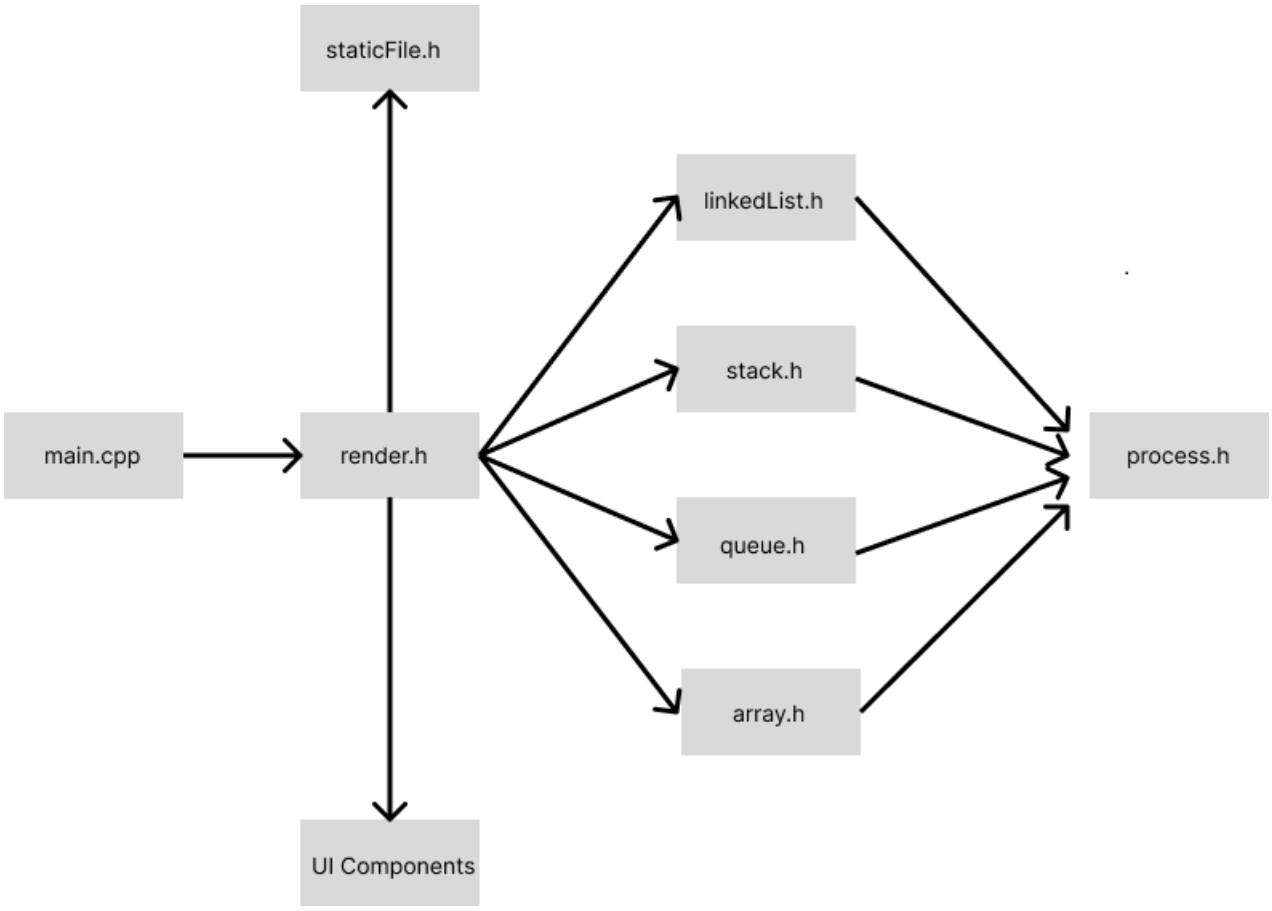
Data Visualization - Solo Project CSC10002 Documentation

This is my solo project in the Programming Technique course, also known as CSC10002. Its content is to clone the website https://visualgo.net to present basic structures such as Linked List, Stack, Queue, Array.

In this project, I use SFML to create GUI. You can read more about SFML at here: https://www.sfmldev.org/documentation/2.5.1

Project Structure

I have a total of 11 main header files and 4 header files to render each data structures, which are separated from "render.h". Additionally, there is a main file to call all of them. The workflow looks like this:



The functions in header files are defined in .cpp files with the same name. However, there are some functions in the .cpp files that are not declared in the header files because they are only used within the file itself. Therefore, I will list all the functions that exist in the .cpp files.

render.h

This file is used to render the screen, and it is devided into 4 files: renderLinkedList.h, renderStack.h, renderQueue.h, and renderArray.h to present each data structure.

Function Name	Usage	File
Sprite addSprite()	Adds a sprite to the window with the specified properties.	render.h
void displayText()	Displays text on the window with the specified properties.	render.h
textBox displayTextBox()	Displays a text box on the window at the specified position.	render.h
void drawSpeedBox()	Draws a speed box on the window.	render.h
void goAndColor()	Performs operations on a linked list and applies	render.h
Function Name	**Usage**	**File**
	color highlights.	
void insertCode()	Inserts code into the window from a specified file.	render.h
void resizeSprite()	Resizes a sprite to the specified height and width.	render.h
int homePage()	Displays the home page and returns the ID of the next screen.	render.h
int linkListPage()	Displays the linked list page and returns the ID of the next screen.	renderLinkedList.h
int LinkList()	Implements the linked list and returns the ID of the next screen.	renderLinkedList.h
int Stack()	Implements the stack and returns the ID of the next screen.	renderStack.h
int Queue()	Implements the queue and returns the ID of the next screen.	renderQueue.h
int arrayHomePage()	Displays the array home page and returns the ID of the next screen.	renderArray.h

int Array()	Implements the array and returns the ID of the next screen.	renderArray.h

process.h

This file is used to handle all events.

Function Name	Usage
`int statusHomePage()`	Handle events from the Home Page screen.
`int statuslinkListPage()`	Handle events from the Linked List Home Page screen.
`int statusLinkList()`	Handle events from the Linked List Visualization screen.
`int statusStack()`	Handle events from the Stack Visualization screen.
`int statusQueue()`	Handle events from the Queue Visualization screen.
`int statusArrayPage()`	Handle events from the Array Home Page screen.
`int statusArray_SD()`	Handle events from the Array Visualization screen.
Function Name	**Usage**
`void initState()`	Reinitialize some global variables when switching screens.
`void initPressState()`	Reinitialize some global variables when clicking a button.
`bool hoverMouse()`	Determine if the mouse is hovering over a sprite.
`bool Press()`	Determine if a sprite has been clicked.

linkedList.h

This file is used to handle various operations related to Linked List.

SinglyLL Class

SinglyLL is a class that defines a node in a singly linked list, with method to modify the position, color, data,... of the node and draw it on a render window.

Method Name	Usage

`SinglyLL(int, int, Font&)`	Constructor to create a new node with the given value, index, and font.
`void changePosition(Vector2f)`	Method to update the position of the node.
`void changeColor(Color)`	Method to update the color of the node.
`void changeRadius(double)`	Method to update the radius of the node.
`void changeData(int)`	Method to update the data of the node.
`void changeDes(string, bool = false)`	Method to update the description of the node.
`void draw(RenderWindow&)`	Method to draw the node on the given RenderWindow.
`bool rightPlace(Font&)`	Method to check if the node is in the correct position according to the font.
`string getDes()`	Method to get the description of the node.
`Vector2f getCenter()`	Method to get the center position of the node.

Property	Usage
`SinglyLL *nxt`	Pointer to the next node in the list.
`int id`	Unique ID of the node.
`int data`	Data value stored in the node.
`Vector2f position`	Position of the node.
`CircleShape m_node`	Circle shape object representing the node.
`Text m_text`	Text object representing the data value of the node.
`Text description`	Text object representing the description of the node.

Functions

There are some functions that are called from renderLinkedList.cpp to process user request.

Function signature	Description

`SinglyLL *createNode()`	Creates a new SinglyLL node with the given value, index, position, and font.
`void deleteLL()`	Deletes all nodes in the SinglyLL linked list starting from the root node. Updates both root and tail pointers.
`void createLL()`	Creates a new SinglyLL linked list with the given number of nodes, values, and font.
`void drawReturnLine()`	Draws an arrow to indicate that the linked list is circular.
`void drawLL()`	Draws the SinglyLL linked list on the given window.
`void insertBefore()`	Inserts a new node with the given value and index before the root node.
`void deleteBefore()`	Deletes the node before the node with the given index.
`void changeIndex()`	Increases the index of all nodes starting from the given node.
`void insertLL()`	Inserts a new node with the given value and index into the SinglyLL linked list.
`void deleteNodeLL()`	Deletes the node with the given index from the SinglyLL linked list.
Function signature	**Description**
`bool format()`	Moving the nodes to their designated positions based (create the motion effect).
`void clearColorLL()`	Remove all effects from the nodes.

stack.h

This file is used to handle various operations related to Stack.

StackVisualize Class

The StackVisualize class provides functionality to visualize and manipulate a stack data structure.

Method	Usage
`int size()`	Returns the size of the stack.
`void create(int n, int val[], Font &font)`	Creates the stack with the specified size and values.

`void push(int x, Font &font, bool Create = false)`	Pushes an element onto the stack.
`void pop()`	Pops the top element from the stack.
`void clear()`	Clears the stack.
`void changeColor(Color color)`	Changes the color of the stack visualization.
`int format(int type = 1)`	Formats the visualization style of the stack.
`void changePosition(Vector2f pos)`	Changes the position of the stack visualization.
`void draw(RenderWindow &window)`	Draws the stack visualization on the specified window.

Property	Usage
`RectangleShape node[12]`	Array of RectangleShape objects representing stack nodes.
`Text t_val[12]`	Array of Text objects representing node values.
`int l`	Integer variable representing the left index of the stack.
Property	**Usage**
`int r`	Integer variable representing the right index of the stack.
`int a[12]`	Array of integers representing the stack elements.

queue.h

This file is used to handle various operation related to Queue.

QueueVisualize Class

The QueueVisualize class provides functionality to visualize and manipulate a queue data structure.

Method	Usage
`QueueVisualize()`	Constructor for creating a `QueueVisualize` object.
`int size()`	Returns the size of the queue.
`void create(int n, int val[], Font &font)`	Creates the queue with the specified size and values.

`void enqueue(int val, Font &font, bool Create = false)`	Adds an element to the back of the queue.
`void dequeue()`	Removes the front element from the queue.
`void clear()`	Clears the queue.
`void changeColor(Color color, int peekPos)`	Changes the color of the queue visualization.
`bool format(int type = 1)`	Formats the visualization style of the queue.
`void changePosition(int id, Vector2f pos, bool status = true)`	Changes the position of a node in the queue.
`void draw(RenderWindow &window)`	Draws the queue visualization on the specified window.

Property	
`int l`	Integer variable representing the left index of the queue.

Property	
`vector<int> a`	Vector of integers representing the queue elements.
`vector<Text> t_val`	Vector of Text objects representing node values.
`vector<RectangleShape> node`	Vector of RectangleShape objects representing queue nodes.

array.h

This file is used to handle various operation related to Array (static and dynamic).

ArrayVisualize Class

The ArrayVisualize class facilitates visualizing and manipulating arrays, offering functionality for adding, deleting, resizing, and modifying element values.

Method	Usage
`ArrayVisualize()`	Constructor for creating an `ArrayVisualize` object.
`void clear()`	Clears the array.

`void create(int n, int val[], Font &font)`	Creates the array with the specified size and values.
`void createNode(int id, int val, Font &font, bool dummy = false)`	Creates a node in the array with the specified ID and value.
`void resize(int n, Font &font)`	Resizes the array to the specified size.
`void add(int x, Font &font)`	Adds an element to the array.
`void changeColor(Color color, int idx)`	Changes the color of the array element at the specified index.
`void changeData(int id, int val)`	Changes the value of the array element with the specified ID.
`void copyData(int idx1, int idx2)`	Copies the value from one array element to another.
`void changeStatus(bool statusArray)`	Changes the status of the array.

Method	Usage
`bool addMore()`	Checks if more elements can be added to the array.
`bool full()`	Checks if the array is full.
`int getData(int idx)`	Returns the value of the array element at the specified index.
`void del()`	Deletes an element from the array.
`bool draw(RenderWindow &window)`	Draws the array visualization on the specified window.

Property	Usage
`vector<int> a`	Vector of integers representing the array elements.
`vector<RectangleShape> node`	Vector of RectangleShape objects representing array nodes.
`vector<Text> t_val, idx`	Vectors of Text objects representing node values and indices.
`int r`	Integer variable representing the right index of the array.
`int capacity`	Integer variable representing the capacity of the array.

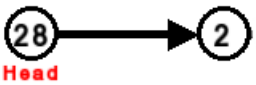
`bool vectorType`	Boolean variable indicating the type of vector used for array visualization.

UI Components

SFML doesn't have built-in support for drawing text boxes, arrows and other shapes, so I need to create them myself using drawing tools and functions provided by SFML.

arrow.h

The arrow is used to pointing from one node to another node in Linked List. I draw a rectangle and append a triangle at the end, forming a complete arrow.



Method	Usage
`Arrow(int num = 1)`	Constructor for creating an `Arrow` object with an optional `num` .
`void create(Vector2f startPoint, Vector2f endPoint, bool change)`	Creates an arrow shape from the given start and end points.
`void createLine(Vector2f startPoint, Vector2f endPoint, bool change)`	Creates a line shape from the given start and end points.
`void createArrow(VertexArray &arrow)`	Creates the arrow shape from a given `VertexArray` .
`void draw(RenderWindow &window)`	Draws the arrow on the specified `RenderWindow` .

Property	Usage
`int numArrow`	Integer variable representing the number of arrows.
`VertexArray line`	VertexArray object representing the line shape of the arrow.
`VertexArray arrow1`	VertexArray object representing one side of the arrow shape.
`VertexArray arrow2`	VertexArray object representing the other side of the arrow shape.
`Vector2f point1`	Vector2f representing the start point of the arrow.

`Vector2f point2`	Vector2f representing the end point of the arrow.

textBox.h

The text box is used to handle the user input and check its validity.

Method	Usage
textBox(Vector2f posChatBox, Texture &t_submitButton, string descrip, Font &font)	Constructor for creating a textBox object.
void handleInput(RenderWindow &window, Event event)	Handles user input events.
void click(RenderWindow &window, Event event)	Handles click events on the textBox.
Method	**Usage**
void submit()	Submits the text entered in the textBox.
void draw(RenderWindow &window)	Draws the textBox on the specified window.
RectangleShape chatbox	RectangleShape representing the chatbox.
Text text	Text object for displaying entered text.
Text note	Text object for displaying additional notes.
Text description	Text object for displaying description text.
Sprite submitButton	Sprite representing the submit button.

highLight.h

The highlight is a rectangle that used to highlight the corresponding line of code.

Highlight Class

The Highlight class is used to create a rectangle that represents a highlight object.

Method	Usage
Highlight()	Constructor for creating a Highlight object.
void makeHighlight(int Line, Color color)	Creates a highlight at the specified line with the given color. If no color is provided, a default color is used.
void line(int x)	Sets the horizontal position of the highlight line.
void draw(RenderWindow &window)	Draws the highlight on the specified window.
bool display	Indicates whether the highlight should be displayed or not.

Property	Usage
RectangleShape highLight	RectangleShape representing the highlight.

Functions

There are functions that help to highlight corresponding line of code.

Function	Usage
void highlightInsertCode(Highlight &highlight)	Highlights the code for an insert operation.
void highlightDeleteCode(Highlight &highlight)	Highlights the code for a delete operation.
void highlightUpdateCode(Highlight &highlight)	Highlights the code for an update operation.

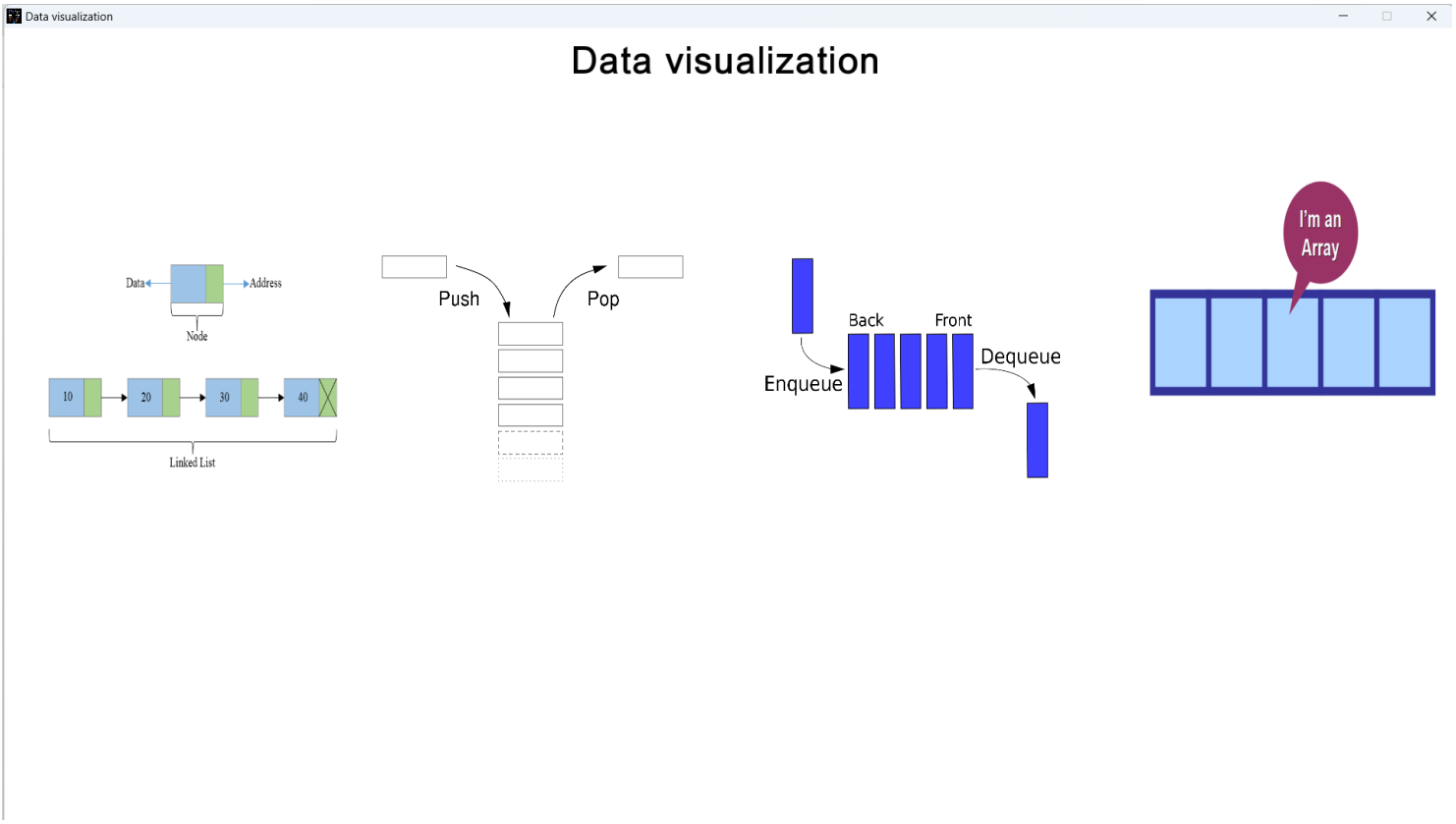
void highlightSearchCode(Highlight &highlight)	Highlights the code for a search operation.
void highlightPeekCode(Highlight &highlight)	Highlights the code for a peek operation.
void highlightPushCode(Highlight &highlight)	Highlights the code for a push operation.
void highlightPopCode(Highlight &highlight)	Highlights the code for a pop operation.
void highlightPeekQueue(Highlight &highlight, int peekPos)	Highlights the code for a peek operation in a queue at the specified position.
void highlightEnqueue(Highlight &highlight)	Highlights the code for an enqueue operation.
void highlightDequeue(Highlight &highlight)	Highlights the code for a dequeue operation.
void highlightAccessArray(Highlight &highlight)	Highlights the code for accessing an array.
void highlightAddArray(Highlight &highlight, bool statusArray)	Highlights the code for adding an element to an array. If `statusArray` is `true` , it indicates a successful addition.
Function	**Usage**
void highlightDeleteArray(Highlight &highlight)	Highlights the code for deleting an element from an array.
void highlightUpdateArray(Highlight &highlight)	Highlights the code for updating an element in an array.
void highlightSearchArray(Highlight &highlight, int correctStatus)	Highlights the code for searching an element in an array. If `correctStatus` is provided, it indicates the search result (found or not found).

Project Setup

- Clone this repository
- Change the directory to src `cd src`
- Run the Data_Visualization.exe `./Data_Visualization.exe`

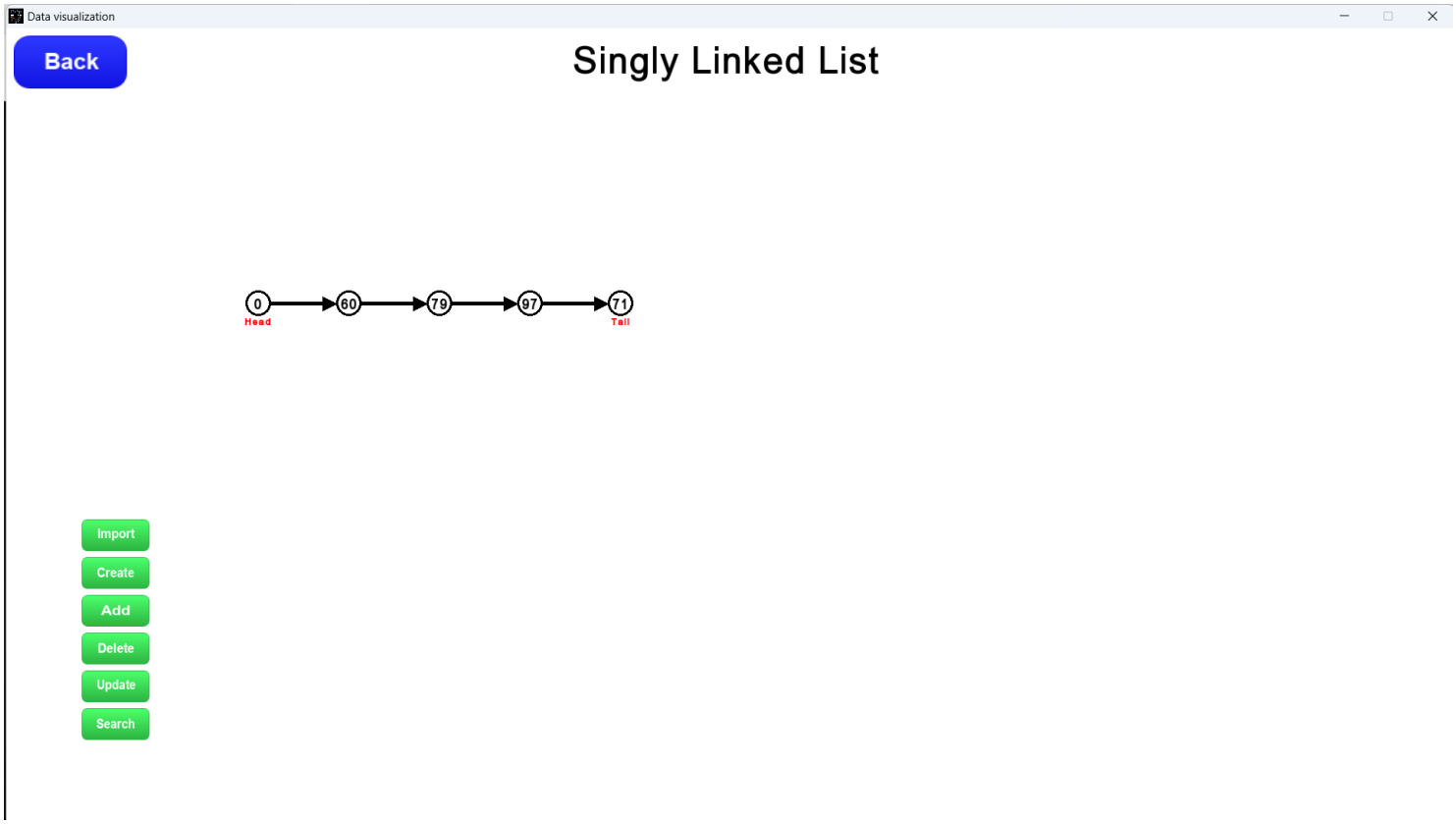
User manual

This is the Home Screen:



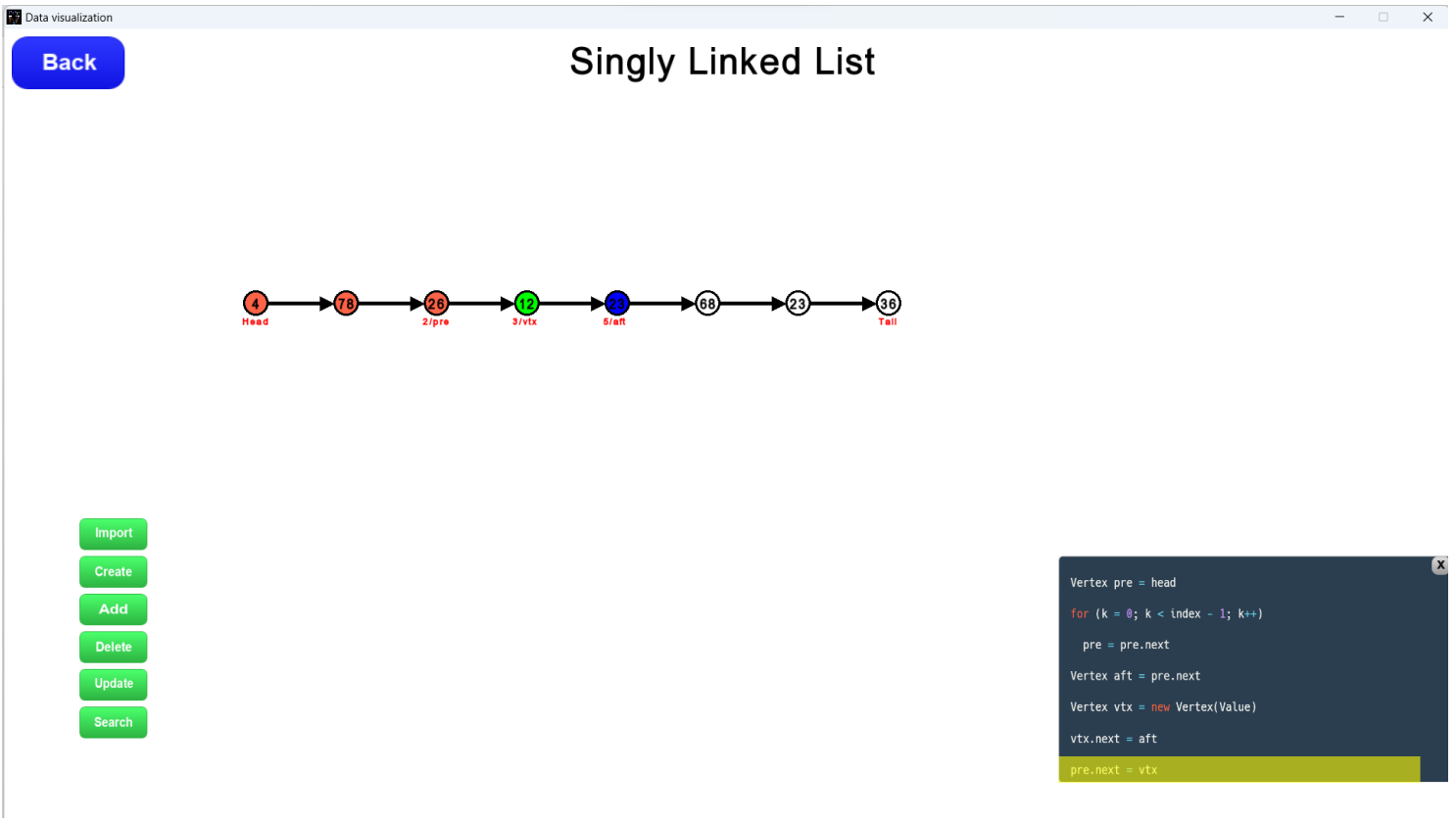
To select a data structure, click on the icon of that data structure.

For example, Singly Linked List is chosen:



You can interact with the data structures using the button.

If a text box appears, you need to input a valid value. And the data structures will start the process.



Github repository

Commit history

Video demo