Visualizer

1.0.0

Generated by Doxygen 1.9.1

Chapter 1

VISUALIZER

- This is my CS163 solo project that is used to visualize data structures.

1.1 Dependency

- C++17
- SDL2 (SDL2_image, SDL2_ttf)
- `https://github.com/nlohmann/json`
- CMake
- Makefile

1.2 Demonstrate

1.3 Features

1.3.1 Main features

- [x] Main menu
- [x] Option menu
- [x] Settings
- [ ] Code highlight
  - [x] Hash table
  - [x] AVL
  - [ ] 234 tree
  - [x] Heap
  - [x] Trie
  - [ ] graph

1.3.2 Data structures

- [x] AVL
  - [x] Initial, init from file
  - [x] Insert a value
  - [x] Delete a value
  - [x] Find a value
- [x] Trie
  - [x] Initial, init from file
  - [x] Insert a value
  - [x] Delete a value
  - [x] Find a value
- [x] Hash table
  - [x] Initial, init from file
  - [x] Insert a value
  - [x] Delete a value
  - [x] Find a value
- [x] Heap
  - [x] Initial, init from file
  - [x] Insert a value
  - [x] Remove the largest/smallest value
  - [x] Get the largest/smallest value
  - [x] Get the size of heap
- [x] Graph
  - [x] Init from matrix, file
  - [x] Connected components
  - [x] Minimum spanning tree
  - [x] Dijkstra
- [ ] 234 tree
  - [ ] Initial, init from file
  - [ ] Insert a value
  - [ ] Delete a value
  - [ ] Find a value

1.4 Documentation

- `Report`
- `Demonstrate video`

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 4

Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

Chapter 6

Namespace Documentation

6.1 DISPLAY Namespace Reference

Name of display.

Variables

- const std::string HOME_ = "home"
- const std::string WORKING_ = "working"

6.1.1 Detailed Description

Name of display.

6.1.2 Variable Documentation

6.1.2.1 HOME_

```
const std::string DISPLAY::HOME_ = "home"
```

Definition at line 152 of file GLOBAL.hpp.

6.1.2.2 WORKING_

```
const std::string DISPLAY::WORKING_ = "working"
```

Definition at line 153 of file GLOBAL.hpp.

6.2 FILEE Namespace Reference

Interact with text files.

Functions

- std::vector< std::string > readFile (std::string path)

6.2.1 Detailed Description

Interact with text files.

6.2.2 Function Documentation

6.2.2.1 readFile()

```
std::vector< std::string > FILEE::readFile (
            std::string path )
```

Definition at line 5 of file file.cpp.

```
{
    std::vector<std::string> result;

    std::ifstream fin(path);

     std::string line;

     while(std::getline(fin, line))
     {
         result.push_back(line);
     }

     fin.close();

     return result;
}
```

6.3 JSON Namespace Reference

Interact with ∗.json files.

Functions

- json ∗ readFile (std::string path)
- void saveFile (std::string path, json ∗data)

6.3.1 Detailed Description

Interact with ∗.json files.

6.3.2 Function Documentation

6.3.2.1 readFile()

```
json * JSON::readFile (
              std::string path )
```

Definition at line 5 of file json.cpp.

```
{
    json* mem = new json();
    std::ifstream fin(path);

     fin » *mem;

     fin.close();

     return mem;
}
```

6.3.2.2 saveFile()

```
void JSON::saveFile (
              std::string path,
              json * data )
```

Definition at line 17 of file json.cpp.

```
{
     std::ofstream fout(path);

     fout « data->dump(4);

     fout.close();

     return ;
}
```

6.4 NUMBER Namespace Reference

Convert between string and interger.

Functions

- int64_t stringToInt (std::string s)
- std::string intToString (int64_t n)
- std::vector< int > stringToArray (std::string s)
- bool isDigit (char c)
- bool isLetter (char c)
- bool isSymbol (char c)
- bool isSign (char c)
- bool isOperator (char c)
- std::string removeLeadingZero (std::string s)
- bool isNumber (std::string s)
- bool isInInterval (std::string s, int64_t a, int64_t b)

Variables

- const int64_t INF = LLONG_MAX

6.4.1 Detailed Description

Convert between string and interger.

6.4.2 Function Documentation

6.4.2.1 intToString()

```
std::string NUMBER::intToString (
            int64_t n )
```

Definition at line 4 of file number.cpp.

```
{
    if(n == 0) return "0";
    std::string result = "";
    bool negative = false;
    if(n < 0)
     {
         negative = true;
         n *= -1;
     }

     while(n)
     {
         result = (char) (n % 10 + '0') + result;
         n /= 10;
     }

     return result;
}
```

6.4.2.2 isDigit()

```
bool NUMBER::isDigit (
            char c )
```

Definition at line 67 of file number.cpp.

```
{
     if((int) c < 48 || (int) c > 57) return false;
     return true;
}
```

6.4.2.3 isInInterval()

```
bool NUMBER::isInInterval (
            std::string s,
            int64_t a,
            int64_t b )
```

Definition at line 106 of file number.cpp.

```
{
    if (!NUMBER::isNumber(s)) return false;

    int64_t n = stringToInt(s);

    return (n >= a && n <= b);
}
```

6.4.2.4 isLetter()

```
bool NUMBER::isLetter (
            char c )
```

Definition at line 73 of file number.cpp.

```
{
    return ((c >= 'a' && c <= 'z') || (c >= 'A' && c <= 'Z'));
}
```

6.4.2.5 isNumber()

```
bool NUMBER::isNumber (
            std::string s )
```

Definition at line 91 of file number.cpp.

```
{
    if ((int) s.size() == 0)
        return false;
    if((int) s.size() == 1)
        return isDigit(s[0]);
    if(!NUMBER::isSign(s[0]) && !NUMBER::isDigit(s[0]))
        return false;

     for (int i = 1; i < (int) s.size(); i++)
         if (!NUMBER::isDigit(s[i])) return false;

     return true;
}
```

6.4.2.6 isOperator()

```
bool NUMBER::isOperator (
            char c )
```

Definition at line 82 of file number.cpp.

```
{
    return (c == '+' || c == '-' || c == '*' || c == '/');
}
```

6.4.2.7 isSign()

```
bool NUMBER::isSign (
            char c )
```

Definition at line 86 of file number.cpp.

```
{
    return (c == '+' || c == '-');
}
```

6.4.2.8 isSymbol()

```
bool NUMBER::isSymbol (
            char c )
```

Definition at line 77 of file number.cpp.

```
{
    return (c == '.' || c == ',' || c == '!' || c == '?' || c == ':' || c == ';' || c == '(' || c == ')'
    || c == '[' || c == ']' || c == '{' || c == '}' || c == '"' || c == '\'');
}
```

6.4.2.9 removeLeadingZero()

```
std::string NUMBER::removeLeadingZero (
            std::string s )
```

6.4.2.10 stringToArray()

```
std::vector< int > NUMBER::stringToArray (
            std::string s )
```

Definition at line 24 of file number.cpp.

```
{
    std::vector<int> result;
    int i = 0;

    while(i != (int) s.size())
    {
        while(i != (int) s.size() && !NUMBER::isDigit(s[i])) i++;
        if(i == (int) s.size()) break;

        int n = 0;
        while(i != (int) s.size() && NUMBER::isDigit(s[i])) n = n * 10 + s[i++] - '0';
        result.push_back(n);
    }
    return result;
}
```

6.4.2.11 stringToInt()

```
int64_t NUMBER::stringToInt (
            std::string s )
```

Definition at line 41 of file number.cpp.

```
{
    if(!NUMBER::isNumber(s)) return NUMBER::INF;
    if((int) s.size() > 18) return NUMBER::INF;
    if((int) s.size() == 1) return (int64_t) (s[0] - '0');

    int64_t n = 0;
    bool negative = false;
    bool sign = false;
    if(NUMBER::isSign(s[0]))
    {
        sign = true;
        negative = (s[0] == '-');
    }

    for(int i = sign; i < (int) s.size(); i++)
    {
        n *= 10;
        n += (int64_t) (s[i] - '0');
    }

    if(negative) n *= -1;

    return n;
}
```

6.4.3 Variable Documentation

6.4.3.1 INF

```
const int64_t NUMBER::INF = LLONG_MAX
```

Definition at line 36 of file services.hpp.

6.5 PATH Namespace Reference

Path to assets, atributes, and saving files.

Namespaces

- ASSETS
- ATB
- SAVING

Variables

- const std::string ASSETS_ = "assets/"
- const std::string ATTRIBUTE_ = "atb/"
- const std::string SAVING_ = "saving/"

6.5.1 Detailed Description

Path to assets, atributes, and saving files.

6.5.2 Variable Documentation

6.5.2.1 ASSETS_

```
const std::string PATH::ASSETS_ = "assets/"
```

Definition at line 161 of file GLOBAL.hpp.

6.5.2.2 ATTRIBUTE_

```
const std::string PATH::ATTRIBUTE_ = "atb/"
```

Definition at line 168 of file GLOBAL.hpp.

6.5.2.3 SAVING_

```
const std::string PATH::SAVING_ = "saving/"
```

Definition at line 180 of file GLOBAL.hpp.

6.6 PATH::ASSETS Namespace Reference

Variables

- const std::string GRAPHICS_ = "assets/graphics/"
- const std::string FONTS_ = "assets/fonts/"
- const std::string SCRIPT_ = "assets/script/"

6.6.1 Variable Documentation

6.6.1.1 FONTS_

```
const std::string PATH::ASSETS::FONTS_ = "assets/fonts/"
```

Definition at line 165 of file GLOBAL.hpp.

6.6.1.2 GRAPHICS_

```
const std::string PATH::ASSETS::GRAPHICS_ = "assets/graphics/"
```

Definition at line 164 of file GLOBAL.hpp.

6.6.1.3 SCRIPT_

```
const std::string PATH::ASSETS::SCRIPT_ = "assets/script/"
```

Definition at line 166 of file GLOBAL.hpp.

6.7 PATH::ATB Namespace Reference

Variables

- const std::string SPRITE_ = "atb/sprite/"
- const std::string OBJECT_ = "atb/object/"
- const std::string DISPLAY_ = "atb/display/"
- const std::string BUTTON_ = "atb/button/"
- const std::string DATA_STRUCTURES_ = "atb/data_structures/"
- const std::string INPUTBOX_ = "atb/input/"
- const std::string SCRIPT_ = "atb/script/"

6.7.1 Variable Documentation

6.7.1.1 BUTTON_

```
const std::string PATH::ATB::BUTTON_ = "atb/button/"
```

Definition at line 174 of file GLOBAL.hpp.

6.7.1.2 DATA_STRUCTURES_

```
const std::string PATH::ATB::DATA_STRUCTURES_ = "atb/data_structures/"
```

Definition at line 175 of file GLOBAL.hpp.

6.7.1.3 DISPLAY_

```
const std::string PATH::ATB::DISPLAY_ = "atb/display/"
```

Definition at line 173 of file GLOBAL.hpp.

6.7.1.4 INPUTBOX_

```
const std::string PATH::ATB::INPUTBOX_ = "atb/input/"
```

Definition at line 176 of file GLOBAL.hpp.

6.7.1.5 OBJECT_

```
const std::string PATH::ATB::OBJECT_ = "atb/object/"
```

Definition at line 172 of file GLOBAL.hpp.

6.7.1.6 SCRIPT_

```
const std::string PATH::ATB::SCRIPT_ = "atb/script/"
```

Definition at line 177 of file GLOBAL.hpp.

6.7.1.7 SPRITE_

```
const std::string PATH::ATB::SPRITE_ = "atb/sprite/"
```

Definition at line 171 of file GLOBAL.hpp.

6.8 PATH::SAVING Namespace Reference

Variables

- const std::string AVL_ = "saving/AVL.txt"
- const std::string HASH_TABLE_ = "saving/HASH_TABLE.txt"
- const std::string GRAPH_ = "saving/GRAPH.txt"
- const std::string TRIE_ = "saving/TRIE.txt"
- const std::string MIN_HEAP_ = "saving/MIN_HEAP.txt"
- const std::string MAX_HEAP_ = "saving/MAX_HEAP.txt"
- const std::string BTREE_4TH_ = "saving/BTREE_4TH.txt"

6.8.1 Variable Documentation

6.8.1.1 AVL_

```
const std::string PATH::SAVING::AVL_ = "saving/AVL.txt"
```

Definition at line 183 of file GLOBAL.hpp.

6.8.1.2 BTREE_4TH_

```
const std::string PATH::SAVING::BTREE_4TH_ = "saving/BTREE_4TH.txt"
```

Definition at line 189 of file GLOBAL.hpp.

6.8.1.3 GRAPH_

```
const std::string PATH::SAVING::GRAPH_ = "saving/GRAPH.txt"
```

Definition at line 185 of file GLOBAL.hpp.

6.8.1.4 HASH_TABLE_

```
const std::string PATH::SAVING::HASH_TABLE_ = "saving/HASH_TABLE.txt"
```

Definition at line 184 of file GLOBAL.hpp.

6.8.1.5 MAX_HEAP_

```
const std::string PATH::SAVING::MAX_HEAP_ = "saving/MAX_HEAP.txt"
```

Definition at line 188 of file GLOBAL.hpp.

6.8.1.6 MIN_HEAP_

```
const std::string PATH::SAVING::MIN_HEAP_ = "saving/MIN_HEAP.txt"
```

Definition at line 187 of file GLOBAL.hpp.

6.8.1.7 TRIE_

```
const std::string PATH::SAVING::TRIE_ = "saving/TRIE.txt"
```

Definition at line 186 of file GLOBAL.hpp.

6.9 RANDOM Namespace Reference

Random intergers, doubles, strings generator.

Functions

- int getInt (int a, int b)
- std::string getInt (int length, int a, int b)
- long long getLongLong (long long a, long long b)
- float getFloat (float a, float b)
- double getDouble (double a, double b)
- char getChar (char a, char b)
- char getChar ()
- std::string getString (int length)
- std::string getString (int length, char a, char b)
- bool flipCoin ()

Variables

- std::mt19937 rng = std::mt19937(std::chrono::steady_clock::now().time_since_epoch().count())

6.9.1 Detailed Description

Random intergers, doubles, strings generator.

6.9.2 Function Documentation

6.9.2.1 flipCoin()

```
bool RANDOM::flipCoin ( )
```

Definition at line 60 of file random.cpp.

```
{
    return getInt(1, 2) - 1;
}
```

6.9.2.2 getChar() [1/2]

```
char RANDOM::getChar ( )
```

Definition at line 39 of file random.cpp.

```
{
    return getChar('a', 'z');
}
```

6.9.2.3 getChar() [2/2]

```
char RANDOM::getChar (
            char a,
            char b )
```

Definition at line 34 of file random.cpp.

```
{
    return std::uniform_int_distribution<char> (a, b)(rng);
}
```

6.9.2.4 getDouble()

```
double RANDOM::getDouble (
            double a,
            double b )
```

Definition at line 29 of file random.cpp.

```
{
    return std::uniform_real_distribution<double> (a, b)(rng);
}
```

6.9.2.5 getFloat()

```
float RANDOM::getFloat (
             float a,
             float b )
```

Definition at line 24 of file random.cpp.

```
{
    return std::uniform_real_distribution<float> (a, b)(rng);
}
```

6.9.2.6 getInt() [1/2]

```
int RANDOM::getInt (
             int a,
             int b )
```

Definition at line 5 of file random.cpp.

```
{
    return std::uniform_int_distribution<int> (a, b)(rng);
}
```

6.9.2.7 getInt() [2/2]

```
std::string RANDOM::getInt (
             int length,
             int a,
             int b )
```

Definition at line 10 of file random.cpp.

```
{
    if(length == 0) return "";
    std::string result = NUMBER::intToString(RANDOM::getInt(a, b));

    while(--length) result = result + " " + NUMBER::intToString(RANDOM::getInt(a, b));
    return result;
}
```

6.9.2.8 getLongLong()

```
long long RANDOM::getLongLong (
             long long a,
             long long b )
```

Definition at line 19 of file random.cpp.

```
{
    return std::uniform_int_distribution<long long> (a, b)(rng);
}
```

6.9.2.9 getString() `[1/2]`

```
std::string RANDOM::getString (
              int length )
```

Definition at line 44 of file random.cpp.

```
{
    std::string s = "";
    for (int i = 0; i < length; i++)
        s += getChar();
    return s;
}
```

6.9.2.10 getString() `[2/2]`

```
std::string RANDOM::getString (
              int length,
              char a,
              char b )
```

Definition at line 52 of file random.cpp.

```
{
    std::string s = "";
    for (int i = 0; i < length; i++)
        s += getChar(a, b);
    return s;
}
```

6.9.3 Variable Documentation

6.9.3.1 rng

```
std::mt19937 RANDOM::rng = std::mt19937(std::chrono::steady_clock::now().time_since_epoch().count())
[extern]
```

Definition at line 3 of file random.cpp.

6.10 SIUSTRING Namespace Reference

Features for std::string.

Functions

- bool isSeparator (char c)
- std::vector< std::string > split (std::string s)

6.10.1 Detailed Description

Features for std::string.

6.10.2 Function Documentation

6.10.2.1 isSeparator()

```
bool SIUSTRING::isSeparator (
             char c )
```

Definition at line 3 of file string.cpp.

```
                                  {
    return (c == ' ' || c == '\t' || c == '\n' || c == '\r' || c == ',');
}
```

6.10.2.2 split()

```
std::vector< std::string > SIUSTRING::split (
             std::string s )
```

Definition at line 7 of file string.cpp.

```
{
    std::vector<std::string> result;

    int i = 0;
    while (i < s.length())
    {
        while(i < s.length() && isSeparator(s[i])) i++;
        if (i >= s.length()) break;
        result.push_back("");
        while(i < s.length() && !isSeparator(s[i])) result.back() += s[i++];
    }

    return result;
}
```

Chapter 7

Class Documentation

7.1 AVL Class Reference

AVL class.

```
#include <AVL.hpp>
```

Public Member Functions

- AVL (SDL_Renderer ∗render, std::mutex &m, TTF_Font ∗f, SDL_Rect v, int cap)
- ∼AVL ()
- void init (std::vector< int > v)
- bool insert (int key)
- bool remove (int key)
- bool search (int key)
- int maxDepth ()
- void setEdgesColor (SDL_Color c)
- void setNodeColor (SDL_Color bg, SDL_Color fg)
- void goOff ()
- void goOn ()
- void goNext ()
- void goBack ()
- void speedUp ()
- void slowDown ()
- void closeScript ()
- bool isReceiveEvent (SDL_Event &e)
- Button ∗ react (SDL_Event &e)
- void rendering ()
- void setting (SDL_Color c1, SDL_Color c2, SDL_Color c3, SDL_Color c4)

Protected Member Functions

- Node ∗ rotateLeft (Node ∗node)
- Node ∗ rotateRight (Node ∗node)
- int getHigh (Node ∗node)
- int balanceFactor (Node ∗node)
- Node ∗ balancing (Node ∗node)
- int maxDepth (Node ∗node)
- Node ∗ insert (Node ∗node, Node ∗newNode)
- Node ∗ unplugSmallest (Node ∗Node, struct Node ∗&n2)
- Node ∗ remove (Node ∗node, int key)
- void search (Node ∗node, int key)
- int locating (Node ∗node, int shiftDown, int shiftRight)
- void renderLine (Node ∗src, Node ∗dst)
- void waitForStep ()
- void highlight (std::vector< int > l)
- void unhighlight (std::vector< int > l)

7.1.1 Detailed Description

AVL class.

Drawable AVL tree.

Definition at line 18 of file AVL.hpp.

7.1.2 Constructor & Destructor Documentation

7.1.2.1 AVL()

```
AVL::AVL (
              SDL_Renderer * render,
              std::mutex & m,
              TTF_Font * f,
              SDL_Rect v,
              int cap )
```

Definition at line 23 of file constructor.cpp.

```
                                                                       : ds_mutex(m)
{
    root = nullptr;
    render = rend;
    font = f;
    viewport = vp;
    capacity = cap;
    sizeOfTree = 0;
    isQueue = false;
    isPause = false;
    edgesColor = {255, 255, 255, 255};
    fontColor = {255, 255, 255, 255};
    nodeColor = {20, 85, 185, 255};
    shiftX = 20;
    shiftY = 20;
    distanceX = 40;
    distanceY = 100;
    isMoving = false;
    stepWait = 600;
    isAnimate = false;

    nodeColor = {20, 85, 185, 255};
    fontColor = {255, 255, 255, 255};
    bgColor = {0, 0, 0, 255};

    std::string fontpath = PATH::ASSETS::FONTS_ + "nimbus-sans-l/regular.otf";
    scriptFont = TTF_OpenFont(fontpath.c_str(), 18);

    currentScript = nullptr;
    Script* insert = new Script(render, scriptFont);
    insert->linking("AVL/insert");
    scripts[DATA_STRUCTURES_OPERATOR::INSERT] = insert;

    Script* remove = new Script(render, scriptFont);
    remove->linking("AVL/remove");
    scripts[DATA_STRUCTURES_OPERATOR::DELETE] = remove;

    Script* search = new Script(render, scriptFont);
    search->linking("AVL/search");
    scripts[DATA_STRUCTURES_OPERATOR::SEARCH] = search;

    Script* init = new Script(render, scriptFont);
    init->linking("AVL/init");
    scripts[DATA_STRUCTURES_OPERATOR::INIT] = init;

    currentScript = insert;
}
```

7.1.2.2 ∼AVL()

`AVL::∼AVL ( )`

Definition at line 9 of file destructor.cpp.

```
{
    if(root != nullptr) delete root;
    if(scriptFont != nullptr) TTF_CloseFont(scriptFont);
    for(auto& script : scripts)
    {
        if(script.second != nullptr) delete script.second;
    }
    scripts.clear();
}
```

7.1.3 Member Function Documentation

7.1.3.1 balanceFactor()

```
int AVL::balanceFactor (
              Node * node )   [protected]
```

Definition at line 9 of file rotate.cpp.

```
{
    if(node == nullptr) return 0;
    return getHigh(node->lson) - getHigh(node->rson);
}
```

7.1.3.2 balancing()

```
AVL::Node * AVL::balancing (
              Node * node )  [protected]
```

Definition at line 79 of file rotate.cpp.

```
{
    if(node == nullptr)
    {
        return nullptr;
    }
    if(node->lson != nullptr) node->lson->repair();
    if(node->rson != nullptr) node->rson->repair();

    node->repair();
    int bf = balanceFactor(node);

    if(bf >= -1 && bf <= 1) return node;
    if(bf > 1)
    {
        if(balanceFactor(node->lson) < 0)
        {
            node->lson = rotateLeft(node->lson);
            node->repair();
        }
        node = rotateRight(node);
    }
    else if(bf < -1)
    {
        if(balanceFactor(node->rson) > 0)
        {
            node->rson = rotateRight(node->rson);
            node->repair();
        }
        node = rotateLeft(node);
        node->repair();
    }
    return node;
}
```

7.1.3.3 closeScript()

```
void AVL::closeScript ( )
```

Definition at line 79 of file event.cpp.

```
{
    currentScript = nullptr;
}
```

7.1.3.4 getHigh()

```
int AVL::getHigh (
              Node * node )  [protected]
```

Definition at line 3 of file rotate.cpp.

```
{
    if(node == nullptr) return 0;
    return node->high;
}
```

7.1.3.5 goBack()

```
void AVL::goBack ( )
```

Definition at line 52 of file step.cpp.

```
{
}
```

7.1.3.6 goNext()

```
void AVL::goNext ( )
```

Definition at line 56 of file step.cpp.

```
{
    pause_mutex.lock();
    isQueue = true;
    pause_mutex.unlock();
    step_cv.notify_one();
}
```

7.1.3.7 goOff()

```
void AVL::goOff ( )
```

Definition at line 72 of file step.cpp.

```
{
    pause_mutex.lock();
    isPause = true;
    pause_mutex.unlock();
}
```

7.1.3.8 goOn()

```
void AVL::goOn ( )
```

Definition at line 64 of file step.cpp.

```
{
    pause_mutex.lock();
    isPause = false;
    pause_mutex.unlock();
    step_cv.notify_one();
}
```

7.1.3.9 highlight()

```
void AVL::highlight (
              std::vector< int > l )  [protected]
```

Definition at line 5 of file step.cpp.

```
{
    if(isAnimate)
    {
        animate_mutex.lock();
        for(int i = 0; i < l.size(); i++)
        {
            currentScript->highlight(l[i]);
        }
        animate_mutex.unlock();
    }
}
```

7.1.3.10 init()

```
void AVL::init (
              std::vector< int > v )
```

Definition at line 4 of file init.cpp.

```
{
    if(root != nullptr)
        delete root;
    root = nullptr;
    cache = nullptr;
    sizeOfTree = 0;
    shiftX = 20;
    shiftY = 20;
    isAnimate = false;
    currentScript = scripts[DATA_STRUCTURES_OPERATOR::INIT];
    for(int i = 0; i < values.size() && sizeOfTree < capacity; i++)
    {
        Sprite* sprite = new Sprite(render);
        sprite->setFont(font);
        sprite->linking("AVL/node");
        sprite->setText(NUMBER::intToString(values[i]));

        root = insert(root, new Node(values[i], sprite));
    }
    maxHigh = maxDepth(root);
}
```

7.1.3.11 insert() [1/2]

```
bool AVL::insert (
              int key )
```

Definition at line 62 of file insert.cpp.

```
{
    if(sizeOfTree == capacity)
    {
        return false;
    }
    currentScript = scripts[DATA_STRUCTURES_OPERATOR::INSERT];
    cache = nullptr;

    Sprite* sprite = new Sprite(render);
    sprite->setFont(font);
    sprite->linking("AVL/node");

    sprite->setFontColor(fontColor);
    sprite->Object::coloring(nodeColor);

    sprite->setText(NUMBER::intToString(key));

    cache = new Node(key, sprite);

    isPause = false;
    isQueue = false;
    isAnimate = true;

    highlight({0});
    waitForStep();
    unhighlight({0});

    root = insert(root, cache);

    maxHigh = maxDepth(root);


    cache = nullptr;

    return true;
}
```

7.1.3.12 insert() [2/2]

```
AVL::Node * AVL::insert (
            Node * node,
            Node * newNode ) [protected]
```

Definition at line 5 of file insert.cpp.

```
{
    if(node == nullptr)
    {
        node = newNode;
         if(isAnimate)
         {
             highlight({1, 2, 3});
             animate_mutex.lock();
             node->sprite->highlight();
             animate_mutex.unlock();

             waitForStep();

             animate_mutex.lock();
             node->sprite->unhighlight();
             animate_mutex.unlock();
             unhighlight({1, 2, 3});
         }
         sizeOfTree++;
         return newNode;
     }
     if(isAnimate)
     {
         animate_mutex.lock();
         node->sprite->highlight();
         animate_mutex.unlock();

         waitForStep();

         animate_mutex.lock();
         node->sprite->unhighlight();
         animate_mutex.unlock();
     }
     if(compare(newNode, node) == -1)
     {
         highlight({4, 5});
         if(isAnimate) waitForStep();
         unhighlight({4, 5});

         node->lson = insert(node->lson, newNode);
     }
     else if(compare(newNode, node) == 1)
     {
         highlight({6, 7});
         if(isAnimate) waitForStep();
         unhighlight({6, 7});
         node->rson = insert(node->rson, newNode);
     }

    highlight({9});
    node = balancing(node);
    unhighlight({9});

    return node;
}
```

7.1.3.13 isReceiveEvent()

```
bool AVL::isReceiveEvent (
            SDL_Event & e )
```

Definition at line 3 of file event.cpp.

```
{
    std::lock_guard<std::mutex> lk(animate_mutex);
    switch(e.type)
    {
        case SDL_MOUSEBUTTONDOWN:
            if(currentScript != nullptr && currentScript->isReceiveEvent(e)) return true;
             if(e.motion.x < viewport.x || viewport.x + viewport.w < e.motion.x) return false;
             if(e.motion.y < viewport.y || viewport.y + viewport.h < e.motion.y) return false;
             if(e.button.button == SDL_BUTTON_LEFT) return false;
             if(root == nullptr) return false;
             return true;
             break;
         case SDL_MOUSEMOTION:
             if(isMoving) return true;
             if(currentScript == nullptr) return false;
             if(currentScript->isReceiveEvent(e)) return true;
             return false;
             break;
         default:
             return false;
             break;
     }
}
```

7.1.3.14 locating()

```
int AVL::locating (
            Node * node,
            int shiftDown,
            int shiftRight )   [protected]
```

Definition at line 83 of file constructor.cpp.

```
{
    if(node == nullptr)
    {
        int shift = maxHigh - shiftDown;
        return (1 « shift) - 1;
    }
    int left = locating(node->lson, shiftDown + 1, shiftRight);

    if(node->sprite != nullptr)
    {
        node->sprite->locatingX(shiftX + shiftRight * distanceX + left * distanceX);
        node->sprite->locatingY(shiftY + shiftDown * distanceY);
        node->sprite->aligning(HORIZONTAL_ALIGN::CENTER, VERTICAL_ALIGN::CENTER);
    }
    locating(node->rson, shiftDown + 1, shiftRight + left + 1);

     return (1 « (maxHigh - shiftDown)) - 1;
}
```

7.1.3.15 maxDepth() [1/2]

```
int AVL::maxDepth ( )
```

Definition at line 70 of file constructor.cpp.

```
{
    return maxDepth(root);
}
```

7.1.3.16 maxDepth() [2/2]

```
int AVL::maxDepth (
            Node * node )  [protected]
```

Definition at line 75 of file constructor.cpp.

```
{
    if(node == nullptr) return 0;

    return 1 + std::max(maxDepth(node->lson), maxDepth(node->rson));
}
```

7.1.3.17 react()

```
Button * AVL::react (
            SDL_Event & e )
```

Definition at line 28 of file event.cpp.

```
{
    std::lock_guard<std::mutex> lk(animate_mutex);
    switch(e.type)
    {
        case SDL_MOUSEBUTTONDOWN:
            if(currentScript != nullptr && currentScript->isReceiveEvent(e))
            {
                return currentScript->react(e);
            }
            if(isMoving)
            {
                isMoving = false;
                int dx = e.motion.x - lastMousePressed.x;
                int dy = e.motion.y - lastMousePressed.y;
                shiftX += dx;
                shiftY += dy;
            }else
            {
                isMoving = true;
                lastMousePressed.x = e.motion.x;
                lastMousePressed.y = e.motion.y;
            }
            return nullptr;
            break;
        case SDL_MOUSEMOTION:
        {
            if(currentScript != nullptr && currentScript->isReceiveEvent(e))
                return currentScript->react(e);
            if(!isMoving) return nullptr;
            int dx = e.motion.x - lastMousePressed.x;
            int dy = e.motion.y - lastMousePressed.y;
            lastMousePressed.x = e.motion.x;
            lastMousePressed.y = e.motion.y;
            shiftX += dx;
            shiftY += dy;
            if(cache != nullptr)
            {
                cache->sprite->moveX(dx);
                cache->sprite->moveY(dy);
            }
            return nullptr;
            break;
        }
        defaut:
            return nullptr;
            break;
    }
    return nullptr;
}
```

7.1.3.18 remove() [1/2]

```
bool AVL::remove (
              int key )
```

Definition at line 136 of file delete.cpp.

```
{
    if(sizeOfTree == 0)
    {
        return false;
    }

    currentScript = scripts[DATA_STRUCTURES_OPERATOR::DELETE];

    isAnimate = true;

    highlight({0});
    waitForStep();
    unhighlight({0});

    root = remove(root, key);
    locating(root, 0, 0);

    cache = nullptr;
    return true;
}
```

7.1.3.19 remove() [2/2]

```
AVL::Node * AVL::remove (
              Node * node,
              int key )  [protected]
```

Definition at line 15 of file delete.cpp.

```
{
    if(node == nullptr)
    {
        highlight({1, 2, 3});
        waitForStep();
        unhighlight({1, 2, 3});
        return node;
    }
    if(isAnimate)
    {
        animate_mutex.lock();
        node->sprite->highlight();
        animate_mutex.unlock();

        waitForStep();

        animate_mutex.lock();
        node->sprite->unhighlight();
        animate_mutex.unlock();
    }
    if(node->key < key)
    {
        highlight({7, 8});
        waitForStep();
        unhighlight({7, 8});

        node->rson = remove(node->rson, key);

        highlight({13});
        waitForStep();

        if(isAnimate)
        {
            animate_mutex.lock();
            node->sprite->highlight();
            animate_mutex.unlock();

            waitForStep();

            animate_mutex.lock();
            node->sprite->unhighlight();
            animate_mutex.unlock();
        }
        node = balancing(node);
        unhighlight({13});

        return node;
    }
    else if(node->key > key)
    {
        highlight({9, 10, 11});
        waitForStep();
        unhighlight({9, 10, 11});
        node->lson = remove(node->lson, key);

        highlight({13});
        waitForStep();

        if(isAnimate)
        {
            animate_mutex.lock();
            node->sprite->highlight();
            animate_mutex.unlock();

            waitForStep();

            animate_mutex.lock();
            node->sprite->unhighlight();
            animate_mutex.unlock();
        }

        node = balancing(node);
        unhighlight({13});


        return node;
    }
    else
    {
        highlight({5, 6});
        waitForStep();
        unhighlight({5, 6});

        if(node->rson == nullptr)
         {
             Node* temp = node->lson;
             node->lson = nullptr;
             node->rson = nullptr;


             highlight({13});
             waitForStep();
             temp = balancing(temp);
             unhighlight({13});

             delete node;
             return temp;
         }else
         {
             Node* temp = unplugSmallest(node->rson, cache);

             cache->lson = node->lson;

             cache->rson = temp;


             node->lson = nullptr;
             node->rson = nullptr;

             delete node;
             node = cache;
             highlight({13});
             waitForStep();
             node = balancing(node);
             unhighlight({13});
             return node;
         }
     }
 }
```

7.1.3.20 rendering()

void AVL::rendering ( )

Definition at line 17 of file rendering.cpp.

```
{
    if(root == nullptr) return ;
    SDL_RenderSetViewport(render, &viewport);

    std::lock_guard< std::mutex > lock(animate_mutex);

    locating(root, 0, 0);

    std::queue< Node* > q;
    q.push(root);

    while(!q.empty())
    {
        Node* u = q.front();
        q.pop();
        if(u->lson != nullptr)
        {
            renderLine(u, u->lson);
            q.push(u->lson);
        }
        if(u->rson != nullptr)
        {
            renderLine(u, u->rson);
            q.push(u->rson);
        }
        u->sprite->rendering();
    }
    if(currentScript != nullptr)
    {
        SDL_RenderSetViewport(render, nullptr);
        currentScript->rendering();
    }
    if(cache != nullptr)
    {
        //cache->sprite->rendering();
    }
}
```

7.1.3.21 renderLine()

```
void AVL::renderLine (
             Node * src,
             Node * dst ) [protected]
```

Definition at line 4 of file rendering.cpp.

```
{
    SDL_Point psrc = {src->sprite->getX() + src->sprite->getW() / 2, src->sprite->getY() +
     src->sprite->getH() / 2};
    SDL_Point pdst = {dst->sprite->getX() + dst->sprite->getW() / 2, dst->sprite->getY() +
     dst->sprite->getH() / 2};

    SDL_SetRenderDrawColor(render, edgesColor.r, edgesColor.g, edgesColor.b, edgesColor.a);
    for(int i = -1; i <= 1; i++)
    {
        for(int j = -1; j <= 1; j++)
            SDL_RenderDrawLine(render, psrc.x + i, psrc.y + j, pdst.x + i, pdst.y + j);
    }
}
```

7.1.3.22 rotateLeft()

```
AVL::Node * AVL::rotateLeft (
                Node * node )  [protected]
```

Definition at line 15 of file rotate.cpp.

```
{
    if(node == nullptr) return nullptr;
    if(node->rson == nullptr) return node;

    if(isAnimate)
    {
        waitForStep();
        node->sprite->highlight();
        node->rson->sprite->highlight();
        if(node->rson->rson != nullptr)
            node->rson->rson->sprite->highlight();
        waitForStep();

        node->sprite->unhighlight();
        node->rson->sprite->unhighlight();
        if(node->rson->rson != nullptr)
            node->rson->rson->sprite->unhighlight();
        waitForStep();
    }

    Node* tmp = node->rson;
    node->rson = tmp->lson;
    tmp->lson = node;
    node->repair();
    tmp->repair();

    return tmp;
}
```

7.1.3.23 rotateRight()

```
AVL::Node * AVL::rotateRight (
                Node * node )  [protected]
```

Definition at line 45 of file rotate.cpp.

```
{
    if(node == nullptr) return nullptr;
    if(node->lson == nullptr) return node;
    if(isAnimate)
    {
        waitForStep();
        node->sprite->highlight();
        node->lson->sprite->highlight();
        if(node->lson->lson != nullptr)
            node->lson->lson->sprite->highlight();
        waitForStep();
        node->sprite->unhighlight();
        node->lson->sprite->unhighlight();
        if(node->lson->lson != nullptr)
            node->lson->lson->sprite->unhighlight();
        waitForStep();
    }
    Node* tmp = node->lson;
    node->lson = tmp->rson;
    int maxDepth;
    tmp->rson = node;
    node->repair();
    tmp->repair();
    return tmp;
}
```

7.1.3.24 search() [1/2]

```
bool AVL::search (
            int key )
```

Definition at line 51 of file search.cpp.

```
{
    if(sizeOfTree == 0)
    {
        return false;
    }
    currentScript = scripts[DATA_STRUCTURES_OPERATOR::SEARCH];
    cache = nullptr;
    Sprite* sprite = new Sprite(render);
    sprite->setFont(font);
    sprite->linking("AVL/node");
    sprite->setText(NUMBER::intToString(key));

    cache = new Node(key, sprite);

    isAnimate = true;

    highlight({0});
    waitForStep();
    unhighlight({0});


    search(root, key);

    delete cache;
    cache = nullptr;
    return true;
}
```

7.1.3.25 search() [2/2]

```
void AVL::search (
            Node * node,
            int key )  [protected]
```

Definition at line 5 of file search.cpp.

```
{

   if(node == nullptr)
   {
        highlight({2, 3, 4});
        waitForStep();
        unhighlight({2, 3, 4});
        return ;
    }

    if(isAnimate)
    {
        animate_mutex.lock();
        node->sprite->highlight();
        animate_mutex.unlock();
        waitForStep();

        animate_mutex.lock();
        node->sprite->unhighlight();
        animate_mutex.unlock();
    }

    if(node->key == key)
    {
        highlight({6, 7, 8});
        waitForStep();
        unhighlight({6, 7, 8});
        return ;
    }
    if(node->key < key)
    {
        highlight({10, 11});
        waitForStep();
        unhighlight({10, 11});
        search(node->rson, key);
    }
    else
    {
        highlight({12, 13, 14});
        waitForStep();
        unhighlight({12, 13, 14});
        search(node->lson, key);
    }
}
```

7.1.3.26 setEdgesColor()

```
void AVL::setEdgesColor (
              SDL_Color c )
```

7.1.3.27 setNodeColor()

```
void AVL::setNodeColor (
              SDL_Color bg,
              SDL_Color fg )
```

7.1.3.28 setting()

```
void AVL::setting (
              SDL_Color c1,
              SDL_Color c2,
              SDL_Color c3,
              SDL_Color c4 )
```

Definition at line 103 of file constructor.cpp.

```
{
    bgColor = c1;
    nodeColor = c2;
    fontColor = c3;
    edgesColor = c4;

    std::queue<Node*> q;
    q.push(root);

    while(!q.empty())
    {
        Node* node = q.front();
        q.pop();
        if(node == nullptr)
            continue;
        node->sprite->coloring(nodeColor);
        node->sprite->setFontColor(fontColor);
        node->sprite->coloring(nodeColor);
        q.push(node->lson);
        q.push(node->rson);
    }
}
```

7.1.3.29 slowDown()

```
void AVL::slowDown ( )
```

Definition at line 84 of file step.cpp.

```
{
    if(stepWait <= 2000) stepWait = stepWait * 2;
}
```

7.1.3.30 speedUp()

```
void AVL::speedUp ( )
```

Definition at line 79 of file step.cpp.

```
{
    if(stepWait >= 100) stepWait = stepWait / 2;
}
```

7.1.3.31 unhighlight()

```
void AVL::unhighlight (
              std::vector< int > l )  [protected]
```

Definition at line 18 of file step.cpp.

```
{
    if(isAnimate)
    {
        animate_mutex.lock();
        for(int i = 0; i < l.size(); i++)
        {
            currentScript->unhighlight(l[i]);
        }
        animate_mutex.unlock();
    }
}
```

7.1.3.32 unplugSmallest()

```
AVL::Node * AVL::unplugSmallest (
              Node * Node,
              struct Node *& n2 )  [protected]
```

Definition at line 4 of file delete.cpp.

```
{
    if(node == nullptr) return nullptr;
    if(node->lson == nullptr)
    {
        n2 = node;
         return node->rson;
     }
     node->lson = unplugSmallest(node->lson, n2);
     return node;
}
```

7.1.3.33 waitForStep()

```
void AVL::waitForStep ( ) [protected]
```

Definition at line 31 of file step.cpp.

```
{
    if(isAnimate)
    {
        ds_mutex.unlock();
        std::this_thread::sleep_for(std::chrono::milliseconds(stepWait));
        ds_mutex.lock();
    }
    std::lock_guard<std::mutex> pause_lock(pause_mutex);
    if(isPause == false)
    {
        return ;
    }

    ds_mutex.unlock();
    std::unique_lock<std::mutex> lk(step_mutex);
    step_cv.wait(lk, [&]{return isQueue == true;});
    isQueue = false;
    ds_mutex.lock();
}
```

The documentation for this class was generated from the following files:

- include/data_structures/AVL.hpp
- src/AVL/constructor.cpp
- src/AVL/destructor.cpp
- src/AVL/event.cpp
- src/AVL/operator/delete.cpp
- src/AVL/operator/init.cpp
- src/AVL/operator/insert.cpp
- src/AVL/operator/search.cpp
- src/AVL/rendering.cpp
- src/AVL/rotate.cpp
- src/AVL/step.cpp

7.2 BTree4th Class Reference

```
#include <btree4th.hpp>
```

Public Member Functions

- BTree4th ()
- ∼BTree4th ()
- void init (std::vector< int > keys)
- void insert (int key)
- void remove (int key)
- bool search (int key)

Protected Member Functions

- Node ∗ leftMost (Node ∗node)
- Node ∗ rightMost (Node ∗node)
- Node ∗ split (Node ∗node)
- Node ∗ addRecordToLeaf (Node ∗node, int key)
- Node ∗ mergeChild (Node ∗node, int index)
- Node ∗ mergeChild (Node ∗node, Node ∗child)
- void insert (Node ∗&node, int key)
- bool search (Node ∗node, int key)

7.2.1 Detailed Description

Definition at line 8 of file btree4th.hpp.

7.2.2 Constructor & Destructor Documentation

7.2.2.1 BTree4th()

```
BTree4th::BTree4th ( )
```

Definition at line 3 of file constructor.cpp.

```
{
    root = nullptr;
}
```

7.2.2.2 ∼BTree4th()

```
BTree4th::∼BTree4th ( )
```

Definition at line 3 of file destructor.cpp.

```
{
    if(root != nullptr) delete root;
}
```

7.2.3 Member Function Documentation

7.2.3.1 addRecordToLeaf()

```
BTree4th::Node * BTree4th::addRecordToLeaf (
              Node * node,
              int key )  [protected]
```

Definition at line 3 of file addRecord.cpp.

```
{
    if(node == nullptr) return nullptr;
    if(!node->isLeaf()) return node;

    int index = 0;
    for(int i = 0; i < 3; i++)
     {
         if(node->key[i] == nullptr)
         {
             node->key[i] = new int(k);
             index = i;
             return node;
         }
     }
     if(index == 1)
     {
         if(k < *(node->key[0]))
             std::swap(node->key[0], node->key[1]);
     }

     if(index == 2)
     {
         if(*(node->key[0]) < *(node->key[1]))
             std::swap(node->key[0], node->key[1]);
         if(*(node->key[1]) < *(node->key[2]))
             std::swap(node->key[1], node->key[2]);
         if(*(node->key[0]) < *(node->key[2]))
             std::swap(node->key[0], node->key[2]);
     }

     return node;
 }
```

7.2.3.2 init()

```
void BTree4th::init (
        std::vector< int > keys )
```

Definition at line 4 of file init.cpp.

```
{
    for(int i = 0; i < keys.size(); i++)
    {
        root = insert(root, keys[i]);
    }
}
```

7.2.3.3 insert() [1/2]

```
void BTree4th::insert (
        int key )
```

Definition at line 3 of file insert.cpp.

```
{
    if(root == nullptr)
    {
        root = new Node(nullptr, key);
        return ;
    }

     if(root->isLeaf() && !root->isFull())
     {
         root = addRecordToLeaf(root, key);
         return ;
     }

     if(root->isFull()) root = split(root);

     Node* current = root;

     do
     {
         int pnext = 3;
         for(int i = 0; i < 3; i++)
         {
             if(current->key[i] == nullptr)
             {
                 pnext = i;
                 break;
             }
             if(key < *(current->key[i]))
             {
                 pnext = i;
                 break;
             }
         }
         Node *& nxt = current->child[pnext];
         if(nxt->isLeaf() && !nxt->isFull())
         {
             nxt = addRecordToLeaf(nxt, key);
             break;
         }
         if(nxt->isFull())
         {
             nxt = split(nxt);
             mergeChild(current, pnext);
         }

     }while(true);
 }
```

7.2.3.4 insert() [2/2]

```
void BTree4th::insert (
            Node *& node,
            int key )  [protected]
```

7.2.3.5 leftMost()

```
Node* BTree4th::leftMost (
            Node * node )  [protected]
```

7.2.3.6 mergeChild() [1/2]

```
BTree4th::Node * BTree4th::mergeChild (
            Node * node,
            int index )  [protected]
```

Definition at line 57 of file addRecord.cpp.

```
{
    if(node == nullptr) return nullptr;
    if(node->child[index] == nullptr) return node;
    if(!node->child[index]->isBinary()) return node;

    for(int i = 1; i >= 0; i--)
    {
        node->key[i+1] = node->key[i];
    }
    for(int i = 2; i >= index; i--)
    {
        node->child[i+1] = node->child[i];
    }
    Node* tmp = node->child[index];

    node->key[index] = tmp->key[0];
    tmp->key[0] = nullptr;

    node->child[index] = tmp->child[0];
    tmp->child[0] = nullptr;

    node->child[index + 1] = tmp->child[1];
    tmp->child[1] = nullptr;

    delete tmp;

    node->repair(false);
    return node;
}
```

7.2.3.7 mergeChild() [2/2]

```
BTree4th::Node ∗ BTree4th::mergeChild (
              Node ∗ node,
              Node ∗ child )   [protected]
```

Definition at line 37 of file addRecord.cpp.

```
{
    if(node == nullptr) return nullptr;
    if(child == nullptr) return node;
    if(!child->isBinary()) return node;

    int index = -1;
    for(int i = 0; i < 3; i++)
    {
        if(node->child[i] == child)
        {
            index = i;
            break;
        }
    }
    if(index == -1) return node;

    return mergeChild(node, index);
}
```

7.2.3.8 remove()

```
void BTree4th::remove (
              int key )
```

7.2.3.9 rightMost()

```
Node∗ BTree4th::rightMost (
              Node ∗ node )   [protected]
```

7.2.3.10 search() [1/2]

```
bool BTree4th::search (
              int key )
```

Definition at line 23 of file search.cpp.

```
{
    return search(root, key);
}
```

7.2.3.11 search() [2/2]

```
bool BTree4th::search (
            Node * node,
            int key ) [protected]
```

Definition at line 3 of file search.cpp.

```
{
    if(node == nullptr) return false;

    if(node->key[0] == nullptr) return false;

    if(node->containsKey(key)) return true;

     for(int i = 0; i < 3; i++)
     {
         if(node->key[i] == nullptr) break;

         if(key < *(node->key[i]))
         {
             return search(node->child[i], key);
         }
     }
     return search(node->child[3], key);
}
```

7.2.3.12 split()

```
BTree4th::Node * BTree4th::split (
            Node * node ) [protected]
```

Definition at line 3 of file split.cpp.

```
{
    if(node == nullptr) return nullptr;
    if(!node->isFull()) return node;

    Node* lson = new Node(node, *(node->key[0]), node->child[0], node->child[1]);
    Node* rson = new Node(node, *(node->key[2]), node->child[2], node->child[3]);

     delete node->key[0];
     node->key[0] = node->key[1];

     node->key[1] = nullptr;

     delete node->key[2];
     node->key[2] = nullptr;

     node->child[0] = lson;
     node->child[1] = rson;
     node->child[2] = nullptr;
     node->child[3] = nullptr;

     return node;
}
```

The documentation for this class was generated from the following files:

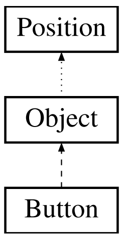
- include/data_structures/btree4th.hpp
- src/btree4th/constructor.cpp
- src/btree4th/destructor.cpp
- src/btree4th/operator/addRecord.cpp
- src/btree4th/operator/init.cpp
- src/btree4th/operator/insert.cpp
- src/btree4th/operator/search.cpp
- src/btree4th/operator/split.cpp

7.3 Button Class Reference

Button class that interact with user input.

```
#include <button.hpp>
```

Inheritance diagram for Button:



Public Member Functions

- Button (SDL_Renderer ∗render)
- ∼Button ()
- bool isHover (int x, int y)
- bool isClicked (int x, int y)
- BUTTON_ACTION getAction ()
- std::string getNextScreen ()
- DATA_STRUCTURES_TYPE getDataType ()
- INPUT_TYPE getInputType ()
- bool isReceiveEvent (SDL_Event &e)
- void rendering ()
- void linking (std::string n)
- void move (int dx, int dy)

Protected Member Functions

- void initSprites (const json &mem)
- void initBackground (const json &mem)
- void initAction (const json &mem)
- void importFromJson ()
- bool isChoosed (int x, int y)
- void fillWithColor ()
- void fillCircleByColor ()
- void fillRectangleByColor ()
- void textToTexture ()
- const SDL_Rect ∗ getCrop ()
- void cropping (int x, int y, int w, int h)
- void cropping (SDL_Rect c)
- void cropping (const json &mem)
- void noCropping ()
- const SDL_Rect ∗ getLocation ()
- int getX ()
- int getY ()
- int getW ()
- int getH ()

- virtual void locating (int x, int y, int w, int h)
- virtual void locating (SDL_Rect l)
- virtual void locating (const json &mem)
- virtual void locatingX (int x)
- virtual void locatingY (int y)
- virtual void locatingW (int w)
- virtual void locatingH (int h)
- virtual void moveX (int delta)
- virtual void moveY (int delta)
- virtual void zoomW (int delta)
- virtual void zoomH (int delta)
- virtual void zoom (double delta)
- virtual void zoomInMiddle (double delta)
- void fitTheTexture ()
- const SDL_Color ∗ getColor ()
- void coloring (int r, int g, int b, int a)
- void coloring (SDL_Color c)
- void coloring (const json &mem)
- void textureFromFile (std::string dir)
- void changeToCircle ()
- void changeToCircle (SDL_Point c)
- void changeToCircle (int x, int y)
- void changeToCircle (SDL_Point c, int r)
- void changeToCircle (int x, int y, int r)
- void changeToRectangle ()
- void setShape (const json &mem)
- bool isLieInside (int x, int y)
- bool isLieInside (SDL_Point p)
- bool isLieInside (SDL_Rect r)
- bool isLieInside (int x, int y, int w, int h)
- void show ()
- void hide ()
- bool isVisible ()
- void importFromJson (const json &mem)
- void setFont (TTF_Font ∗f)
- void setText (std::string t)
- void addText (std::string t)
- void addCharacter (char c)
- void removeCharacter ()
- void removeCharacter (int n)
- std::string getText ()
- int getSize ()

7.3.1 Detailed Description

Button class that interact with user input.

Drawable

Definition at line 15 of file button.hpp.

7.3.2 Constructor & Destructor Documentation

7.3.2.1 Button()

```
Button::Button (
              SDL_Renderer * render )
```

Definition at line 6 of file constructor.cpp.

```
                                      : Object(render)
{
    this->render = render;
    status = BUTTON_STATUS::RELEASED;
     action = BUTTON_ACTION::NONE;
}
```

7.3.2.2 ∼Button()

```
Button::∼Button ( )
```

Definition at line 3 of file destructor.cpp.

```
                  {
    render = nullptr;

    for(auto&i : sprites) {
        delete i;
    }
    sprites.clear();
}
```

7.3.3 Member Function Documentation

7.3.3.1 addCharacter()

```
void Object::addCharacter (
              char c )   [inherited]
```

Definition at line 22 of file font.cpp.

```
{
     text += c;
     textToTexture();
}
```

7.3.3.2 addText()

```
void Object::addText (
            std::string t ) [inherited]
```

Definition at line 16 of file font.cpp.

```
{
    text += t;
    textToTexture();
}
```

7.3.3.3 changeToCircle() [1/5]

```
void Object::changeToCircle ( ) [inherited]
```

Definition at line 5 of file shape.cpp.

```
{
    shapeType = SHAPE::CIRCLE;
    radius = std::min(getW(), getH()) / 2;

    center.x = getX() + getW() / 2;
    center.y = getY() + getH() / 2;
    fillCircleByColor();
}
```

7.3.3.4 changeToCircle() [2/5]

```
void Object::changeToCircle (
            int x,
            int y ) [inherited]
```

Definition at line 24 of file shape.cpp.

```
{
    changeToCircle({x, y});
}
```

7.3.3.5 changeToCircle() [3/5]

```
void Object::changeToCircle (
            int x,
            int y,
            int r ) [inherited]
```

Definition at line 37 of file shape.cpp.

```
{
    shapeType = SHAPE::CIRCLE;
    radius = r;
    center.x = x;
    center.y = y;
    fillCircleByColor();
}
```

7.3.3.6 changeToCircle() [4/5]

```
void Object::changeToCircle (
             SDL_Point c )   [inherited]
```

Definition at line 15 of file shape.cpp.

```
{
    shapeType = SHAPE::CIRCLE;
    center = c;
    radius = std::min(getW() - c.x, c.x - getX());
    radius = std::min(radius, std::min(getH() - c.y, c.y - getY()));
    fillCircleByColor();
}
```

7.3.3.7 changeToCircle() [5/5]

```
void Object::changeToCircle (
             SDL_Point c,
             int r )   [inherited]
```

Definition at line 29 of file shape.cpp.

```
{
    shapeType = SHAPE::CIRCLE;
    radius = r;
    center = c;
    fillCircleByColor();
}
```

7.3.3.8 changeToRectangle()

```
void Object::changeToRectangle ( )   [inherited]
```

Definition at line 46 of file shape.cpp.

```
{
    shapeType = SHAPE::RECTANGLE;
    fillRectangleByColor();
}
```

7.3.3.9 coloring() [1/3]

```
void Object::coloring (
             const json & mem )   [inherited]
```

Definition at line 30 of file coloring.cpp.

```
{
    if(mem.contains("r") && mem.contains("g") && mem.contains("b"))
    {
        if(mem.contains("a")) coloring(mem["r"], mem["g"], mem["b"], mem["a"]);
        else coloring(mem["r"], mem["g"], mem["b"], 255);
    }
}
```

7.3.3.10 coloring() [2/3]

```
void Object::coloring (
              int r,
              int g,
              int b,
              int a )  [inherited]
```

Definition at line 8 of file coloring.cpp.

```
{
    if(color == nullptr) color = new SDL_Color;
    color->r = r;
    color->g = g;
    color->b = b;
    color->a = a;

    fillWithColor();
}
```

7.3.3.11 coloring() [3/3]

```
void Object::coloring (
              SDL_Color c )  [inherited]
```

Definition at line 19 of file coloring.cpp.

```
{
    if(color == nullptr) color = new SDL_Color;
    color->r = c.r;
    color->g = c.g;
    color->b = c.b;
    color->a = c.a;

    fillWithColor();
}
```

7.3.3.12 cropping() [1/3]

```
void Object::cropping (
              const json & mem )  [inherited]
```

Definition at line 26 of file cropping.cpp.

```
{
    if(mem.contains("x") && mem.contains("y") && mem.contains("w") && mem.contains("h"))
        cropping(mem["x"], mem["y"], mem["w"], mem["h"]);
}
```

7.3.3.13 cropping() [2/3]

```
void Object::cropping (
              int x,
              int y,
              int w,
              int h )  [inherited]
```

Definition at line 8 of file cropping.cpp.

```
{
    if(crop == nullptr) crop = new SDL_Rect;
    crop->x = x;
    crop->y = y;
    crop->w = w;
    crop->h = h;
}
```

7.3.3.14 cropping() [3/3]

```
void Object::cropping (
             SDL_Rect c )  [inherited]
```

Definition at line 17 of file cropping.cpp.

```
{
    if(crop == nullptr) crop = new SDL_Rect;
    crop->x = c.x;
    crop->y = c.y;
    crop->w = c.w;
    crop->h = c.h;
}
```

7.3.3.15 fillCircleByColor()

```
void Object::fillCircleByColor ( )  [protected], [inherited]
```

Definition at line 91 of file shape.cpp.

```
{
    if(location == nullptr) locating(0, 0, 0, 0);

    if(texture != nullptr) SDL_DestroyTexture(texture);
    texture = nullptr;

    Uint32 rmask, gmask, bmask, amask;
    Uint32 pixelColor;
#if SDL_BYTEORDER == SDL_BIG_ENDIAN
     rmask = 0xff000000;
     gmask = 0x00ff0000;
     bmask = 0x0000ff00;
     amask = 0x000000ff;
     pixelColor = (color->r « 24) | (color->g « 16) | (color->b « 8) | color->a;
#else
     rmask = 0x000000ff;
     gmask = 0x0000ff00;
     bmask = 0x00ff0000;
     amask = 0xff000000;
     pixelColor = (color->a « 24) | (color->b « 16) | (color->g « 8) | color->r;
#endif

     SDL_Surface *surf = SDL_CreateRGBSurface(0, getW(), getH(), 32, rmask, gmask, bmask, amask);
     SDL_SetSurfaceBlendMode(surf, SDL_BLENDMODE_BLEND);

     texture = SDL_CreateTextureFromSurface(render, surf);
     SDL_FreeSurface(surf);

     Uint32 *pixels = new Uint32[getW() * getH()];
     memset(pixels, 0, getW() * getH() * sizeof(Uint32));

     SDL_Point p = {getW() / 2, getH() / 2};
     center = p;

     if(radius > std::min(getW(), getH()) / 2) radius = std::min(getW(), getH()) / 2;

     for(int i = p.x - radius; i <= p.x + radius; i++)
         for(int j = p.y - radius; j <= p.y + radius; j++)
             if((i - p.x) * (i - p.x) + (j - p.y) * (j - p.y) <= radius * radius)
             {
                 int index = i * getW() + j;
                 if(index < 0 || index >= getW() * getH()) continue;
                 pixels[index] = pixelColor;
             }

     SDL_UpdateTexture(texture, nullptr, pixels, getW() * sizeof(Uint32));
     delete[] pixels;
}
```

7.3.3.16 fillRectangleByColor()

```
void Object::fillRectangleByColor ( )  [protected], [inherited]
```

Definition at line 74 of file shape.cpp.

```
{
    if(location == nullptr) locating(0, 0, 0, 0);

    if(texture != nullptr) SDL_DestroyTexture(texture);
    texture = nullptr;

    SDL_Surface* surf = SDL_CreateRGBSurfaceWithFormat(0, getW(), getH(), 32, SDL_PIXELFORMAT_RGBA32);

    SDL_SetSurfaceBlendMode(surf, SDL_BLENDMODE_BLEND);

    SDL_FillRect(surf, nullptr, SDL_MapRGBA(surf->format, color->r, color->g, color->b, color->a));

    texture = SDL_CreateTextureFromSurface(render, surf);

    SDL_FreeSurface(surf);
}
```

7.3.3.17 fillWithColor()

```
void Object::fillWithColor ( )  [protected], [inherited]
```

Definition at line 39 of file coloring.cpp.

```
{
    if(shapeType == SHAPE::NONE) return fillRectangleByColor();
    if(shapeType == SHAPE::RECTANGLE) return fillRectangleByColor();
    if(shapeType == SHAPE::CIRCLE) return fillCircleByColor();
}
```

7.3.3.18 fitTheTexture()

```
void Object::fitTheTexture ( )  [inherited]
```

Definition at line 140 of file locating.cpp.

```
{
    if(texture == nullptr) return;
    SDL_QueryTexture(texture, nullptr, nullptr, &location->w, &location->h);
}
```

7.3.3.19 getAction()

```
BUTTON_ACTION Button::getAction ( )
```

Definition at line 3 of file action.cpp.

```
{
    return action;
}
```

7.3.3.20 getColor()

```
const SDL_Color * Object::getColor ( )  [inherited]
```

Definition at line 3 of file coloring.cpp.

```
{
    return color;
}
```

7.3.3.21 getCrop()

```
const SDL_Rect * Object::getCrop ( )  [inherited]
```

Definition at line 3 of file cropping.cpp.

```
{
    return crop;
}
```

7.3.3.22 getDataType()

```
DATA_STRUCTURES_TYPE Button::getDataType ( )
```

Definition at line 15 of file action.cpp.

```
{
    if(!args.contains("data-type"))
        return DATA_STRUCTURES_TYPE::NONE;

    std::string type = args["data-type"].get<std::string>();

    if(type == "AVL")
        return DATA_STRUCTURES_TYPE::AVL;

    if(type == "HASH_TABLE")
        return DATA_STRUCTURES_TYPE::HASH_TABLE;

    if(type == "GRAPH")
        return DATA_STRUCTURES_TYPE::GRAPH;

    if(type == "TRIE")
        return DATA_STRUCTURES_TYPE::TRIE;

    if(type == "BTREE_4TH")
        return DATA_STRUCTURES_TYPE::BTREE_4TH;
    if(type == "MIN_HEAP")
        return DATA_STRUCTURES_TYPE::MIN_HEAP;
    if(type == "MAX_HEAP")
        return DATA_STRUCTURES_TYPE::MAX_HEAP;

    return DATA_STRUCTURES_TYPE::NONE;
}
```

7.3.3.23 getH()

```
int Object::getH ( )  [inherited]
```

Definition at line 47 of file locating.cpp.

```
{
    return location->h;
}
```

7.3.3.24 getInputType()

```
INPUT_TYPE Button::getInputType ( )
```

Definition at line 44 of file action.cpp.

```
{
    if(!args.contains("input-type"))
        return INPUT_TYPE::NONE;
    std::string type = args["input-type"].get<std::string>();

    if(type == "INT") return INPUT_TYPE::INT;
    if(type == "ARRAY") return INPUT_TYPE::ARRAY;
    if(type == "STRING") return INPUT_TYPE::STRING;
    if(type == "STRINGS") return INPUT_TYPE::STRINGS;
    return INPUT_TYPE::NONE;
}
```

7.3.3.25 getLocation()

```
const SDL_Rect * Object::getLocation ( )  [inherited]
```

Definition at line 27 of file locating.cpp.

```
{
    return location;
}
```

7.3.3.26 getNextScreen()

```
std::string Button::getNextScreen ( )
```

Definition at line 8 of file action.cpp.

```
{
    if(!args.contains("next-screen"))
        return "";
    return args["next-screen"].get<std::string>();
}
```

7.3.3.27 getSize()

```
int Object::getSize ( )  [inherited]
```

Definition at line 68 of file font.cpp.

```
{
    return text.size();
}
```

7.3.3.28 getText()

```
std::string Object::getText ( )  [inherited]
```

Definition at line 63 of file font.cpp.

```
{
    return text;
}
```

7.3.3.29 getW()

```
int Object::getW ( )  [inherited]
```

Definition at line 42 of file locating.cpp.

```
{
    return location->w;
}
```

7.3.3.30 getX()

```
int Object::getX ( )  [inherited]
```

Definition at line 32 of file locating.cpp.

```
{
    return location->x;
}
```

7.3.3.31 getY()

```
int Object::getY ( )  [inherited]
```

Definition at line 37 of file locating.cpp.

```
{
    return location->y;
}
```

7.3.3.32 hide()

```
void Object::hide ( )  [inherited]
```

Definition at line 8 of file visible.cpp.

```
{
    visible = false;
}
```

7.3.3.33 importFromJson() [1/2]

```
void Button::importFromJson ( )  [protected]
```

Definition at line 13 of file constructor.cpp.

```
{
    json* mem = JSON::readFile(PATH::ATB::BUTTON_ + name + ".json");
    if(mem->contains("background"))
        initBackground((*mem)["background"]);
    if(mem->contains("sprites"))
        initSprites((*mem)["sprites"]);
    if(mem->contains("action"))
        initAction((*mem)["action"]);
}
```

7.3.3.34 importFromJson() [2/2]

```
void Object::importFromJson (
              const json & mem )  [inherited]
```

Definition at line 21 of file constructor.cpp.

```
{
    if(mem.contains("location"))
        locating(mem["location"]);

    if(mem.contains("crop"))
        cropping(mem["crop"]);

    if(mem.contains("color"))
        coloring(mem["color"]);

    if(mem.contains("shape"))
        setShape(mem["shape"]);

    if(mem.contains("visible"))
        visible = mem["visible"];

    if(mem.contains("image"))
        textureFromFile(PATH::ASSETS::GRAPHICS_ + mem["image"].get<std::string>());
    return ;
}
```

7.3.3.35 initAction()

```
void Button::initAction (
              const json & mem )  [protected]
```

Definition at line 24 of file constructor.cpp.

```
{


    if(mem.contains("type"))
    {
        if(mem["type"].get<std::string>() == "CHANGE_SCREEN")
            action = BUTTON_ACTION::CHANGE_SCREEN;
        else if(mem["type"].get<std::string>() == "INSERT")
            action = BUTTON_ACTION::INSERT;
        else if(mem["type"].get<std::string>() == "DELETE")
            action = BUTTON_ACTION::DELETE;
        else if(mem["type"].get<std::string>() == "INIT")
            action = BUTTON_ACTION::INIT;
        else if(mem["type"].get<std::string>() == "SEARCH")
            action = BUTTON_ACTION::SEARCH;
        else if(mem["type"].get<std::string>() == "SETTING")
            action = BUTTON_ACTION::SETTING;
        else if(mem["type"].get<std::string>() == "DONE")
            action = BUTTON_ACTION::DONE;
        else if(mem["type"].get<std::string>() == "EDGES")
            action = BUTTON_ACTION::EDGES;
        else if(mem["type"].get<std::string>() == "RANDOM")
            action = BUTTON_ACTION::RANDOM;
        else if(mem["type"].get<std::string>() == "GO_BACK")
            action = BUTTON_ACTION::GO_BACK;
        else if(mem["type"].get<std::string>() == "GO_NEXT")
            action = BUTTON_ACTION::GO_NEXT;
        else if(mem["type"].get<std::string>() == "GO_ON")
            action = BUTTON_ACTION::GO_ON;
        else if(mem["type"].get<std::string>() == "GO_OFF")
            action = BUTTON_ACTION::GO_OFF;
        else if(mem["type"].get<std::string>() == "SPEED_UP")
            action = BUTTON_ACTION::SPEED_UP;
        else if(mem["type"].get<std::string>() == "SLOW_DOWN")
            action = BUTTON_ACTION::SLOW_DOWN;
        else if(mem["type"].get<std::string>() == "TOP")
            action = BUTTON_ACTION::TOP;
        else if(mem["type"].get<std::string>() == "SIZE")
            action = BUTTON_ACTION::SIZE;
        else if(mem["type"].get<std::string>() == "CONNECTED_COMPONENTS")
            action = BUTTON_ACTION::CONNECTED_COMPONENTS;
        else if(mem["type"].get<std::string>() == "MST")
```

```
            action = BUTTON_ACTION::MST;
        else if(mem["type"].get<std::string>() == "DIJKSTRA")
            action = BUTTON_ACTION::DIJKSTRA;
        else if(mem["type"].get<std::string>() == "RANDOM2")
            action = BUTTON_ACTION::RANDOM2;
        else if(mem["type"].get<std::string>() == "RANDOM3")
            action = BUTTON_ACTION::RANDOM3;
        else if(mem["type"].get<std::string>() == "RANDOM4")
            action = BUTTON_ACTION::RANDOM4;
        else if(mem["type"].get<std::string>() == "RANDOM5")
            action = BUTTON_ACTION::RANDOM5;
        else if(mem["type"].get<std::string>() == "RANDOM6")
            action = BUTTON_ACTION::RANDOM6;
        else if(mem["type"].get<std::string>() == "RANDOM7")
            action = BUTTON_ACTION::RANDOM7;
        else if(mem["type"].get<std::string>() == "RANDOM8")
            action = BUTTON_ACTION::RANDOM8;
        else if(mem["type"].get<std::string>() == "RANDOM9")
            action = BUTTON_ACTION::RANDOM9;
        else if(mem["type"].get<std::string>() == "RANDOM10")
            action = BUTTON_ACTION::RANDOM10;
        else if(mem["type"].get<std::string>() == "RANDOM11")
            action = BUTTON_ACTION::RANDOM11;
        else if(mem["type"].get<std::string>() == "RANDOM12")
            action = BUTTON_ACTION::RANDOM12;
        else if(mem["type"].get<std::string>() == "RANDOM13")
            action = BUTTON_ACTION::RANDOM13;
        else if(mem["type"].get<std::string>() == "RANDOM14")
            action = BUTTON_ACTION::RANDOM14;
        else if(mem["type"].get<std::string>() == "RANDOM15")
            action = BUTTON_ACTION::RANDOM15;
        else if(mem["type"].get<std::string>() == "RANDOM16")
            action = BUTTON_ACTION::RANDOM16;
        else if(mem["type"].get<std::string>() == "FILE")
            action = BUTTON_ACTION::FILE;
        else if(mem["type"].get<std::string>() == "CLOSE")
            action = BUTTON_ACTION::CLOSE;
        else
            action = BUTTON_ACTION::NONE;
    }
    if(mem.contains("args"))
        args = mem["args"];
}
```

7.3.3.36 initBackground()

```
void Button::initBackground (
                const json & mem )   [protected]
```

Definition at line 111 of file constructor.cpp.

```
{
    Object::importFromJson(mem);
}
```

7.3.3.37 initSprites()

```
void Button::initSprites (
                const json & mem )   [protected]
```

Definition at line 116 of file constructor.cpp.

```
{
    for(auto& sprite : mem)
    {
        sprites.push_back(new Sprite(render));
        sprites.back()->linking(sprite["name"].get<std::string>());
        sprites.back()->moveX(getX());
        sprites.back()->moveY(getY());
    }
}
```

7.3.3.38 isChoosed()

```
bool Button::isChoosed (
              int x,
              int y )  [protected]
```

Definition at line 4 of file mouse_action.cpp.

```
{
    return x >= getX() && x < getX() + getW() && y >= getY() && y < getY() + getH();
}
```

7.3.3.39 isClicked()

```
bool Button::isClicked (
              int x,
              int y )
```

Definition at line 32 of file mouse_action.cpp.

```
{
    if (isChoosed(x, y))
    {
        sprites[0]->hide();
        sprites[1]->show();
        status = BUTTON_STATUS::HOVER;
        return true;
    }
    else
    {
        sprites[0]->show();
        sprites[1]->hide();
        status = BUTTON_STATUS::RELEASED;
        return false;
    }
}
```

7.3.3.40 isHover()

```
bool Button::isHover (
              int x,
              int y )
```

Definition at line 9 of file mouse_action.cpp.

```
{
    if(!isVisible())
    {
        return false;
    }
    if (isChoosed(x, y))
    {
        status = BUTTON_STATUS::HOVER;
        sprites[0]->hide();
        sprites[1]->show();
        return true;
    }
    else
    {
        sprites[0]->show();
        sprites[1]->hide();
        status = BUTTON_STATUS::RELEASED;
        return false;
    }
}
```

7.3.3.41 isLieInside() [1/4]

```
bool Object::isLieInside (
                int x,
                int y )  [inherited]
```

Definition at line 3 of file locating.cpp.

```
{
    if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
    return (x >= location->x && x < location->x + location->w && y >= location->y && y < location->y +
     location->h);
}
```

7.3.3.42 isLieInside() [2/4]

```
bool Object::isLieInside (
                int x,
                int y,
                int w,
                int h )  [inherited]
```

Definition at line 21 of file locating.cpp.

```
{
    if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
    return (x >= location->x && x + w <= location->x + location->w && y >= location->y && y + h <=
    location->y + location->h);
}
```

7.3.3.43 isLieInside() [3/4]

```
bool Object::isLieInside (
                SDL_Point p )  [inherited]
```

Definition at line 9 of file locating.cpp.

```
{
    if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
    return (p.x >= location->x && p.x < location->x + location->w && p.y >= location->y && p.y <
    location->y + location->h);
}
```

7.3.3.44 isLieInside() [4/4]

```
bool Object::isLieInside (
                SDL_Rect r )  [inherited]
```

Definition at line 15 of file locating.cpp.

```
{
    if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
    return (r.x >= location->x && r.x + r.w <= location->x + location->w && r.y >= location->y && r.y +
    r.h <= location->y + location->h);
}
```

7.3.3.45 isReceiveEvent()

```
bool Button::isReceiveEvent (
              SDL_Event & e )
```

Definition at line 3 of file event.cpp.

```
{
    switch(e.type)
    {
        case SDL_MOUSEMOTION:
            return isChoosed(e.motion.x, e.motion.y) || status == BUTTON_STATUS::HOVER;
            break;
         case SDL_MOUSEBUTTONDOWN:
             return isChoosed(e.button.x, e.button.y);
             break;
         default:
             return false;
     }


}
```

7.3.3.46 isVisible()

```
bool Object::isVisible ( )  [inherited]
```

Definition at line 13 of file visible.cpp.

```
{
    return visible;
}
```

7.3.3.47 linking()

```
void Button::linking (
              std::string n )
```

Definition at line 128 of file constructor.cpp.

```
{
    name = n;
    importFromJson();
}
```

7.3.3.48 locating() [1/3]

```
void Object::locating (
              const json & mem )  [virtual], [inherited]
```

Definition at line 70 of file locating.cpp.

```
{
    if(mem.contains("x") && mem.contains("y") && mem.contains("w") && mem.contains("h"))
        locating(mem["x"], mem["y"], mem["w"], mem["h"]);
}
```

7.3.3.49 locating() [2/3]

```
void Object::locating (
             int x,
             int y,
             int w,
             int h )  [virtual], [inherited]
```

Reimplemented in Sprite.

Definition at line 52 of file locating.cpp.

```
{
    if(location == nullptr) location = new SDL_Rect;
    location->x = x;
    location->y = y;
    location->w = w;
    location->h = h;
}
```

7.3.3.50 locating() [3/3]

```
void Object::locating (
             SDL_Rect l )  [virtual], [inherited]
```

Reimplemented in Sprite.

Definition at line 61 of file locating.cpp.

```
{
    if(location == nullptr) location = new SDL_Rect;
    location->x = l.x;
    location->y = l.y;
    location->w = l.w;
    location->h = l.h;
}
```

7.3.3.51 locatingH()

```
void Object::locatingH (
             int h )  [virtual], [inherited]
```

Definition at line 94 of file locating.cpp.

```
{
    if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
    location->h = h;
}
```

7.3.3.52 locatingW()

```
void Object::locatingW (
             int w )  [virtual], [inherited]
```

Definition at line 88 of file locating.cpp.

```
{
    if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
    location->w = w;
}
```

7.3.3.53 locatingX()

```
void Object::locatingX (
             int x )  [virtual], [inherited]
```

Reimplemented in Sprite.

Definition at line 76 of file locating.cpp.

```
{
    if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
    location->x = x;
}
```

7.3.3.54 locatingY()

```
void Object::locatingY (
             int y )  [virtual], [inherited]
```

Reimplemented in Sprite.

Definition at line 82 of file locating.cpp.

```
{
    if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
    location->y = y;
}
```

7.3.3.55 move()

```
void Button::move (
             int dx,
             int dy )
```

Definition at line 134 of file constructor.cpp.

```
{
    Object::moveX(dx);
    Object::moveY(dy);
    for(auto& sprite : sprites)
    {
        sprite->moveX(dx);
        sprite->moveY(dy);
    }
}
```

7.3.3.56 moveX()

```
void Object::moveX (
             int delta )  [virtual], [inherited]
```

Reimplemented in Sprite.

Definition at line 100 of file locating.cpp.

```
{
    if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
    location->x += delta;
}
```

7.3.3.57 moveY()

```
void Object::moveY (
             int delta )  [virtual], [inherited]
```

Reimplemented in Sprite.

Definition at line 106 of file locating.cpp.

```
{
    if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
    location->y += delta;
}
```

7.3.3.58 noCropping()

```
void Object::noCropping ( )  [inherited]
```

Definition at line 32 of file cropping.cpp.

```
{
    if(crop != nullptr) delete crop;
    crop = nullptr;
}
```

7.3.3.59 removeCharacter() [1/2]

```
void Object::removeCharacter ( )  [inherited]
```

Definition at line 28 of file font.cpp.

```
{
    if (text.size() > 0)
        text.pop_back();
    textToTexture();
}
```

7.3.3.60 removeCharacter() [2/2]

```
void Object::removeCharacter (
             int n )  [inherited]
```

Definition at line 35 of file font.cpp.

```
{
    if(n == 0) return ;
    if(text.size() <= n) text.clear();
    else text.erase(text.end() - n, text.end());
    textToTexture();
}
```

7.3.3.61 rendering()

```
void Button::rendering ( )  [virtual]
```

Reimplemented from Object.

Definition at line 4 of file rendering.cpp.

```
{
    Object::rendering();
    if(status == BUTTON_STATUS::HOVER)
    {
        sprites[1]->rendering();

     }else if(status == BUTTON_STATUS::RELEASED)
         sprites[0]->rendering();
}
```

7.3.3.62 setFont()

```
void Object::setFont (
            TTF_Font * f )  [inherited]
```

Definition at line 4 of file font.cpp.

```
{
    font = f;
    textToTexture();
}
```

7.3.3.63 setShape()

```
void Object::setShape (
            const json & mem )  [inherited]
```

Definition at line 52 of file shape.cpp.

```
{
    if(mem["type"].get<std::string>() == "CIRCLE")
    {
        if(mem.contains("center"))
        {
            if(mem.contains("radius"))
                changeToCircle(mem["center"]["x"], mem["center"]["y"], mem["radius"]);
            else changeToCircle(mem["center"]["x"], mem["center"]["y"]);
        }else changeToCircle();

        return ;
    }

    if(mem["type"].get<std::string>() == "NONE" || mem["type"].get<std::string>() == "RECTANGLE")
    {
        changeToRectangle();
        return ;
    }

}
```

7.3.3.64 setText()

```
void Object::setText (
            std::string t )  [inherited]
```

Definition at line 10 of file font.cpp.

```
{
    text = t;
    textToTexture();
}
```

7.3.3.65 show()

```
void Object::show ( )  [inherited]
```

Definition at line 3 of file visible.cpp.

```
{
    visible = true;
}
```

7.3.3.66 textToTexture()

```
void Object::textToTexture ( )  [protected], [inherited]
```

Definition at line 43 of file font.cpp.

```
{
    if(font == nullptr) return ;
    if(color == nullptr) return ;
    if(render == nullptr) return ;
    if(texture != nullptr)
    {
        SDL_DestroyTexture(texture);
    }
    texture = nullptr;

    SDL_Surface* surface = TTF_RenderText_Blended(font, text.c_str(), *color);

    if(surface == nullptr) return ;

    texture = SDL_CreateTextureFromSurface(render, surface);
    SDL_FreeSurface(surface);
    fitTheTexture();
}
```

7.3.3.67 textureFromFile()

```
void Object::textureFromFile (
            std::string dir )  [inherited]
```

Definition at line 4 of file external_storage.cpp.

```
{
    SDL_Surface *surface = IMG_Load(dir.c_str());

    texture = SDL_CreateTextureFromSurface(render, surface);
    SDL_FreeSurface(surface);
}
```

7.3.3.68 zoom()

```
void Object::zoom (
             double delta )  [virtual], [inherited]
```

Definition at line 123 of file locating.cpp.

```
{
    if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
    location->w *= delta;
    location->h *= delta;
}
```

7.3.3.69 zoomH()

```
void Object::zoomH (
             int delta )  [virtual], [inherited]
```

Definition at line 118 of file locating.cpp.

```
{
    if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
    location->h += delta;
}
```

7.3.3.70 zoomInMiddle()

```
void Object::zoomInMiddle (
             double delta )  [virtual], [inherited]
```

Definition at line 130 of file locating.cpp.

```
{
    if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
    SDL_Point center = {location->x + location->w / 2, location->y + location->h / 2};
    location->w *= delta;
    location->h *= delta;
    location->x = center.x - location->w / 2;
    location->y = center.y - location->h / 2;
}
```

7.3.3.71 zoomW()

```
void Object::zoomW (
             int delta )  [virtual], [inherited]
```

Definition at line 112 of file locating.cpp.

```
{
    if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
    location->w += delta;
}
```

The documentation for this class was generated from the following files:

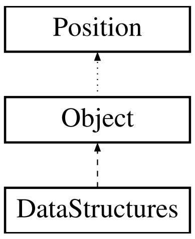
- include/button.hpp
- src/button/action.cpp
- src/button/constructor.cpp
- src/button/destructor.cpp
- src/button/event.cpp
- src/button/mouse_action.cpp
- src/button/rendering.cpp

7.4 DataStructures Class Reference

Container that contains all data structures.

```
#include <data_structures.hpp>
```

Inheritance diagram for DataStructures:



Public Member Functions

- DataStructures (SDL_Renderer ∗r, TTF_Font ∗f, std::mutex &m)
- ∼DataStructures ()
- void linking (std::string n)
- void setDataType (DATA_STRUCTURES_TYPE t)
- DATA_STRUCTURES_TYPE getDataType ()
- std::string getName ()
- void rendering ()
- void init (InputBox ∗inp)
- void insert (InputBox ∗inp)
- void remove (InputBox ∗inp)
- void search (InputBox ∗inp)
- void setting (InputBox ∗inp)
- void top ()
- void size ()
- void scc ()
- void mst ()
- void dijkstra (InputBox ∗inp)
- void goBack ()
- void goNext ()
- void goOn ()
- void goOff ()
- void speedUp ()
- void slowDown ()
- void closeScript ()
- bool isReceiveEvent (SDL_Event &e)
- Button ∗ react (SDL_Event &e)

Public Attributes

- int capacity

Protected Member Functions

- void initBackground (const json &mem)
- void initLinker (const json &mem)
- void initDisplay (const json &mem)
- void importFromJson ()
- void initAVL (InputBox ∗inp)
- void insertAVL (InputBox ∗inp)
- void removeAVL (InputBox ∗inp)
- void searchAVL (InputBox ∗inp)
- void settingAVL (InputBox ∗inp)
- void initTrie (InputBox ∗inp)
- void insertTrie (InputBox ∗inp)
- void removeTrie (InputBox ∗inp)
- void searchTrie (InputBox ∗inp)
- void settingTrie (InputBox ∗inp)
- void initHashTable (InputBox ∗inp)
- void insertHashTable (InputBox ∗inp)
- void removeHashTable (InputBox ∗inp)
- void searchHashTable (InputBox ∗inp)
- void settingHashTable (InputBox ∗inp)
- void initMinHeap (InputBox ∗inp)
- void insertMinHeap (InputBox ∗inp)
- void removeMinHeap (InputBox ∗inp)
- void searchMinHeap (InputBox ∗inp)
- void settingMinHeap (InputBox ∗inp)
- void initGraph (InputBox ∗inp)
- void Dijkstra (InputBox ∗inp)
- void MST ()
- void SCC ()
- void settingGraph (InputBox ∗inp)
- void fillWithColor ()
- void fillCircleByColor ()
- void fillRectangleByColor ()
- void textToTexture ()
- const SDL_Rect ∗ getCrop ()
- void cropping (int x, int y, int w, int h)
- void cropping (SDL_Rect c)
- void cropping (const json &mem)
- void noCropping ()
- const SDL_Rect ∗ getLocation ()
- int getX ()
- int getY ()
- int getW ()
- int getH ()
- virtual void locating (int x, int y, int w, int h)
- virtual void locating (SDL_Rect l)
- virtual void locating (const json &mem)
- virtual void locatingX (int x)
- virtual void locatingY (int y)
- virtual void locatingW (int w)
- virtual void locatingH (int h)
- virtual void moveX (int delta)
- virtual void moveY (int delta)
- virtual void zoomW (int delta)

- virtual void zoomH (int delta)
- virtual void zoom (double delta)
- virtual void zoomInMiddle (double delta)
- void fitTheTexture ()
- const SDL_Color ∗ getColor ()
- void coloring (int r, int g, int b, int a)
- void coloring (SDL_Color c)
- void coloring (const json &mem)
- void textureFromFile (std::string dir)
- void changeToCircle ()
- void changeToCircle (SDL_Point c)
- void changeToCircle (int x, int y)
- void changeToCircle (SDL_Point c, int r)
- void changeToCircle (int x, int y, int r)
- void changeToRectangle ()
- void setShape (const json &mem)
- bool isLieInside (int x, int y)
- bool isLieInside (SDL_Point p)
- bool isLieInside (SDL_Rect r)
- bool isLieInside (int x, int y, int w, int h)
- void show ()
- void hide ()
- bool isVisible ()
- void importFromJson (const json &mem)
- void setFont (TTF_Font ∗f)
- void setText (std::string t)
- void addText (std::string t)
- void addCharacter (char c)
- void removeCharacter ()
- void removeCharacter (int n)
- std::string getText ()
- int getSize ()

7.4.1 Detailed Description

Container that contains all data structures.

Drawable

contain AVL

contain Trie

contain Hash Table

contain Graph

contain Heap

Definition at line 40 of file data_structures.hpp.

7.4.2 Constructor & Destructor Documentation

7.4.2.1 DataStructures()

```
DataStructures::DataStructures (
              SDL_Renderer * r,
              TTF_Font * f,
              std::mutex & m )
```

Definition at line 3 of file constructor.cpp.

```
: Object(r), ds_mutex(m)
{
    font = f;
    render = r;
    avl = nullptr;
    trie = nullptr;
    hashTable = nullptr;
    minheap = nullptr;
    graph = nullptr;
}
```

7.4.2.2 ∼DataStructures()

```
DataStructures::∼DataStructures ( )
```

Definition at line 3 of file destructor.cpp.

```
{
    render = nullptr;
    for(auto& i : node)
        delete i;
    for(auto& i : displays)
        delete i;
    node.clear();
    if(avl != nullptr) delete avl;
    if(trie != nullptr) delete trie;
    if(hashTable != nullptr) delete hashTable;
    if(minheap != nullptr) delete minheap;
    if(graph != nullptr) delete graph;
}
```

7.4.3 Member Function Documentation

7.4.3.1 addCharacter()

```
void Object::addCharacter (
              char c )  [inherited]
```

Definition at line 22 of file font.cpp.

```
{
    text += c;
    textToTexture();
}
```

7.4.3.2 addText()

```
void Object::addText (
            std::string t )  [inherited]
```

Definition at line 16 of file font.cpp.

```
{
    text += t;
    textToTexture();
}
```

7.4.3.3 changeToCircle() [1/5]

```
void Object::changeToCircle ( )  [inherited]
```

Definition at line 5 of file shape.cpp.

```
{
    shapeType = SHAPE::CIRCLE;
    radius = std::min(getW(), getH()) / 2;

    center.x = getX() + getW() / 2;
    center.y = getY() + getH() / 2;
    fillCircleByColor();
}
```

7.4.3.4 changeToCircle() [2/5]

```
void Object::changeToCircle (
            int x,
            int y )  [inherited]
```

Definition at line 24 of file shape.cpp.

```
{
    changeToCircle({x, y});
}
```

7.4.3.5 changeToCircle() [3/5]

```
void Object::changeToCircle (
            int x,
            int y,
            int r )  [inherited]
```

Definition at line 37 of file shape.cpp.

```
{
    shapeType = SHAPE::CIRCLE;
    radius = r;
    center.x = x;
    center.y = y;
    fillCircleByColor();
}
```

7.4.3.6 changeToCircle() [4/5]

```
void Object::changeToCircle (
            SDL_Point c )  [inherited]
```

Definition at line 15 of file shape.cpp.

```
{
    shapeType = SHAPE::CIRCLE;
    center = c;
    radius = std::min(getW() - c.x, c.x - getX());
    radius = std::min(radius, std::min(getH() - c.y, c.y - getY()));
    fillCircleByColor();
}
```

7.4.3.7 changeToCircle() [5/5]

```
void Object::changeToCircle (
            SDL_Point c,
            int r )  [inherited]
```

Definition at line 29 of file shape.cpp.

```
{
    shapeType = SHAPE::CIRCLE;
    radius = r;
    center = c;
    fillCircleByColor();
}
```

7.4.3.8 changeToRectangle()

```
void Object::changeToRectangle ( )  [inherited]
```

Definition at line 46 of file shape.cpp.

```
{
    shapeType = SHAPE::RECTANGLE;
    fillRectangleByColor();
}
```

7.4.3.9 closeScript()

```
void DataStructures::closeScript ( )
```

Definition at line 78 of file step.cpp.

```
{
    switch(type)
    {
        case DATA_STRUCTURES_TYPE::AVL:
            if(avl == nullptr) return ;
            avl->closeScript();
            break;
        case DATA_STRUCTURES_TYPE::TRIE:
            if(trie == nullptr) return ;
            trie->closeScript();
            break;
        case DATA_STRUCTURES_TYPE::HASH_TABLE:
            if(hashTable == nullptr) return ;
            hashTable->closeScript();
            break;
        case DATA_STRUCTURES_TYPE::MIN_HEAP:
            if(minheap == nullptr) return ;
            minheap->closeScript();
            break;
        case DATA_STRUCTURES_TYPE::MAX_HEAP:
            if(minheap == nullptr) return ;
            minheap->closeScript();
            break;
        default:
            break;
    }
}
```

7.4.3.10 coloring() [1/3]

```
void Object::coloring (
               const json & mem )   [inherited]
```

Definition at line 30 of file coloring.cpp.

```
{
    if(mem.contains("r") && mem.contains("g") && mem.contains("b"))
    {
        if(mem.contains("a")) coloring(mem["r"], mem["g"], mem["b"], mem["a"]);
        else coloring(mem["r"], mem["g"], mem["b"], 255);
    }
}
```

7.4.3.11 coloring() [2/3]

```
void Object::coloring (
               int r,
               int g,
               int b,
               int a )   [inherited]
```

Definition at line 8 of file coloring.cpp.

```
{
    if(color == nullptr) color = new SDL_Color;
    color->r = r;
    color->g = g;
    color->b = b;
    color->a = a;

    fillWithColor();
}
```

7.4.3.12 coloring() [3/3]

```
void Object::coloring (
               SDL_Color c )   [inherited]
```

Definition at line 19 of file coloring.cpp.

```
{
    if(color == nullptr) color = new SDL_Color;
    color->r = c.r;
    color->g = c.g;
    color->b = c.b;
    color->a = c.a;

    fillWithColor();
}
```

7.4.3.13 cropping() [1/3]

```
void Object::cropping (
            const json & mem )  [inherited]
```

Definition at line 26 of file cropping.cpp.

```
{
    if(mem.contains("x") && mem.contains("y") && mem.contains("w") && mem.contains("h"))
        cropping(mem["x"], mem["y"], mem["w"], mem["h"]);
}
```

7.4.3.14 cropping() [2/3]

```
void Object::cropping (
            int x,
            int y,
            int w,
            int h )  [inherited]
```

Definition at line 8 of file cropping.cpp.

```
{
    if(crop == nullptr) crop = new SDL_Rect;
    crop->x = x;
    crop->y = y;
    crop->w = w;
    crop->h = h;
}
```

7.4.3.15 cropping() [3/3]

```
void Object::cropping (
            SDL_Rect c )  [inherited]
```

Definition at line 17 of file cropping.cpp.

```
{
    if(crop == nullptr) crop = new SDL_Rect;
    crop->x = c.x;
    crop->y = c.y;
    crop->w = c.w;
    crop->h = c.h;
}
```

7.4.3.16 Dijkstra()

```
void DataStructures::Dijkstra (
            InputBox * inp )  [protected]
```

7.4.3.17 dijkstra()

```
void DataStructures::dijkstra (
            InputBox * inp )
```

Definition at line 3 of file dijkstra.cpp.

```
{
    int src = NUMBER::stringToInt(inp->getText(1));
    int dst = NUMBER::stringToInt(inp->getText(2));
    graph->Dijkstra(src, dst);
}
```

7.4.3.18 fillCircleByColor()

```
void Object::fillCircleByColor ( )  [protected], [inherited]
```

Definition at line 91 of file shape.cpp.

```
{
    if(location == nullptr) locating(0, 0, 0, 0);

    if(texture != nullptr) SDL_DestroyTexture(texture);
    texture = nullptr;

    Uint32 rmask, gmask, bmask, amask;
    Uint32 pixelColor;
#if SDL_BYTEORDER == SDL_BIG_ENDIAN
    rmask = 0xff000000;
    gmask = 0x00ff0000;
    bmask = 0x0000ff00;
    amask = 0x000000ff;
    pixelColor = (color->r « 24) | (color->g « 16) | (color->b « 8) | color->a;
#else
    rmask = 0x000000ff;
    gmask = 0x0000ff00;
    bmask = 0x00ff0000;
    amask = 0xff000000;
    pixelColor = (color->a « 24) | (color->b « 16) | (color->g « 8) | color->r;
#endif

    SDL_Surface *surf = SDL_CreateRGBSurface(0, getW(), getH(), 32, rmask, gmask, bmask, amask);
    SDL_SetSurfaceBlendMode(surf, SDL_BLENDMODE_BLEND);

    texture = SDL_CreateTextureFromSurface(render, surf);
    SDL_FreeSurface(surf);

    Uint32 *pixels = new Uint32[getW() * getH()];
    memset(pixels, 0, getW() * getH() * sizeof(Uint32));

    SDL_Point p = {getW() / 2, getH() / 2};
    center = p;

    if(radius > std::min(getW(), getH()) / 2) radius = std::min(getW(), getH()) / 2;

    for(int i = p.x - radius; i <= p.x + radius; i++)
        for(int j = p.y - radius; j <= p.y + radius; j++)
            if((i - p.x) * (i - p.x) + (j - p.y) * (j - p.y) <= radius * radius)
            {
                int index = i * getW() + j;
                if(index < 0 || index >= getW() * getH()) continue;
                pixels[index] = pixelColor;
            }

    SDL_UpdateTexture(texture, nullptr, pixels, getW() * sizeof(Uint32));
    delete[] pixels;
}
```

7.4.3.19 fillRectangleByColor()

```
void Object::fillRectangleByColor ( )  [protected], [inherited]
```

Definition at line 74 of file shape.cpp.

```
{
    if(location == nullptr) locating(0, 0, 0, 0);

    if(texture != nullptr) SDL_DestroyTexture(texture);
    texture = nullptr;

    SDL_Surface* surf = SDL_CreateRGBSurfaceWithFormat(0, getW(), getH(), 32, SDL_PIXELFORMAT_RGBA32);
    SDL_SetSurfaceBlendMode(surf, SDL_BLENDMODE_BLEND);

    SDL_FillRect(surf, nullptr, SDL_MapRGBA(surf->format, color->r, color->g, color->b, color->a));

    texture = SDL_CreateTextureFromSurface(render, surf);

    SDL_FreeSurface(surf);
}
```

7.4.3.20 fillWithColor()

```
void Object::fillWithColor ( )  [protected], [inherited]
```

Definition at line 39 of file coloring.cpp.

```
{
    if(shapeType == SHAPE::NONE) return fillRectangleByColor();
    if(shapeType == SHAPE::RECTANGLE) return fillRectangleByColor();
    if(shapeType == SHAPE::CIRCLE) return fillCircleByColor();
}
```

7.4.3.21 fitTheTexture()

```
void Object::fitTheTexture ( )  [inherited]
```

Definition at line 140 of file locating.cpp.

```
{
    if(texture == nullptr) return;
    SDL_QueryTexture(texture, nullptr, nullptr, &location->w, &location->h);
}
```

7.4.3.22 getColor()

```
const SDL_Color * Object::getColor ( )  [inherited]
```

Definition at line 3 of file coloring.cpp.

```
{
    return color;
}
```

7.4.3.23 getCrop()

```
const SDL_Rect * Object::getCrop ( )  [inherited]
```

Definition at line 3 of file cropping.cpp.

```
{
    return crop;
}
```

7.4.3.24 getDataType()

```
DATA_STRUCTURES_TYPE DataStructures::getDataType ( )
```

Definition at line 4 of file operator.cpp.

```
{
    return type;
}
```

7.4.3.25 getH()

```
int Object::getH ( )  [inherited]
```

Definition at line 47 of file locating.cpp.

```
{
    return location->h;
}
```

7.4.3.26 getLocation()

```
const SDL_Rect * Object::getLocation ( )  [inherited]
```

Definition at line 27 of file locating.cpp.

```
{
    return location;
}
```

7.4.3.27 getName()

```
std::string DataStructures::getName ( )
```

Definition at line 9 of file operator.cpp.

```
{
    return name;
}
```

7.4.3.28 getSize()

```
int Object::getSize ( )  [inherited]
```

Definition at line 68 of file font.cpp.

```
{
    return text.size();
}
```

7.4.3.29 getText()

```
std::string Object::getText ( )  [inherited]
```

Definition at line 63 of file font.cpp.

```
{
    return text;
}
```

7.4.3.30 getW()

```
int Object::getW ( )  [inherited]
```

Definition at line 42 of file locating.cpp.

```
{
    return location->w;
}
```

7.4.3.31 getX()

```
int Object::getX ( )  [inherited]
```

Definition at line 32 of file locating.cpp.

```
{
    return location->x;
}
```

7.4.3.32 getY()

```
int Object::getY ( )  [inherited]
```

Definition at line 37 of file locating.cpp.

```
{
    return location->y;
}
```

7.4.3.33 goBack()

```
void DataStructures::goBack ( )
```

Definition at line 3 of file step.cpp.

```
{

}
```

7.4.3.34 goNext()

```
void DataStructures::goNext ( )
```

Definition at line 9 of file step.cpp.

```
{
    switch(type)
    {
        case DATA_STRUCTURES_TYPE::AVL:
            if(avl == nullptr) return ;
            avl->goNext();
            break;
        default:
            break;
    }
}
```

7.4.3.35 goOff()

```
void DataStructures::goOff ( )
```

Definition at line 37 of file step.cpp.

```
{
    switch(type)
    {
        case DATA_STRUCTURES_TYPE::AVL:
            if(avl == nullptr) return ;
            avl->goOff();
            break;
        default:
            break;
    }
}
```

7.4.3.36 goOn()

```
void DataStructures::goOn ( )
```

Definition at line 23 of file step.cpp.

```
{
    switch(type)
    {
        case DATA_STRUCTURES_TYPE::AVL:
            if(avl == nullptr) return ;
            avl->goOn();
            break;
        default:
            break;
    }
}
```

7.4.3.37 hide()

```
void Object::hide ( )  [inherited]
```

Definition at line 8 of file visible.cpp.

```
{
    visible = false;
}
```

7.4.3.38 importFromJson() [1/2]

```
void DataStructures::importFromJson ( )  [protected]
```

Definition at line 51 of file constructor.cpp.

```
{
    json* mem = JSON::readFile(PATH::ATB::DATA_STRUCTURES_+ name + ".json");

    if(mem->contains("background"))
        initBackground((*mem)["background"]);
    if(mem->contains(("sprite-structure")))
        initLinker((*mem)["sprite-structure"]);
    if(mem->contains("display"))
        initDisplay((*mem)["display"]);
}
```

7.4.3.39 importFromJson() [2/2]

```
void Object::importFromJson (
            const json & mem )  [inherited]
```

Definition at line 21 of file constructor.cpp.

```
{
    if(mem.contains("location"))
        locating(mem["location"]);

    if(mem.contains("crop"))
        cropping(mem["crop"]);

    if(mem.contains("color"))
        coloring(mem["color"]);

    if(mem.contains("shape"))
        setShape(mem["shape"]);

    if(mem.contains("visible"))
        visible = mem["visible"];

    if(mem.contains("image"))
        textureFromFile(PATH::ASSETS::GRAPHICS_ + mem["image"].get<std::string>());
    return ;
}
```

7.4.3.40 init()

```
void DataStructures::init (
            InputBox * inp )
```

Definition at line 14 of file operator.cpp.

```
{
    switch(type)
    {
        case DATA_STRUCTURES_TYPE::AVL:
            initAVL(inp);
            break;
        case DATA_STRUCTURES_TYPE::TRIE:
            initTrie(inp);
            break;
        case DATA_STRUCTURES_TYPE::HASH_TABLE:
            initHashTable(inp);
            break;
        case DATA_STRUCTURES_TYPE::MIN_HEAP:
            initMinHeap(inp);
            break;
        case DATA_STRUCTURES_TYPE::MAX_HEAP:
            initMinHeap(inp);
            break;
        case DATA_STRUCTURES_TYPE::BTREE_4TH:
            //initBTree4th(inp);
            break;
        case DATA_STRUCTURES_TYPE::GRAPH:
            initGraph(inp);
            break;
        case DATA_STRUCTURES_TYPE::NONE:
            break;
    }
}
```

7.4.3.41 initAVL()

```
void DataStructures::initAVL (
            InputBox * inp )  [protected]
```

Definition at line 3 of file init.cpp.

```
{
    std::vector<int> v = NUMBER::stringToArray(inp->getText(1));
    avl->init(v);
}
```

7.4.3.42 initBackground()

```
void DataStructures::initBackground (
            const json & mem )  [protected]
```

Definition at line 14 of file constructor.cpp.

```
{
    Object::importFromJson(mem);
}
```

7.4.3.43 initDisplay()

```
void DataStructures::initDisplay (
            const json & mem )  [protected]
```

Definition at line 32 of file constructor.cpp.

```
{

    for(auto& i : mem)
    {
        SDL_Rect viewport = {0, 0, 0, 0};
        if(i.contains("viewport"))
        {
            viewport.x = i["viewport"]["x"];
            viewport.y = i["viewport"]["y"];
            viewport.w = i["viewport"]["w"];
            viewport.h = i["viewport"]["h"];
        }
        displays.push_back(new Display(render, viewport));
        if(i.contains("name"))
            displays.back()->linking(i["name"].get<std::string>());
    }
}
```

7.4.3.44 initGraph()

```
void DataStructures::initGraph (
            InputBox * inp )  [protected]
```

Definition at line 3 of file init.cpp.

```
{
    std::vector<std::vector<int> > g;

    g.resize(capacity);

    for(int i = 0; i < capacity; i++)
     {
        g[i].resize(capacity);
        for(int j = 0; j < capacity; j++)
        {
            g[i][j] = NUMBER::stringToInt(inp->getText(i * capacity + j + 1));
        }
     }

     graph->init(g);
}
```

7.4.3.45 initHashTable()

```
void DataStructures::initHashTable (
            InputBox * inp )  [protected]
```

Definition at line 3 of file init.cpp.

```
{
    int HASH_KEY = NUMBER::stringToInt(inp->getText(1));
    std::vector<int> v = NUMBER::stringToArray(inp->getText(2));
    hashTable->init(v, HASH_KEY);
}
```

7.4.3.46 initLinker()

```
void DataStructures::initLinker (
                const json & mem )  [protected]
```

Definition at line 25 of file constructor.cpp.

```
{
    if(!mem.contains("name"))
        return ;
    spriteLinker = mem["name"];
}
```

7.4.3.47 initMinHeap()

```
void DataStructures::initMinHeap (
                InputBox * inp )  [protected]
```

Definition at line 3 of file init.cpp.

```
{
    std::vector<int> v = NUMBER::stringToArray(inp->getText(1));
    minheap->init(v);
}
```

7.4.3.48 initTrie()

```
void DataStructures::initTrie (
                InputBox * inp )  [protected]
```

Definition at line 3 of file init.cpp.

```
{
    std::vector<std::string> v = SIUSTRING::split(inp->getText(1));
    trie->init(v);
}
```

7.4.3.49 insert()

```
void DataStructures::insert (
                InputBox * inp )
```

Definition at line 44 of file operator.cpp.

```
{
    switch(type)
    {
        case DATA_STRUCTURES_TYPE::AVL:
            insertAVL(inp);
            break;
        case DATA_STRUCTURES_TYPE::TRIE:
            insertTrie(inp);
            break;
        case DATA_STRUCTURES_TYPE::HASH_TABLE:
            insertHashTable(inp);
            break;
        case DATA_STRUCTURES_TYPE::MIN_HEAP:
            insertMinHeap(inp);
            break;
        default:
            break;
        case DATA_STRUCTURES_TYPE::MAX_HEAP:
            insertMinHeap(inp);
            break;
        case DATA_STRUCTURES_TYPE::BTREE_4TH:
            break;
        case DATA_STRUCTURES_TYPE::GRAPH:
            break;
        case DATA_STRUCTURES_TYPE::NONE:
            break;
    }
}
```

7.4.3.50 insertAVL()

```
void DataStructures::insertAVL (
            InputBox * inp ) [protected]
```

Definition at line 3 of file insert.cpp.

```
{
    int value = NUMBER::stringToInt(inp->getText(1));
    avl->insert(value);
}
```

7.4.3.51 insertHashTable()

```
void DataStructures::insertHashTable (
            InputBox * inp ) [protected]
```

Definition at line 4 of file insert.cpp.

```
{
    int value = NUMBER::stringToInt(inp->getText(1));
    hashTable->insert(value);
}
```

7.4.3.52 insertMinHeap()

```
void DataStructures::insertMinHeap (
            InputBox * inp ) [protected]
```

Definition at line 3 of file insert.cpp.

```
{
    int value = NUMBER::stringToInt(inp->getText(1));
    minheap->insert(value);
}
```

7.4.3.53 insertTrie()

```
void DataStructures::insertTrie (
            InputBox * inp ) [protected]
```

Definition at line 3 of file insert.cpp.

```
{
    std::string value = inp->getText(1);
    trie->insert(value);
}
```

7.4.3.54 isLieInside() [1/4]

```
bool Object::isLieInside (
            int x,
            int y )  [inherited]
```

Definition at line 3 of file locating.cpp.

```
{
    if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
    return (x >= location->x && x < location->x + location->w && y >= location->y && y < location->y +
     location->h);
}
```

7.4.3.55 isLieInside() [2/4]

```
bool Object::isLieInside (
            int x,
            int y,
            int w,
            int h )  [inherited]
```

Definition at line 21 of file locating.cpp.

```
{
    if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
    return (x >= location->x && x + w <= location->x + location->w && y >= location->y && y + h <=
    location->y + location->h);
}
```

7.4.3.56 isLieInside() [3/4]

```
bool Object::isLieInside (
            SDL_Point p )  [inherited]
```

Definition at line 9 of file locating.cpp.

```
{
    if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
    return (p.x >= location->x && p.x < location->x + location->w && p.y >= location->y && p.y <
    location->y + location->h);
}
```

7.4.3.57 isLieInside() [4/4]

```
bool Object::isLieInside (
            SDL_Rect r )  [inherited]
```

Definition at line 15 of file locating.cpp.

```
{
    if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
    return (r.x >= location->x && r.x + r.w <= location->x + location->w && r.y >= location->y && r.y +
    r.h <= location->y + location->h);
}
```

7.4.3.58 isReceiveEvent()

```
bool DataStructures::isReceiveEvent (
            SDL_Event & e )
```

Definition at line 6 of file event.cpp.

```
{
    switch(e.type)
    {
        case SDL_QUIT:
            return false;
            break;
        default:
            if(avl != nullptr && avl->isReceiveEvent(e))
                return true;
            if(trie != nullptr && trie->isReceiveEvent(e))
                return true;
            if(hashTable != nullptr && hashTable->isReceiveEvent(e))
                return true;
            if(minheap != nullptr && minheap->isReceiveEvent(e))
                return true;
            if(graph != nullptr && graph->isReceiveEvent(e))
                return true;
            for(auto &i : displays)
                if(i->isReceiveEvent(e))
                    return true;
            return false;
            break;
    }
}
```

7.4.3.59 isVisible()

```
bool Object::isVisible ( ) [inherited]
```

Definition at line 13 of file visible.cpp.

```
{
    return visible;
}
```

7.4.3.60 linking()

```
void DataStructures::linking (
            std::string n )
```

Definition at line 19 of file constructor.cpp.

```
{
    name = n;
    importFromJson();
}
```

7.4.3.61 locating() [1/3]

```
void Object::locating (
            const json & mem ) [virtual], [inherited]
```

Definition at line 70 of file locating.cpp.

```
{
    if(mem.contains("x") && mem.contains("y") && mem.contains("w") && mem.contains("h"))
        locating(mem["x"], mem["y"], mem["w"], mem["h"]);
}
```

7.4.3.62 locating() [2/3]

```
void Object::locating (
             int x,
             int y,
             int w,
             int h ) [virtual], [inherited]
```

Reimplemented in Sprite.

Definition at line 52 of file locating.cpp.

```
{
    if(location == nullptr) location = new SDL_Rect;
    location->x = x;
    location->y = y;
    location->w = w;
    location->h = h;
}
```

7.4.3.63 locating() [3/3]

```
void Object::locating (
             SDL_Rect l ) [virtual], [inherited]
```

Reimplemented in Sprite.

Definition at line 61 of file locating.cpp.

```
{
    if(location == nullptr) location = new SDL_Rect;
    location->x = l.x;
    location->y = l.y;
    location->w = l.w;
    location->h = l.h;
}
```

7.4.3.64 locatingH()

```
void Object::locatingH (
             int h ) [virtual], [inherited]
```

Definition at line 94 of file locating.cpp.

```
{
    if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
    location->h = h;
}
```

7.4.3.65 locatingW()

```
void Object::locatingW (
             int w ) [virtual], [inherited]
```

Definition at line 88 of file locating.cpp.

```
{
    if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
    location->w = w;
}
```

7.4.3.66 locatingX()

```
void Object::locatingX (
            int x )  [virtual], [inherited]
```

Reimplemented in Sprite.

Definition at line 76 of file locating.cpp.

```
{
    if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
    location->x = x;
}
```

7.4.3.67 locatingY()

```
void Object::locatingY (
            int y )  [virtual], [inherited]
```

Reimplemented in Sprite.

Definition at line 82 of file locating.cpp.

```
{
    if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
    location->y = y;
}
```

7.4.3.68 moveX()

```
void Object::moveX (
            int delta )  [virtual], [inherited]
```

Reimplemented in Sprite.

Definition at line 100 of file locating.cpp.

```
{
    if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
    location->x += delta;
}
```

7.4.3.69 moveY()

```
void Object::moveY (
            int delta )  [virtual], [inherited]
```

Reimplemented in Sprite.

Definition at line 106 of file locating.cpp.

```
{
    if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
    location->y += delta;
}
```

7.4.3.70 MST()

```
void DataStructures::MST ( ) [protected]
```

7.4.3.71 mst()

```
void DataStructures::mst ( )
```

Definition at line 3 of file mst.cpp.

```
{
    graph->MST();
}
```

7.4.3.72 noCropping()

```
void Object::noCropping ( ) [inherited]
```

Definition at line 32 of file cropping.cpp.

```
{
    if(crop != nullptr) delete crop;
    crop = nullptr;
}
```

7.4.3.73 react()

```
Button * DataStructures::react (
               SDL_Event & e )
```

Definition at line 33 of file event.cpp.

```
{
    Button* but = nullptr;
    if(avl != nullptr && avl->isReceiveEvent(e))
         but = avl->react(e);
    if(but == nullptr && trie != nullptr && trie->isReceiveEvent(e))
        but = trie->react(e);
    if(but == nullptr && hashTable != nullptr && hashTable->isReceiveEvent(e))
        but = hashTable->react(e);
    if(but == nullptr && minheap != nullptr && minheap->isReceiveEvent(e))
        but = minheap->react(e);
    if(but == nullptr && graph != nullptr && graph->isReceiveEvent(e))
        but = graph->react(e);
    if(but != nullptr) return but;
    for(auto &i : displays)
        if(i->isReceiveEvent(e))
            but = i->react(e);
    return but;
}
```

7.4.3.74 remove()

```
void DataStructures::remove (
            InputBox * inp )
```

Definition at line 74 of file operator.cpp.

```
{
    switch(type)
    {
        case DATA_STRUCTURES_TYPE::AVL:
            removeAVL(inp);
            break;
        case DATA_STRUCTURES_TYPE::TRIE:
            removeTrie(inp);
            break;
        case DATA_STRUCTURES_TYPE::HASH_TABLE:
            removeHashTable(inp);
            break;
        case DATA_STRUCTURES_TYPE::MIN_HEAP:
            removeMinHeap(inp);
            break;
        case DATA_STRUCTURES_TYPE::MAX_HEAP:
            removeMinHeap(inp);
            break;
        case DATA_STRUCTURES_TYPE::BTREE_4TH:
            break;
        case DATA_STRUCTURES_TYPE::GRAPH:
            break;
        case DATA_STRUCTURES_TYPE::NONE:
            break;
    }
}
```

7.4.3.75 removeAVL()

```
void DataStructures::removeAVL (
            InputBox * inp ) [protected]
```

Definition at line 3 of file remove.cpp.

```
{
    int value = NUMBER::stringToInt(inp->getText(1));

    avl->remove(value);
}
```

7.4.3.76 removeCharacter() [1/2]

```
void Object::removeCharacter ( ) [inherited]
```

Definition at line 28 of file font.cpp.

```
{
    if (text.size() > 0)
        text.pop_back();
    textToTexture();
}
```

7.4.3.77 removeCharacter() [2/2]

```
void Object::removeCharacter (
             int n )  [inherited]
```

Definition at line 35 of file font.cpp.

```
{
    if(n == 0) return ;
    if(text.size() <= n) text.clear();
    else text.erase(text.end() - n, text.end());
    textToTexture();
}
```

7.4.3.78 removeHashTable()

```
void DataStructures::removeHashTable (
             InputBox * inp )  [protected]
```

Definition at line 3 of file remove.cpp.

```
{
    int value = NUMBER::stringToInt(inp->getText(1));

    hashTable->remove(value);
}
```

7.4.3.79 removeMinHeap()

```
void DataStructures::removeMinHeap (
             InputBox * inp )  [protected]
```

Definition at line 3 of file pop.cpp.

```
{
    int value = NUMBER::stringToInt(inp->getText(1));

    while(value--) minheap->pop();
}
```

7.4.3.80 removeTrie()

```
void DataStructures::removeTrie (
             InputBox * inp )  [protected]
```

Definition at line 3 of file remove.cpp.

```
{
    trie->remove(inp->getText(1));
}
```

7.4.3.81 rendering()

```
void DataStructures::rendering ( ) [virtual]
```

Reimplemented from Object.

Definition at line 3 of file rendering.cpp.

```
{
    Object::rendering();
    for(auto& i : node)
        i->rendering();
    for(auto& i : displays)
        i->rendering();
     if(avl != nullptr)
         avl->rendering();
     if(trie != nullptr)
         trie->rendering();
     if(hashTable != nullptr)
         hashTable->rendering();
     if(minheap != nullptr)
         minheap->rendering();
     if(graph != nullptr)
         graph->rendering();
}
```

7.4.3.82 SCC()

```
void DataStructures::SCC ( ) [protected]
```

7.4.3.83 scc()

```
void DataStructures::scc ( )
```

Definition at line 3 of file scc.cpp.

```
{
    graph->SCC();
}
```

7.4.3.84 search()

```
void DataStructures::search (
               InputBox * inp )
```

Definition at line 102 of file operator.cpp.

```
{
    switch(type)
    {
        case DATA_STRUCTURES_TYPE::AVL:
            searchAVL(inp);
            break;
        case DATA_STRUCTURES_TYPE::TRIE:
            searchTrie(inp);
            break;
        case DATA_STRUCTURES_TYPE::HASH_TABLE:
            searchHashTable(inp);
            break;
        default:
            break;
    }
}
```

7.4.3.85 searchAVL()

```
void DataStructures::searchAVL (
            InputBox * inp )   [protected]
```

Definition at line 3 of file search.cpp.

```
{
    int value = NUMBER::stringToInt(inp->getText(1));

    avl->search(value);
}
```

7.4.3.86 searchHashTable()

```
void DataStructures::searchHashTable (
            InputBox * inp )   [protected]
```

Definition at line 3 of file search.cpp.

```
{
    int value = NUMBER::stringToInt(inp->getText(1));

    hashTable->search(value);
}
```

7.4.3.87 searchMinHeap()

```
void DataStructures::searchMinHeap (
            InputBox * inp )   [protected]
```

7.4.3.88 searchTrie()

```
void DataStructures::searchTrie (
            InputBox * inp )   [protected]
```

Definition at line 3 of file search.cpp.

```
{
    trie->search(inp->getText(1));
}
```

7.4.3.89 setDataType()

```
void DataStructures::setDataType (
            DATA_STRUCTURES_TYPE t )
```

Definition at line 63 of file constructor.cpp.

```
{
    type = t;
    switch(type)
    {
        case DATA_STRUCTURES_TYPE::AVL:
            linking("AVL");
            avl = new AVL(render, ds_mutex, font, {10, 10, 800, 600}, 128);
            break;
        case DATA_STRUCTURES_TYPE::TRIE:
            linking("trie");
            trie = new Trie(render, ds_mutex, font, {10, 10, 800, 600}, 3000);
            break;
        case DATA_STRUCTURES_TYPE::HASH_TABLE:
            linking("hash_table");
            hashTable = new HashTable(render, ds_mutex, font, {10, 10, 800, 600}, 128);
            break;
        case DATA_STRUCTURES_TYPE::GRAPH:
            linking("graph");
            graph = new Graph(render, ds_mutex, font, {10, 10, 800, 600}, 128);
            break;
        case DATA_STRUCTURES_TYPE::BTREE_4TH:
            linking("btree4th");
            break;
        case DATA_STRUCTURES_TYPE::MIN_HEAP:
            linking("minheap");
            minheap = new minHeap(render, ds_mutex, font, {10, 10, 800, 600}, 128);
            minheap->setmin();
            break;
        case DATA_STRUCTURES_TYPE::MAX_HEAP:
            linking("minheap");
            minheap = new minHeap(render, ds_mutex, font, {10, 10, 800, 600}, 128);
            minheap->setmax();
            break;
        case DATA_STRUCTURES_TYPE::NONE:
            break;
    }
}
```

7.4.3.90 setFont()

```
void Object::setFont (
            TTF_Font * f )   [inherited]
```

Definition at line 4 of file font.cpp.

```
{
    font = f;
    textToTexture();
}
```

7.4.3.91 setShape()

```
void Object::setShape (
            const json & mem )   [inherited]
```

Definition at line 52 of file shape.cpp.

```
{
    if(mem["type"].get<std::string>() == "CIRCLE")
    {
        if(mem.contains("center"))
        {
            if(mem.contains("radius"))
                changeToCircle(mem["center"]["x"], mem["center"]["y"], mem["radius"]);
            else changeToCircle(mem["center"]["x"], mem["center"]["y"]);
        }else changeToCircle();

        return ;
    }

    if(mem["type"].get<std::string>() == "NONE" || mem["type"].get<std::string>() == "RECTANGLE")
    {
        changeToRectangle();
        return ;
    }

}
```

7.4.3.92 setText()

```
void Object::setText (
              std::string t )  [inherited]
```

Definition at line 10 of file font.cpp.

```
{
    text = t;
    textToTexture();
}
```

7.4.3.93 setting()

```
void DataStructures::setting (
              InputBox * inp )
```

Definition at line 152 of file operator.cpp.

```
{
    switch(type)
    {
        case DATA_STRUCTURES_TYPE::AVL:
            settingAVL(inp);
            break;
        case DATA_STRUCTURES_TYPE::TRIE:
            settingTrie(inp);
            break;
        case DATA_STRUCTURES_TYPE::GRAPH:
            settingGraph(inp);
            break;
        case DATA_STRUCTURES_TYPE::MAX_HEAP:
            settingMinHeap(inp);
            break;
        case DATA_STRUCTURES_TYPE::MIN_HEAP:
            settingMinHeap(inp);
            break;
        case DATA_STRUCTURES_TYPE::HASH_TABLE:
            settingHashTable(inp);
            break;
        case DATA_STRUCTURES_TYPE::BTREE_4TH:
            break;
        case DATA_STRUCTURES_TYPE::NONE:
            break;
    }
}
```

7.4.3.94 settingAVL()

```
void DataStructures::settingAVL (
            InputBox * inp )  [protected]
```

Definition at line 5 of file setting.cpp.

```
{
    std::vector<int> v1 = NUMBER::stringToArray(inp->getText(1));
    std::vector<int> v2 = NUMBER::stringToArray(inp->getText(2));
    std::vector<int> v3 = NUMBER::stringToArray(inp->getText(3));
    std::vector<int> v4 = NUMBER::stringToArray(inp->getText(4));

    SDL_Color c1;
    c1.r = v1[0];
    c1.g = v1[1];
    c1.b = v1[2];
    if(v1.size() >= 4) c1.a = v1[3];
    else c1.a = 255;

    SDL_Color c2;
    c2.r = v2[0];
    c2.g = v2[1];
    c2.b = v2[2];
    if(v2.size() >= 4) c2.a = v2[3];
    else c2.a = 255;

    SDL_Color c3;
    c3.r = v3[0];
    c3.g = v3[1];
    c3.b = v3[2];
    if(v3.size() >= 4) c3.a = v3[3];
    else c3.a = 255;

    SDL_Color c4;
    c4.r = v4[0];
    c4.g = v4[1];
    c4.b = v4[2];
    if(v4.size() >= 4) c4.a = v4[3];
    else c4.a = 255;

    Object::coloring(c1);
    Object::fillWithColor();

    avl->setting(c1, c2, c3, c4);
}
```

7.4.3.95 settingGraph()

```
void DataStructures::settingGraph (
            InputBox * inp )  [protected]
```

Definition at line 5 of file setting.cpp.

```
{
    std::vector<int> v1 = NUMBER::stringToArray(inp->getText(1));
    std::vector<int> v2 = NUMBER::stringToArray(inp->getText(2));
    std::vector<int> v3 = NUMBER::stringToArray(inp->getText(3));
    std::vector<int> v4 = NUMBER::stringToArray(inp->getText(4));

    SDL_Color c1;
    c1.r = v1[0];
    c1.g = v1[1];
    c1.b = v1[2];
    if(v1.size() >= 4) c1.a = v1[3];
    else c1.a = 255;

    SDL_Color c2;
    c2.r = v2[0];
    c2.g = v2[1];
    c2.b = v2[2];
    if(v2.size() >= 4) c2.a = v2[3];
    else c2.a = 255;

    SDL_Color c3;
    c3.r = v3[0];
    c3.g = v3[1];
    c3.b = v3[2];
    if(v3.size() >= 4) c3.a = v3[3];
    else c3.a = 255;

    SDL_Color c4;
    c4.r = v4[0];
    c4.g = v4[1];
    c4.b = v4[2];
    if(v4.size() >= 4) c4.a = v4[3];
    else c4.a = 255;

    Object::coloring(c1);
    Object::fillWithColor();

    graph->setting(c1, c2, c3, c4);
}
```

7.4.3.96 settingHashTable()

```
void DataStructures::settingHashTable (
            InputBox * inp ) [protected]
```

Definition at line 5 of file setting.cpp.

```
{
    std::vector<int> v1 = NUMBER::stringToArray(inp->getText(1));
    std::vector<int> v2 = NUMBER::stringToArray(inp->getText(2));
    std::vector<int> v3 = NUMBER::stringToArray(inp->getText(3));
    std::vector<int> v4 = NUMBER::stringToArray(inp->getText(4));

    SDL_Color c1;
    c1.r = v1[0];
    c1.g = v1[1];
    c1.b = v1[2];
    if(v1.size() >= 4) c1.a = v1[3];
    else c1.a = 255;

    SDL_Color c2;
    c2.r = v2[0];
    c2.g = v2[1];
    c2.b = v2[2];
    if(v2.size() >= 4) c2.a = v2[3];
    else c2.a = 255;

    SDL_Color c3;
    c3.r = v3[0];
    c3.g = v3[1];
    c3.b = v3[2];
    if(v3.size() >= 4) c3.a = v3[3];
    else c3.a = 255;

    SDL_Color c4;
    c4.r = v4[0];
    c4.g = v4[1];
    c4.b = v4[2];
    if(v4.size() >= 4) c4.a = v4[3];
    else c4.a = 255;

    Object::coloring(c1);
    Object::fillWithColor();

    hashTable->setting(c1, c2, c3, c4);
}
```

7.4.3.97 settingMinHeap()

```
void DataStructures::settingMinHeap (
            InputBox * inp ) [protected]
```

Definition at line 5 of file setting.cpp.

```
{
    std::vector<int> v1 = NUMBER::stringToArray(inp->getText(1));
    std::vector<int> v2 = NUMBER::stringToArray(inp->getText(2));
    std::vector<int> v3 = NUMBER::stringToArray(inp->getText(3));
     std::vector<int> v4 = NUMBER::stringToArray(inp->getText(4));

     SDL_Color c1;
     c1.r = v1[0];
     c1.g = v1[1];
     c1.b = v1[2];
     if(v1.size() >= 4) c1.a = v1[3];
     else c1.a = 255;

     SDL_Color c2;
     c2.r = v2[0];
     c2.g = v2[1];
     c2.b = v2[2];
     if(v2.size() >= 4) c2.a = v2[3];
     else c2.a = 255;

     SDL_Color c3;
     c3.r = v3[0];
     c3.g = v3[1];
     c3.b = v3[2];
     if(v3.size() >= 4) c3.a = v3[3];
     else c3.a = 255;

     SDL_Color c4;
     c4.r = v4[0];
     c4.g = v4[1];
     c4.b = v4[2];
     if(v4.size() >= 4) c4.a = v4[3];
     else c4.a = 255;

     Object::coloring(c1);
     Object::fillWithColor();

     minheap->setting(c1, c2, c3, c4);
 }
```

7.4.3.98 settingTrie()

```
void DataStructures::settingTrie (
               InputBox * inp )  [protected]
```

Definition at line 5 of file setting.cpp.

```
{
    std::vector<int> v1 = NUMBER::stringToArray(inp->getText(1));
    std::vector<int> v2 = NUMBER::stringToArray(inp->getText(2));
    std::vector<int> v3 = NUMBER::stringToArray(inp->getText(3));
     std::vector<int> v4 = NUMBER::stringToArray(inp->getText(4));

     SDL_Color c1;
     c1.r = v1[0];
     c1.g = v1[1];
     c1.b = v1[2];
     if(v1.size() >= 4) c1.a = v1[3];
     else c1.a = 255;

     SDL_Color c2;
     c2.r = v2[0];
     c2.g = v2[1];
     c2.b = v2[2];
     if(v2.size() >= 4) c2.a = v2[3];
     else c2.a = 255;

     SDL_Color c3;
     c3.r = v3[0];
     c3.g = v3[1];
     c3.b = v3[2];
     if(v3.size() >= 4) c3.a = v3[3];
     else c3.a = 255;

     SDL_Color c4;
     c4.r = v4[0];
     c4.g = v4[1];
     c4.b = v4[2];
     if(v4.size() >= 4) c4.a = v4[3];
    else c4.a = 255;

    Object::coloring(c1);
    Object::fillWithColor();

    trie->setting(c1, c2, c3, c4);
}
```

7.4.3.99 show()

```
void Object::show ( )  [inherited]
```

Definition at line 3 of file visible.cpp.

```
{
    visible = true;
}
```

7.4.3.100 size()

```
void DataStructures::size ( )
```

Definition at line 137 of file operator.cpp.

```
{
    switch(type)
    {
        case DATA_STRUCTURES_TYPE::MIN_HEAP:
            minheap->size();
            break;
        case DATA_STRUCTURES_TYPE::MAX_HEAP:
            minheap->size();
            break;
        default:
            break;
    }
}
```

7.4.3.101 slowDown()

```
void DataStructures::slowDown ( )
```

Definition at line 65 of file step.cpp.

```
{
    switch(type)
    {
        case DATA_STRUCTURES_TYPE::AVL:
            if(avl == nullptr) return ;
            avl->slowDown();
            break;
        default:
            break;
    }
}
```

7.4.3.102 speedUp()

```
void DataStructures::speedUp ( )
```

Definition at line 51 of file step.cpp.

```
{
    switch(type)
    {
        case DATA_STRUCTURES_TYPE::AVL:
            if(avl == nullptr) return ;
            avl->speedUp();
            break;
        default:
            break;
    }
}
```

7.4.3.103 textToTexture()

```
void Object::textToTexture ( )  [protected], [inherited]
```

Definition at line 43 of file font.cpp.

```
{
    if(font == nullptr) return ;
    if(color == nullptr) return ;
    if(render == nullptr) return ;
    if(texture != nullptr)
    {
        SDL_DestroyTexture(texture);
    }
    texture = nullptr;

    SDL_Surface* surface = TTF_RenderText_Blended(font, text.c_str(), *color);

    if(surface == nullptr) return ;

    texture = SDL_CreateTextureFromSurface(render, surface);
    SDL_FreeSurface(surface);
    fitTheTexture();
}
```

7.4.3.104 textureFromFile()

```
void Object::textureFromFile (
              std::string dir )  [inherited]
```

Definition at line 4 of file external_storage.cpp.

```
{
    SDL_Surface *surface = IMG_Load(dir.c_str());

    texture = SDL_CreateTextureFromSurface(render, surface);
    SDL_FreeSurface(surface);
}
```

7.4.3.105 top()

```
void DataStructures::top ( )
```

Definition at line 120 of file operator.cpp.

```
{
    switch(type)
    {
        case DATA_STRUCTURES_TYPE::MIN_HEAP:
            minheap->top();
            break;
        case DATA_STRUCTURES_TYPE::MAX_HEAP:
            minheap->top();
            break;
            break;
        default:
            break;
    }
}
```

7.4.3.106 zoom()

```
void Object::zoom (
              double delta )  [virtual], [inherited]
```

Definition at line 123 of file locating.cpp.

```
{
    if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
    location->w *= delta;
    location->h *= delta;
}
```

7.4.3.107 zoomH()

```
void Object::zoomH (
              int delta )  [virtual], [inherited]
```

Definition at line 118 of file locating.cpp.

```
{
    if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
    location->h += delta;
}
```

7.4.3.108 zoomInMiddle()

```
void Object::zoomInMiddle (
              double delta )  [virtual], [inherited]
```

Definition at line 130 of file locating.cpp.

```
{
    if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
    SDL_Point center = {location->x + location->w / 2, location->y + location->h / 2};
    location->w *= delta;
    location->h *= delta;
    location->x = center.x - location->w / 2;
    location->y = center.y - location->h / 2;
}
```

7.4.3.109 zoomW()

```
void Object::zoomW (
             int delta )  [virtual], [inherited]
```

Definition at line 112 of file locating.cpp.

```
{
    if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
    location->w += delta;
}
```

7.4.4 Member Data Documentation

7.4.4.1 capacity

```
int DataStructures::capacity
```

Definition at line 130 of file data_structures.hpp.

The documentation for this class was generated from the following files:

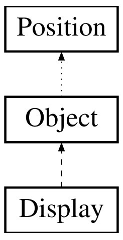
- include/data_structures.hpp
- src/data_structures/constructor.cpp
- src/data_structures/destructor.cpp
- src/data_structures/event.cpp
- src/data_structures/operator/AVL/init.cpp
- src/data_structures/operator/AVL/insert.cpp
- src/data_structures/operator/AVL/remove.cpp
- src/data_structures/operator/AVL/search.cpp
- src/data_structures/operator/AVL/setting.cpp
- src/data_structures/operator/graph/dijkstra.cpp
- src/data_structures/operator/graph/mst.cpp
- src/data_structures/operator/graph/scc.cpp
- src/data_structures/operator/minheap/pop.cpp
- src/data_structures/operator.cpp
- src/data_structures/rendering.cpp
- src/data_structures/step.cpp

7.5 Display Class Reference

container of button intermediate between button and user input, window

```
#include <display.hpp>
```

Inheritance diagram for Display:



Public Member Functions

- Display (SDL_Renderer ∗r, SDL_Rect v)
- ∼Display ()
- void linking (std::string n)
- bool isReceiveEvent (SDL_Event &e)
- Button ∗ react (SDL_Event &e)
- void rendering ()

Protected Member Functions

- void initBackground (const json &mem)
- void initButtons (const json &mem)
- void importFromJson ()
- bool isButtonReceiveEvent (SDL_Event &e)
- void fillWithColor ()
- void fillCircleByColor ()
- void fillRectangleByColor ()
- void textToTexture ()
- const SDL_Rect ∗ getCrop ()
- void cropping (int x, int y, int w, int h)
- void cropping (SDL_Rect c)
- void cropping (const json &mem)
- void noCropping ()
- const SDL_Rect ∗ getLocation ()
- int getX ()
- int getY ()
- int getW ()
- int getH ()
- virtual void locating (int x, int y, int w, int h)
- virtual void locating (SDL_Rect l)
- virtual void locating (const json &mem)
- virtual void locatingX (int x)
- virtual void locatingY (int y)
- virtual void locatingW (int w)
- virtual void locatingH (int h)
- virtual void moveX (int delta)
- virtual void moveY (int delta)
- virtual void zoomW (int delta)
- virtual void zoomH (int delta)
- virtual void zoom (double delta)
- virtual void zoomInMiddle (double delta)
- void fitTheTexture ()
- const SDL_Color ∗ getColor ()
- void coloring (int r, int g, int b, int a)
- void coloring (SDL_Color c)
- void coloring (const json &mem)
- void textureFromFile (std::string dir)
- void changeToCircle ()
- void changeToCircle (SDL_Point c)
- void changeToCircle (int x, int y)
- void changeToCircle (SDL_Point c, int r)
- void changeToCircle (int x, int y, int r)
- void changeToRectangle ()

- void setShape (const json &mem)
- bool isLieInside (int x, int y)
- bool isLieInside (SDL_Point p)
- bool isLieInside (SDL_Rect r)
- bool isLieInside (int x, int y, int w, int h)
- void show ()
- void hide ()
- bool isVisible ()
- void importFromJson (const json &mem)
- void setFont (TTF_Font ∗f)
- void setText (std::string t)
- void addText (std::string t)
- void addCharacter (char c)
- void removeCharacter ()
- void removeCharacter (int n)
- std::string getText ()
- int getSize ()

7.5.1 Detailed Description

container of button intermediate between button and user input, window

Drawable

Definition at line 19 of file display.hpp.

7.5.2 Constructor & Destructor Documentation

7.5.2.1 Display()

```
Display::Display (
            SDL_Renderer ∗ r,
            SDL_Rect v )
```

Definition at line 4 of file constructor.cpp.

```
                                               : Object(r)
{
    render = r;
    viewport = v;
}
```

7.5.2.2 ∼Display()

```
Display::∼Display ( )
```

Definition at line 3 of file destructor.cpp.

```
{

    return ;
}
```

7.5.3 Member Function Documentation

7.5.3.1 addCharacter()

```
void Object::addCharacter (
            char c )  [inherited]
```

Definition at line 22 of file font.cpp.

```
{
    text += c;
    textToTexture();
}
```

7.5.3.2 addText()

```
void Object::addText (
            std::string t )  [inherited]
```

Definition at line 16 of file font.cpp.

```
{
    text += t;
    textToTexture();
}
```

7.5.3.3 changeToCircle() [1/5]

```
void Object::changeToCircle ( )  [inherited]
```

Definition at line 5 of file shape.cpp.

```
{
    shapeType = SHAPE::CIRCLE;
    radius = std::min(getW(), getH()) / 2;

    center.x = getX() + getW() / 2;
    center.y = getY() + getH() / 2;
    fillCircleByColor();
}
```

7.5.3.4 changeToCircle() [2/5]

```
void Object::changeToCircle (
            int x,
            int y )  [inherited]
```

Definition at line 24 of file shape.cpp.

```
{
    changeToCircle({x, y});
}
```

7.5.3.5 changeToCircle() [3/5]

```
void Object::changeToCircle (
              int x,
              int y,
              int r )  [inherited]
```

Definition at line 37 of file shape.cpp.

```
{
    shapeType = SHAPE::CIRCLE;
    radius = r;
    center.x = x;
    center.y = y;
    fillCircleByColor();
}
```

7.5.3.6 changeToCircle() [4/5]

```
void Object::changeToCircle (
              SDL_Point c )  [inherited]
```

Definition at line 15 of file shape.cpp.

```
{
    shapeType = SHAPE::CIRCLE;
    center = c;
    radius = std::min(getW() - c.x, c.x - getX());
    radius = std::min(radius, std::min(getH() - c.y, c.y - getY()));
    fillCircleByColor();
}
```

7.5.3.7 changeToCircle() [5/5]

```
void Object::changeToCircle (
              SDL_Point c,
              int r )  [inherited]
```

Definition at line 29 of file shape.cpp.

```
{
    shapeType = SHAPE::CIRCLE;
    radius = r;
    center = c;
    fillCircleByColor();
}
```

7.5.3.8 changeToRectangle()

```
void Object::changeToRectangle ( )  [inherited]
```

Definition at line 46 of file shape.cpp.

```
{
    shapeType = SHAPE::RECTANGLE;
    fillRectangleByColor();
}
```

7.5.3.9 coloring() [1/3]

```
void Object::coloring (
            const json & mem )  [inherited]
```

Definition at line 30 of file coloring.cpp.

```
{
    if(mem.contains("r") && mem.contains("g") && mem.contains("b"))
    {
        if(mem.contains("a")) coloring(mem["r"], mem["g"], mem["b"], mem["a"]);
        else coloring(mem["r"], mem["g"], mem["b"], 255);
    }
}
```

7.5.3.10 coloring() [2/3]

```
void Object::coloring (
            int r,
            int g,
            int b,
            int a )  [inherited]
```

Definition at line 8 of file coloring.cpp.

```
{
    if(color == nullptr) color = new SDL_Color;
    color->r = r;
    color->g = g;
    color->b = b;
    color->a = a;

    fillWithColor();
}
```

7.5.3.11 coloring() [3/3]

```
void Object::coloring (
            SDL_Color c )  [inherited]
```

Definition at line 19 of file coloring.cpp.

```
{
    if(color == nullptr) color = new SDL_Color;
    color->r = c.r;
    color->g = c.g;
    color->b = c.b;
    color->a = c.a;

    fillWithColor();
}
```

7.5.3.12 cropping() [1/3]

```
void Object::cropping (
            const json & mem )  [inherited]
```

Definition at line 26 of file cropping.cpp.

```
{
    if(mem.contains("x") && mem.contains("y") && mem.contains("w") && mem.contains("h"))
        cropping(mem["x"], mem["y"], mem["w"], mem["h"]);
}
```

7.5.3.13 cropping() [2/3]

```
void Object::cropping (
              int x,
              int y,
              int w,
              int h )  [inherited]
```

Definition at line 8 of file cropping.cpp.

```
{
    if(crop == nullptr) crop = new SDL_Rect;
    crop->x = x;
    crop->y = y;
    crop->w = w;
    crop->h = h;
}
```

7.5.3.14 cropping() [3/3]

```
void Object::cropping (
              SDL_Rect c )  [inherited]
```

Definition at line 17 of file cropping.cpp.

```
{
    if(crop == nullptr) crop = new SDL_Rect;
    crop->x = c.x;
    crop->y = c.y;
    crop->w = c.w;
    crop->h = c.h;
}
```

7.5.3.15 fillCircleByColor()

```
void Object::fillCircleByColor ( )  [protected], [inherited]
```

Definition at line 91 of file shape.cpp.

```
{
    if(location == nullptr) locating(0, 0, 0, 0);

    if(texture != nullptr) SDL_DestroyTexture(texture);
    texture = nullptr;

    Uint32 rmask, gmask, bmask, amask;
    Uint32 pixelColor;
#if SDL_BYTEORDER == SDL_BIG_ENDIAN
     rmask = 0xff000000;
     gmask = 0x00ff0000;
     bmask = 0x0000ff00;
     amask = 0x000000ff;
     pixelColor = (color->r « 24) | (color->g « 16) | (color->b « 8) | color->a;
#else
     rmask = 0x000000ff;
     gmask = 0x0000ff00;
     bmask = 0x00ff0000;
     amask = 0xff000000;
     pixelColor = (color->a « 24) | (color->b « 16) | (color->g « 8) | color->r;
#endif

     SDL_Surface *surf = SDL_CreateRGBSurface(0, getW(), getH(), 32, rmask, gmask, bmask, amask);
     SDL_SetSurfaceBlendMode(surf, SDL_BLENDMODE_BLEND);

     texture = SDL_CreateTextureFromSurface(render, surf);
     SDL_FreeSurface(surf);

     Uint32 *pixels = new Uint32[getW() * getH()];
    memset(pixels, 0, getW() * getH() * sizeof(Uint32));

    SDL_Point p = {getW() / 2, getH() / 2};
    center = p;

    if(radius > std::min(getW(), getH()) / 2) radius = std::min(getW(), getH()) / 2;

    for(int i = p.x - radius; i <= p.x + radius; i++)
        for(int j = p.y - radius; j <= p.y + radius; j++)
            if((i - p.x) * (i - p.x) + (j - p.y) * (j - p.y) <= radius * radius)
            {
                int index = i * getW() + j;
                if(index < 0 || index >= getW() * getH()) continue;
                pixels[index] = pixelColor;
            }

    SDL_UpdateTexture(texture, nullptr, pixels, getW() * sizeof(Uint32));
    delete[] pixels;
}
```

7.5.3.16 fillRectangleByColor()

`void Object::fillRectangleByColor ( ) [protected], [inherited]`

Definition at line 74 of file shape.cpp.

```
{
    if(location == nullptr) locating(0, 0, 0, 0);

    if(texture != nullptr) SDL_DestroyTexture(texture);
    texture = nullptr;

    SDL_Surface* surf = SDL_CreateRGBSurfaceWithFormat(0, getW(), getH(), 32, SDL_PIXELFORMAT_RGBA32);

    SDL_SetSurfaceBlendMode(surf, SDL_BLENDMODE_BLEND);

    SDL_FillRect(surf, nullptr, SDL_MapRGBA(surf->format, color->r, color->g, color->b, color->a));

    texture = SDL_CreateTextureFromSurface(render, surf);

    SDL_FreeSurface(surf);
}
```

7.5.3.17 fillWithColor()

`void Object::fillWithColor ( ) [protected], [inherited]`

Definition at line 39 of file coloring.cpp.

```
{
    if(shapeType == SHAPE::NONE) return fillRectangleByColor();
    if(shapeType == SHAPE::RECTANGLE) return fillRectangleByColor();
    if(shapeType == SHAPE::CIRCLE) return fillCircleByColor();
}
```

7.5.3.18 fitTheTexture()

`void Object::fitTheTexture ( ) [inherited]`

Definition at line 140 of file locating.cpp.

```
{
    if(texture == nullptr) return;
    SDL_QueryTexture(texture, nullptr, nullptr, &location->w, &location->h);
}
```

7.5.3.19 getColor()

```
const SDL_Color * Object::getColor ( )  [inherited]
```

Definition at line 3 of file coloring.cpp.

```
{
    return color;
}
```

7.5.3.20 getCrop()

```
const SDL_Rect * Object::getCrop ( )  [inherited]
```

Definition at line 3 of file cropping.cpp.

```
{
    return crop;
}
```

7.5.3.21 getH()

```
int Object::getH ( )  [inherited]
```

Definition at line 47 of file locating.cpp.

```
{
    return location->h;
}
```

7.5.3.22 getLocation()

```
const SDL_Rect * Object::getLocation ( )  [inherited]
```

Definition at line 27 of file locating.cpp.

```
{
    return location;
}
```

7.5.3.23 getSize()

```
int Object::getSize ( )  [inherited]
```

Definition at line 68 of file font.cpp.

```
{
    return text.size();
}
```

7.5.3.24 getText()

```
std::string Object::getText ( )  [inherited]
```

Definition at line 63 of file font.cpp.

```
{
    return text;
}
```

7.5.3.25 getW()

```
int Object::getW ( )  [inherited]
```

Definition at line 42 of file locating.cpp.

```
{
    return location->w;
}
```

7.5.3.26 getX()

```
int Object::getX ( )  [inherited]
```

Definition at line 32 of file locating.cpp.

```
{
    return location->x;
}
```

7.5.3.27 getY()

```
int Object::getY ( )  [inherited]
```

Definition at line 37 of file locating.cpp.

```
{
    return location->y;
}
```

7.5.3.28 hide()

```
void Object::hide ( )  [inherited]
```

Definition at line 8 of file visible.cpp.

```
{
    visible = false;
}
```

7.5.3.29 importFromJson() [1/2]

```
void Display::importFromJson ( )  [protected]
```

Definition at line 25 of file constructor.cpp.

```
{
    json* mem = JSON::readFile(PATH::ATB::DISPLAY_ + name + ".json");

    if(mem->contains("background"))
        initBackground((*mem)["background"]);
    if(mem->contains("buttons"))
        initButtons((*mem)["buttons"]);
    delete mem;
}
```

7.5.3.30 importFromJson() [2/2]

```
void Object::importFromJson (
              const json & mem )  [inherited]
```

Definition at line 21 of file constructor.cpp.

```
{
    if(mem.contains("location"))
        locating(mem["location"]);

    if(mem.contains("crop"))
        cropping(mem["crop"]);

    if(mem.contains("color"))
        coloring(mem["color"]);

    if(mem.contains("shape"))
        setShape(mem["shape"]);

    if(mem.contains("visible"))
        visible = mem["visible"];

    if(mem.contains("image"))
        textureFromFile(PATH::ASSETS::GRAPHICS_ + mem["image"].get<std::string>());
    return ;
}
```

7.5.3.31 initBackground()

```
void Display::initBackground (
              const json & mem )  [protected]
```

Definition at line 10 of file constructor.cpp.

```
{
    Object::importFromJson(mem);
}
```

7.5.3.32 initButtons()

```
void Display::initButtons (
             const json & mem )  [protected]
```

Definition at line 15 of file constructor.cpp.

```
{
    for(auto& i : mem)
    {
        Button* b = new Button(render);
        b->linking(i["name"].get<std::string>());
        buts.push_back(b);
    }
}
```

7.5.3.33 isButtonReceiveEvent()

```
bool Display::isButtonReceiveEvent (
             SDL_Event & e )  [protected]
```

Definition at line 3 of file event.cpp.

```
{
    for(auto& but : buts)
        if(but->isReceiveEvent(e)) return true;
    return false;
}
```

7.5.3.34 isLieInside() [1/4]

```
bool Object::isLieInside (
             int x,
             int y )  [inherited]
```

Definition at line 3 of file locating.cpp.

```
{
    if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
    return (x >= location->x && x < location->x + location->w && y >= location->y && y < location->y +
     location->h);
}
```

7.5.3.35 isLieInside() [2/4]

```
bool Object::isLieInside (
             int x,
             int y,
             int w,
             int h )  [inherited]
```

Definition at line 21 of file locating.cpp.

```
{
    if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
    return (x >= location->x && x + w <= location->x + location->w && y >= location->y && y + h <=
    location->y + location->h);
}
```

7.5.3.36 isLieInside() [3/4]

```
bool Object::isLieInside (
            SDL_Point p )  [inherited]
```

Definition at line 9 of file locating.cpp.

```
{
    if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
    return (p.x >= location->x && p.x < location->x + location->w && p.y >= location->y && p.y <
    location->y + location->h);
}
```

7.5.3.37 isLieInside() [4/4]

```
bool Object::isLieInside (
            SDL_Rect r )  [inherited]
```

Definition at line 15 of file locating.cpp.

```
{
    if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
    return (r.x >= location->x && r.x + r.w <= location->x + location->w && r.y >= location->y && r.y +
    r.h <= location->y + location->h);
}
```

7.5.3.38 isReceiveEvent()

```
bool Display::isReceiveEvent (
            SDL_Event & e )
```

Definition at line 10 of file event.cpp.

```
{

    switch(e.type)
    {
        case SDL_QUIT:
            return false;
            break;
        default:
            if(isButtonReceiveEvent(e)) return true;
            return false;
            break;
    }
}
```

7.5.3.39 isVisible()

```
bool Object::isVisible ( )  [inherited]
```

Definition at line 13 of file visible.cpp.

```
{
    return visible;
}
```

7.5.3.40 linking()

```
void Display::linking (
             std::string n )
```

Definition at line 37 of file constructor.cpp.

```
{
    name = n;
    importFromJson();
}
```

7.5.3.41 locating() [1/3]

```
void Object::locating (
             const json & mem )  [virtual], [inherited]
```

Definition at line 70 of file locating.cpp.

```
{
    if(mem.contains("x") && mem.contains("y") && mem.contains("w") && mem.contains("h"))
        locating(mem["x"], mem["y"], mem["w"], mem["h"]);
}
```

7.5.3.42 locating() [2/3]

```
void Object::locating (
             int x,
             int y,
             int w,
             int h )  [virtual], [inherited]
```

Reimplemented in Sprite.

Definition at line 52 of file locating.cpp.

```
{
    if(location == nullptr) location = new SDL_Rect;
    location->x = x;
    location->y = y;
    location->w = w;
    location->h = h;
}
```

7.5.3.43 locating() [3/3]

```
void Object::locating (
             SDL_Rect l )  [virtual], [inherited]
```

Reimplemented in Sprite.

Definition at line 61 of file locating.cpp.

```
{
    if(location == nullptr) location = new SDL_Rect;
    location->x = l.x;
    location->y = l.y;
    location->w = l.w;
    location->h = l.h;
}
```

7.5.3.44 locatingH()

```
void Object::locatingH (
            int h )  [virtual], [inherited]
```

Definition at line 94 of file locating.cpp.

```
{
    if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
    location->h = h;
}
```

7.5.3.45 locatingW()

```
void Object::locatingW (
            int w )  [virtual], [inherited]
```

Definition at line 88 of file locating.cpp.

```
{
    if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
    location->w = w;
}
```

7.5.3.46 locatingX()

```
void Object::locatingX (
            int x )  [virtual], [inherited]
```

Reimplemented in Sprite.

Definition at line 76 of file locating.cpp.

```
{
    if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
    location->x = x;
}
```

7.5.3.47 locatingY()

```
void Object::locatingY (
            int y )  [virtual], [inherited]
```

Reimplemented in Sprite.

Definition at line 82 of file locating.cpp.

```
{
    if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
    location->y = y;
}
```

7.5.3.48 moveX()

```
void Object::moveX (
             int delta )  [virtual], [inherited]
```

Reimplemented in Sprite.

Definition at line 100 of file locating.cpp.

```
{
    if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
    location->x += delta;
}
```

7.5.3.49 moveY()

```
void Object::moveY (
             int delta )  [virtual], [inherited]
```

Reimplemented in Sprite.

Definition at line 106 of file locating.cpp.

```
{
    if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
    location->y += delta;
}
```

7.5.3.50 noCropping()

```
void Object::noCropping ( )  [inherited]
```

Definition at line 32 of file cropping.cpp.

```
{
    if(crop != nullptr) delete crop;
    crop = nullptr;
}
```

7.5.3.51 react()

```
Button * Display::react (
             SDL_Event & e )
```

Definition at line 25 of file event.cpp.

```
{
    switch(e.type)
    {
        case SDL_MOUSEMOTION:
            for(auto& but : buts)
                but->isHover(e.motion.x, e.motion.y);
            return nullptr;
            break;
        case SDL_MOUSEBUTTONDOWN:
            for(auto& but :buts)
                if(but->isClicked(e.motion.x, e.motion.y)) return but;
            return nullptr;
            break;
        default:
            return nullptr;
            break;
    }
}
```

7.5.3.52 removeCharacter() [1/2]

```
void Object::removeCharacter ( )  [inherited]
```

Definition at line 28 of file font.cpp.

```
{
    if (text.size() > 0)
        text.pop_back();
    textToTexture();
}
```

7.5.3.53 removeCharacter() [2/2]

```
void Object::removeCharacter (
              int n )  [inherited]
```

Definition at line 35 of file font.cpp.

```
{
    if(n == 0) return ;
    if(text.size() <= n) text.clear();
    else text.erase(text.end() - n, text.end());
    textToTexture();
}
```

7.5.3.54 rendering()

```
void Display::rendering ( )  [virtual]
```

Reimplemented from Object.

Definition at line 4 of file rendering.cpp.

```
{
    SDL_RenderSetViewport(render, &viewport);
    Object::rendering();
    for(auto& i : buts)
        i->rendering();
}
```

7.5.3.55 setFont()

```
void Object::setFont (
              TTF_Font * f )  [inherited]
```

Definition at line 4 of file font.cpp.

```
{
    font = f;
    textToTexture();
}
```

7.5.3.56 setShape()

```
void Object::setShape (
             const json & mem )  [inherited]
```

Definition at line 52 of file shape.cpp.

```
{
    if(mem["type"].get<std::string>() == "CIRCLE")
    {
        if(mem.contains("center"))
        {
            if(mem.contains("radius"))
                changeToCircle(mem["center"]["x"], mem["center"]["y"], mem["radius"]);
            else changeToCircle(mem["center"]["x"], mem["center"]["y"]);
        }else changeToCircle();

        return ;
    }

    if(mem["type"].get<std::string>() == "NONE" || mem["type"].get<std::string>() == "RECTANGLE")
    {
        changeToRectangle();
        return ;
    }

}
```

7.5.3.57 setText()

```
void Object::setText (
             std::string t )  [inherited]
```

Definition at line 10 of file font.cpp.

```
{
    text = t;
    textToTexture();
}
```

7.5.3.58 show()

```
void Object::show ( )  [inherited]
```

Definition at line 3 of file visible.cpp.

```
{
    visible = true;
}
```

7.5.3.59 textToTexture()

```
void Object::textToTexture ( )  [protected], [inherited]
```

Definition at line 43 of file font.cpp.

```
{
    if(font == nullptr) return ;
    if(color == nullptr) return ;
    if(render == nullptr) return ;
    if(texture != nullptr)
    {
        SDL_DestroyTexture(texture);
    }
    texture = nullptr;

    SDL_Surface* surface = TTF_RenderText_Blended(font, text.c_str(), *color);

    if(surface == nullptr) return ;

    texture = SDL_CreateTextureFromSurface(render, surface);
    SDL_FreeSurface(surface);
    fitTheTexture();
}
```

7.5.3.60 textureFromFile()

```
void Object::textureFromFile (
              std::string dir )  [inherited]
```

Definition at line 4 of file external_storage.cpp.

```
{
    SDL_Surface *surface = IMG_Load(dir.c_str());

    texture = SDL_CreateTextureFromSurface(render, surface);
    SDL_FreeSurface(surface);
}
```

7.5.3.61 zoom()

```
void Object::zoom (
              double delta )  [virtual], [inherited]
```

Definition at line 123 of file locating.cpp.

```
{
    if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
    location->w *= delta;
    location->h *= delta;
}
```

7.5.3.62 zoomH()

```
void Object::zoomH (
              int delta )  [virtual], [inherited]
```

Definition at line 118 of file locating.cpp.

```
{
    if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
    location->h += delta;
}
```

7.5.3.63 zoomInMiddle()

```
void Object::zoomInMiddle (
             double delta )  [virtual], [inherited]
```

Definition at line 130 of file locating.cpp.

```
{
    if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
    SDL_Point center = {location->x + location->w / 2, location->y + location->h / 2};
    location->w *= delta;
    location->h *= delta;
    location->x = center.x - location->w / 2;
    location->y = center.y - location->h / 2;
}
```

7.5.3.64 zoomW()

```
void Object::zoomW (
             int delta )  [virtual], [inherited]
```

Definition at line 112 of file locating.cpp.

```
{
    if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
    location->w += delta;
}
```

The documentation for this class was generated from the following files:

- include/display.hpp
- src/display/constructor.cpp
- src/display/destructor.cpp
- src/display/event.cpp
- src/display/rendering.cpp

7.6 distanceHeap Struct Reference

Public Member Functions

- distanceHeap (Graph ∗g)
- ∼distanceHeap ()
- bool swapable (int i, int j)
- void swap (int i, int j)
- void insert (int v)
- int pop ()
- bool empty ()

Public Attributes

- Graph ∗ g
- std::vector< int > value

7.6.1 Detailed Description

Definition at line 3 of file dijkstra.cpp.

7.6.2 Constructor & Destructor Documentation

7.6.2.1 distanceHeap()

```
distanceHeap::distanceHeap (
              Graph * g )  [inline]
```

Definition at line 8 of file dijkstra.cpp.

```
    {
        this->g = g;
     }
```

7.6.2.2 ∼distanceHeap()

```
distanceHeap::∼distanceHeap ( )  [inline]
```

Definition at line 12 of file dijkstra.cpp.

```
     {
        value.clear();
        g = nullptr;
     }
```

7.6.3 Member Function Documentation

7.6.3.1 empty()

```
bool distanceHeap::empty ( )  [inline]
```

Definition at line 71 of file dijkstra.cpp.

```
{ return value.empty(); }
```

7.6.3.2 insert()

```
void distanceHeap::insert (
            int v )  [inline]
```

Definition at line 28 of file dijkstra.cpp.

```
    {
        value.push_back(v);
        int index = value.size() - 1;
        do
        {
            int parent = (index - 1) / 2;
            if(swapable(parent, index))
            {
                swap(parent, index);
            }
            index = parent;
        }while(index != 0);
    }
```

7.6.3.3 pop()

```
int distanceHeap::pop ( )  [inline]
```

Definition at line 42 of file dijkstra.cpp.

```
    {
        if(value.size() == 0) return 2e9;
        int result = value[0];

        value[0] = value[value.size() - 1];
        value.pop_back();

        int index = 0;

        while(index < value.size())
        {
            int left = index * 2 + 1;
            int right = index * 2 + 2;
            int next = index;

            if(left < value.size() && swapable(next, left))
                next = left;
            if(right < value.size() && swapable(next, right))
                next = right;

            if(next == index) break;

            swap(index, next);
            index = next;
        }

        return result;
    }
```

7.6.3.4 swap()

```
void distanceHeap::swap (
            int i,
            int j )  [inline]
```

Definition at line 24 of file dijkstra.cpp.

```
    {
        std::swap(value[i], value[j]);
    }
```

7.6.3.5 swapable()

```
bool distanceHeap::swapable (
              int i,
              int j )  [inline]
```

Definition at line 18 of file dijkstra.cpp.

```
{
    if(g->distance[value[i]] == -1) return true;
    if(g->distance[value[j]] == -1) return false;
    return g->distance[value[i]] > g->distance[value[j]];
}
```

7.6.4 Member Data Documentation

7.6.4.1 g

```
Graph* distanceHeap::g
```

Definition at line 5 of file dijkstra.cpp.

7.6.4.2 value

```
std::vector<int> distanceHeap::value
```

Definition at line 6 of file dijkstra.cpp.

The documentation for this struct was generated from the following file:

- src/graph/operator/dijkstra.cpp

7.7 DSU Struct Reference

Public Member Functions

- DSU (int n)
- ∼DSU ()
- void unionEdge (Graph::Edge ∗e)
- void unionEdge (Graph::Node ∗u, Graph::Node ∗v)
- void unionEdge (int u, int v)
- int find (Graph::Node ∗v)
- int find (int v)
- bool isUnionized (Graph::Node ∗u, Graph::Node ∗v)
- bool isUnionized (Graph::Edge ∗e)
- bool isUnionized (int u, int v)

Public Attributes

- std::vector< int > parent

7.7.1 Detailed Description

Definition at line 4 of file mst.cpp.

7.7.2 Constructor & Destructor Documentation

7.7.2.1 DSU()

```
DSU::DSU (
             int n )  [inline]
```

Definition at line 7 of file mst.cpp.

```
{
    parent.resize(n + 1);
     for(int i = 0; i < n; i++)
         parent[i] = -1;
 }
```

7.7.2.2 ∼DSU()

```
DSU::∼DSU ( )  [inline]
```

Definition at line 13 of file mst.cpp.

```
{
    parent.clear();
}
```

7.7.3 Member Function Documentation

7.7.3.1 find() [1/2]

```
int DSU::find (
             Graph::Node * v )  [inline]
```

Definition at line 36 of file mst.cpp.

```
{
    return find(v->value);
}
```

7.7.3.2 find() [2/2]

```
int DSU::find (
             int v )  [inline]
```

Definition at line 40 of file mst.cpp.

```
{
    if(parent[v] < 0)
        return v;
    return parent[v] = find(parent[v]);
}
```

7.7.3.3 isUnionized() [1/3]

```
bool DSU::isUnionized (
             Graph::Edge * e )  [inline]
```

Definition at line 51 of file mst.cpp.

```
{
    return isUnionized(e->u, e->v);
}
```

7.7.3.4 isUnionized() [2/3]

```
bool DSU::isUnionized (
             Graph::Node * u,
             Graph::Node * v )  [inline]
```

Definition at line 47 of file mst.cpp.

```
{
    return isUnionized(u->value, v->value);
}
```

7.7.3.5 isUnionized() [3/3]

```
bool DSU::isUnionized (
             int u,
             int v )  [inline]
```

Definition at line 55 of file mst.cpp.

```
{
    return find(u) == find(v);
}
```

7.7.3.6 unionEdge() [1/3]

```
void DSU::unionEdge (
            Graph::Edge * e )  [inline]
```

Definition at line 18 of file mst.cpp.

```
    {
        unionEdge(e->u, e->v);
    }
```

7.7.3.7 unionEdge() [2/3]

```
void DSU::unionEdge (
            Graph::Node * u,
            Graph::Node * v )  [inline]
```

Definition at line 22 of file mst.cpp.

```
    {
        unionEdge(u->value, v->value);
    }
```

7.7.3.8 unionEdge() [3/3]

```
void DSU::unionEdge (
            int u,
            int v )  [inline]
```

Definition at line 26 of file mst.cpp.

```
    {
        int a = find(u);
        int b = find(v);
        if(a == b) return ;

        parent[a] += parent[b];
        parent[b] = a;
    }
```

7.7.4 Member Data Documentation

7.7.4.1 parent

```
std::vector<int> DSU::parent
```

Definition at line 6 of file mst.cpp.

The documentation for this struct was generated from the following file:

- src/graph/operator/mst.cpp

7.8 Graph Class Reference

Graph class.

```
#include <graph.hpp>
```

Public Member Functions

- Graph (SDL_Renderer ∗r, std::mutex &m, TTF_Font ∗f, SDL_Rect v, int capacity)
- ∼Graph ()
- void Dijkstra (int start, int end)
- void MST ()
- void SCC ()
- void init (std::vector< std::vector< int > > value)
- bool isReceiveEvent (SDL_Event &e)
- Button ∗ react (SDL_Event &e)
- void rendering ()
- void setting (SDL_Color c1, SDL_Color c2, SDL_Color c3, SDL_Color c4)

Protected Member Functions

- void unionEdges ()
- void Tarjan (Node ∗u)
- void repair ()
- void renderEdge (Edge ∗edge)
- void waitForStep ()

Friends

- struct distanceHeap
- struct DSU

7.8.1 Detailed Description

Graph class.

Drawable graph.

Definition at line 23 of file graph.hpp.

7.8.2 Constructor & Destructor Documentation

7.8.2.1 Graph()

```
Graph::Graph (
              SDL_Renderer * r,
              std::mutex & m,
              TTF_Font * f,
              SDL_Rect v,
              int capacity )
```

Definition at line 33 of file constructor.cpp.

```
                                                                                   : ds_mutex(m)
{
    render = r;
    font = f;
    viewport = v;
    capacity = capacity;
    edgesColor = {255, 255, 255, 255};
    shiftX = 20;
    shiftY = 20;

    nodirect = false;

    nodeColor = {20, 75, 185, 255};
    fontColor = {255, 255, 255, 255};

    stepWait = 600;

    isMoving = false;
    chosenNode = nullptr;
    lastMousePressed = {0, 0};


}
```

7.8.2.2 ∼Graph()

```
Graph::∼Graph ( )
```

Definition at line 19 of file destructor.cpp.

```
{
    for(auto &i : nodes)
        delete i;
    nodes.clear();

    for(auto &i : edges)
        delete i;
    edges.clear();
}
```

7.8.3 Member Function Documentation

7.8.3.1 Dijkstra()

```
void Graph::Dijkstra (
              int start,
              int end )
```

Definition at line 74 of file dijkstra.cpp.

```
{
    repair();
    distance.clear();
    distance.resize(nodes.size() + 1, -1);
    distanceHeap heap(this);
    distance[start] = 0;

    heap.insert(start);

    for(auto i : nodes) i->sprite->coloring(SDL_Color{50, 50, 50, 255});

    while(!heap.empty())
    {
        int u = heap.pop();
        nodes[u]->sprite->coloring(SDL_Color{0, 125, 0, 255});
        waitForStep();
        for(auto e : nodes[u]->edges)
        {
            int v = e->v->value;


            if(distance[v] == -1 || distance[v] > distance[u] + e->weight)
            {
                nodes[v]->sprite->coloring(SDL_Color{255, 255, 0, 255});
                waitForStep();
                nodes[v]->sprite->coloring(SDL_Color{0, 255, 255, 255});
                waitForStep();
                distance[v] = distance[u] + e->weight;
                heap.insert(v);
            }
        }
        nodes[u]->sprite->coloring(SDL_Color{0, 255, 0, 255});
    }
}
```

7.8.3.2 init()

```
void Graph::init (
              std::vector< std::vector< int > > value )
```

Definition at line 56 of file init.cpp.

```
{
    capacity = g.size();

    for(auto i : nodes)
        delete i;
    for(auto i : edges)
        delete i;
    nodes.clear();
    edges.clear();

    for(int i = 0; i < capacity; i++)
    {
        int x = 10;
        int y = 10;
        int rep = 20;
        do
        {
            x = RANDOM::getInt(10, 720);
            y = RANDOM::getInt(10, 520);
            rep--;
        }while(rep != 0 && isCollision(x, y));
        Sprite* spr = new Sprite(render);
        spr->setFont(font);
        spr->linking("AVL/node");
        spr->locatingX(x);
        spr->locatingY(y);
        spr->setText(NUMBER::intToString(i));
        spr->setFontColor(fontColor);
        spr->coloring(nodeColor);
        spr->aligning(HORIZONTAL_ALIGN::CENTER, VERTICAL_ALIGN::CENTER);

        Node* node = new Node(i, spr);

        nodes.push_back(node);
    }

    for(int i = 0; i < capacity; i++)
    {
        for(int j = 0; j < capacity && j < g[i].size(); j++)
        {
            if(g[i][j] != 0)
            {
                Sprite* spr = new Sprite(render);
                 spr->setFont(font);
                 spr->linking("graph/weight");

                 spr->locatingX((nodes[i]->sprite->getX() + nodes[j]->sprite->getX()) / 2);
                 spr->locatingY((nodes[i]->sprite->getY() + nodes[j]->sprite->getY()) / 2);

                 spr->setText(NUMBER::intToString(g[i][j]));
                 spr->aligning(HORIZONTAL_ALIGN::CENTER, VERTICAL_ALIGN::CENTER);
                 Edge* edge = new Edge(nodes[i], nodes[j], g[i][j], spr);
                 edges.push_back(edge);
                 nodes[i]->addEdge(edge);
             }
         }
     }
}
```

7.8.3.3 isReceiveEvent()

```
bool Graph::isReceiveEvent (
               SDL_Event & e )
```

Definition at line 4 of file event.cpp.

```
{
     std::lock_guard<std::mutex> lk(animate_mutex);
     switch(e.type)
     {
         case SDL_MOUSEBUTTONDOWN:
              if(e.motion.x < viewport.x || viewport.x + viewport.w < e.motion.x) return false;
              if(e.motion.y < viewport.y || viewport.y + viewport.h < e.motion.y) return false;
              if(e.button.button == SDL_BUTTON_LEFT) return false;
              for(auto i : nodes)
                  if(i->sprite->isLieInside(e.motion.x, e.motion.y))
                      return true;
              return false;
              break;
          case SDL_MOUSEMOTION:
              if(isMoving) return true;
              return false;
              break;
          default:
              return false;
              break;
      }
}
```

7.8.3.4 MST()

```
void Graph::MST ( )
```

Definition at line 62 of file mst.cpp.

```
{
      sortedEdges.clear();

      for(auto i : edges)
          sortedEdges.push_back(i);

      std::sort(
              sortedEdges.begin(),
              sortedEdges.end(),
              [&](Edge* u, Edge* v){
                  if(u == nullptr) return false;
                  if(v == nullptr) return true;
                  return u->weight < v->weight;
              }
      );

      unionEdges();
}
```

7.8.3.5 react()

```
Button * Graph::react (
             SDL_Event & e )
```

Definition at line 29 of file event.cpp.

```
{
    std::lock_guard<std::mutex> lk(animate_mutex);
    switch(e.type)
    {
        case SDL_MOUSEBUTTONDOWN:
            if(isMoving)
            {
                isMoving = false;
                chosenNode = nullptr;
            }else
            {
                isMoving = true;

                lastMousePressed.x = e.motion.x;
                lastMousePressed.y = e.motion.y;
                for(auto i : nodes)
                    if(i->sprite->isLieInside(e.motion.x, e.motion.y))
                    {
                        chosenNode = i;
                        break;
                    }
            }
            return nullptr;
            break;
        case SDL_MOUSEMOTION:
            {
                if(!isMoving) return nullptr;
                int dx = e.motion.x - lastMousePressed.x;
                int dy = e.motion.y - lastMousePressed.y;
                lastMousePressed.x = e.motion.x;
                lastMousePressed.y = e.motion.y;
                if(chosenNode == nullptr) return nullptr;
                chosenNode->sprite->moveX(dx);
                chosenNode->sprite->moveY(dy);
                return nullptr;
                break;
            }
defaut:
            return nullptr;
            break;
    }
    return nullptr;
}
```

7.8.3.6 renderEdge()

```
void Graph::renderEdge (
             Edge * edge )   [protected]
```

Definition at line 8 of file rendering.cpp.

```
{
    if(e->mark == 3)
        SDL_SetRenderDrawColor(render, 50, 50, 50, 255);
    else if(e->mark == 2)
        SDL_SetRenderDrawColor(render, 255, 0, 0, 255);
    else if(e->mark == 1)
        SDL_SetRenderDrawColor(render, 0, 255, 0, 255);
    else
        SDL_SetRenderDrawColor(render, edgesColor.r, edgesColor.g, edgesColor.b, edgesColor.a);
    const SDL_Rect* srcloc = (e->u->sprite->getLocation());
    const SDL_Rect* dstloc = (e->v->sprite->getLocation());

    SDL_Point src;
    src.x = srcloc->x + srcloc->w/2;
    src.y = srcloc->y + srcloc->h/2;


    SDL_Point dst;
    dst.x = dstloc->x + dstloc->w/2;
    dst.y = dstloc->y + dstloc->h/2;

    int sign = 1;
    if(dst.x < src.x)
        sign = -1;
    e->sprite->locatingX((src.x + dst.x) / 2 + sign * 10);

    sign = 1;
    if(dst.y < src.y)
        sign = -1;
    e->sprite->locatingY((src.y + dst.y) / 2 + sign * 10);

    for (int i = -1; i <= 1; i++)
    {
        for(int j = -1; j <= 1; j++)
        {
            SDL_RenderDrawLine(render, src.x + i, src.y + j, dst.x + i, dst.y + j);
        }
    }
    if(nodirect) return ;
    Point v;
    v.x = dst.x - src.x;
    v.y = dst.y - src.y;
    double len = sqrt(v.x * v.x + v.y * v.y);
    v.x /= len;
    v.y /= len;


    Point u = {v.y, -v.x};

    SDL_Point p1;
    p1.x = dst.x - v.x * 40 + u.x * 10;
    p1.y = dst.y - v.y * 40 + u.y * 10;

    SDL_Point p2;
    p2.x = dst.x - v.x * 40 - u.x * 10;
    p2.y = dst.y - v.y * 40 - u.y * 10;

    SDL_Point p3;
    p3.x = dst.x - v.x * 28;
    p3.y = dst.y - v.y * 28;

    for(int i = -1; i <= 1; i++)
        for(int j = -1; j <= 1; j++)
        {
            SDL_RenderDrawLine(render, p1.x + i, p1.y + j, p3.x + i, p3.y + j);
            SDL_RenderDrawLine(render, p2.x + i, p2.y + j, p3.x + i, p3.y + j);
        }
}
```

7.8.3.7 rendering()

`void Graph::rendering ( )`

Definition at line 78 of file rendering.cpp.

```
{
    for(auto i : edges)
    {
        renderEdge(i);
        if(i->sprite != nullptr) i->sprite->rendering();
    }
    for(auto i : nodes)
    {
        i->sprite->rendering();
    }
}
```

7.8.3.8 repair()

`void Graph::repair ( ) [protected]`

Definition at line 3 of file repair.cpp.

```
{
    for(auto i : edges)
    {
        i->mark = 0;
    }
    setting(bgColor, nodeColor, fontColor, edgesColor);
     nodirect = false;
}
```

7.8.3.9 SCC()

```
void Graph::SCC ( )
```

Definition at line 37 of file scc.cpp.

```
{
    low.clear();
    order.clear();
    components.clear();
    repair();
    state = 0;

    low.resize(nodes.size() + 1);
    order.resize(nodes.size() + 1);

    for(auto i : nodes)
    {
        if(order[i->value] == 0)
            Tarjan(i);
    }

    for(auto i : components)
    {
        SDL_Color c;
        c.r = RANDOM::getInt(0, 255);
        c.g = RANDOM::getInt(0, 255);
        c.b = RANDOM::getInt(0, 255);
        c.a = 255;
        for(auto j : i)
        {
            j->sprite->coloring(c);
        }
    }
}
```

7.8.3.10 setting()

```
void Graph::setting (
            SDL_Color c1,
            SDL_Color c2,
            SDL_Color c3,
            SDL_Color c4 )
```

Definition at line 57 of file constructor.cpp.

```
{
    bgColor = c1;
    nodeColor = c2;
    fontColor = c3;
    edgesColor = c4;
    for(auto i : nodes)
    {
        i->sprite->setFontColor(fontColor);
        i->sprite->coloring(nodeColor);
    }
}
```

7.8.3.11 Tarjan()

```
void Graph::Tarjan (
              Node * u )   [protected]
```

Definition at line 4 of file scc.cpp.

```
{
     order[u->value] = low[u->value] = ++state;
     buffer.push(u);

     for(auto i : u->edges)
      {
          if(order[i->v->value] == 0)
          {
              Tarjan(i->v);
              low[u->value] = std::min(low[u->value], low[i->v->value]);
          }
          else if(order[i->v->value] != 0 && low[i->v->value] != 0)
          {
              low[u->value] = std::min(low[u->value], order[i->v->value]);
          }
      }

      if(low[u->value] == order[u->value])
      {
          std::vector<Node*> component;
          Node* v;
          do
          {
              v = buffer.top();
              buffer.pop();
              component.push_back(v);
              low[v->value] = 0;
          } while(v != u);
          components.push_back(component);
      }
 }
```

7.8.3.12 unionEdges()

```
void Graph::unionEdges ( )   [protected]
```

Definition at line 82 of file mst.cpp.

```
{
     repair();
     DSU dsu(nodes.size());
     nodirect = true;
     for(auto i : sortedEdges)
     {
         i->mark = 3;
         waitForStep();
         if(!dsu.isUnionized(i))
         {
             i->mark = 1;
             waitForStep();
             dsu.unionEdge(i);
         }else
         {
             i->mark = 2;
             waitForStep();
          }
     }
 }
```

7.8.3.13 waitForStep()

```
void Graph::waitForStep ( )   [protected]
```

Definition at line 75 of file event.cpp.

```
{
     ds_mutex.unlock();
     std::this_thread::sleep_for(std::chrono::milliseconds(stepWait));
     ds_mutex.lock();
 }
```

7.8.4 Friends And Related Function Documentation

7.8.4.1 distanceHeap

```
friend struct distanceHeap  [friend]
```

Definition at line 78 of file graph.hpp.

7.8.4.2 DSU

```
friend struct DSU  [friend]
```

Definition at line 86 of file graph.hpp.

The documentation for this class was generated from the following files:

- include/data_structures/graph.hpp
- src/graph/constructor.cpp
- src/graph/destructor.cpp
- src/graph/event.cpp
- src/graph/operator/dijkstra.cpp
- src/graph/operator/init.cpp
- src/graph/operator/mst.cpp
- src/graph/operator/repair.cpp
- src/graph/operator/scc.cpp
- src/graph/rendering.cpp

7.9 HashTable Class Reference

HashTable class.

```
#include <hash_table.hpp>
```

Public Member Functions

- HashTable (SDL_Renderer ∗render, std::mutex &m, TTF_Font ∗font, SDL_Rect v, int cap)
- ∼HashTable ()
- void init (std::vector< int > v, int KEY)
- void insert (int key)
- void remove (int key)
- bool search (int key)
- void setEdgesColor (SDL_Color c)
- void setNodeColor (SDL_Color bg, SDL_Color fg)
- void goOff ()
- void goOn ()
- void goNext ()
- void goBack ()
- void speedUp ()
- void slowDown ()
- bool isReceiveEvent (SDL_Event &e)
- Button ∗ react (SDL_Event &e)
- void closeScript ()
- void rendering ()
- void setting (SDL_Color c1, SDL_Color c2, SDL_Color c3, SDL_Color c4)

Protected Member Functions

- Node ∗ insert (Node ∗root, int k)
- Node ∗ remove (Node ∗root, int k)
- bool search (Node ∗root, int k)
- int locating (Node ∗node, int shiftDown, int shiftRigh)
- int locating (Head ∗∗table, int shiftDown, int shiftRight)
- void renderEdges (Node ∗src, Node ∗dst)
- void defaultSetting ()
- void drawEdge (Node ∗src, Node ∗dst)
- void waitForStep ()
- void highlight (std::vector< int > l)
- void unhighlight (std::vector< int > l)

7.9.1 Detailed Description

HashTable class.

Drawable HashTable.

Definition at line 23 of file hash_table.hpp.

7.9.2 Constructor & Destructor Documentation

7.9.2.1 HashTable()

```
HashTable::HashTable (
               SDL_Renderer * render,
               std::mutex & m,
               TTF_Font * font,
               SDL_Rect v,
               int cap )
```

Definition at line 18 of file constructor.cpp.

```
                                                                                      : ds_mutex(m)
{
    render = r;
    font = f;
    viewport = v;
    capacity = cap;

    table = nullptr;

    currentSize = 0;

    isQueue = false;
    isPause = false;

    edgesColor = {255, 255, 255, 255};
    nodeColor = {20, 85, 185, 255};
    fontColor = {255, 255, 255, 255};

    shiftX = 20;
    shiftY = 20;
    distanceX = 100;
    distanceY = 70;
    isMoving = false;

    stepWait = 600;

    std::string fontpath = PATH::ASSETS::FONTS_ + "nimbus-sans-l/regular.otf";
    scriptFont = TTF_OpenFont(fontpath.c_str(), 18);

    currentScript = nullptr;
    Script* insert = new Script(render, scriptFont);
    insert->linking("hash_table/insert");
    scripts[DATA_STRUCTURES_OPERATOR::INSERT] = insert;

    Script* remove = new Script(render, scriptFont);
    remove->linking("hash_table/remove");
    scripts[DATA_STRUCTURES_OPERATOR::DELETE] = remove;

    Script* search = new Script(render, scriptFont);
    search->linking("hash_table/search");
    scripts[DATA_STRUCTURES_OPERATOR::SEARCH] = search;

    Script* init = new Script(render, scriptFont);
    init->linking("hash_table/init");
    scripts[DATA_STRUCTURES_OPERATOR::INIT] = init;

}
```

7.9.2.2 ∼HashTable()

```
HashTable::∼HashTable ( )
```

Definition at line 15 of file destructor.cpp.

```
{
    if(table != nullptr)
    {
        for(int i = 0; i < HASH_KEY; i++)
        {
            if(table[i] != nullptr) delete table[i];
        }
        delete [] table;
    }
    for(auto i : scripts)
        delete i.second;
    TTF_CloseFont(scriptFont);
}
```

7.9.3 Member Function Documentation

7.9.3.1 closeScript()

```
void HashTable::closeScript ( )
```

Definition at line 53 of file event.cpp.

```
{
    currentScript = nullptr;
}
```

7.9.3.2 defaultSetting()

```
void HashTable::defaultSetting ( )  [protected]
```

Definition at line 100 of file constructor.cpp.

```
{
    HASH_KEY = 19;
    table = new Head*[HASH_KEY];
    for(int i = 0; i < HASH_KEY; i++)
    {
        Sprite* spr = new Sprite(render);
        spr->setFont(font);
        spr->setFontColor(fontColor);
        spr->coloring(nodeColor);
        spr->linking("hash-table/head");
        table[i] = new Head(spr);
    }
    locating(table, 0, 0);
}
```

7.9.3.3 drawEdge()

```
void HashTable::drawEdge (
            Node * src,
            Node * dst )  [protected]
```

Definition at line 3 of file rendering.cpp.

```
{
    SDL_Rect srcRect = *src->sprite->getLocation();
    SDL_Rect dstRect = *dst->sprite->getLocation();

    SDL_SetRenderDrawColor(render, edgesColor.r, edgesColor.g, edgesColor.b, edgesColor.a);
    for(int j = -1; j <= 1; j++)
     {
         SDL_RenderDrawLine(render,srcRect.x + srcRect.w, srcRect.y + srcRect.h / 2 + j, dstRect.x,
     dstRect.y + dstRect.h / 2 + j);
     }

     for(int j =  -3; j <= 3; j++)
     {
         SDL_RenderDrawLine(render, dstRect.x - 8, dstRect.y - 5 + j + dstRect.h / 2, dstRect.x, dstRect.y
     + dstRect.h / 2);
         SDL_RenderDrawLine(render, dstRect.x - 8, dstRect.y + 5 + j + dstRect.h / 2, dstRect.x, dstRect.y
     + dstRect.h / 2);
         SDL_RenderDrawLine(render, dstRect.x - 8, dstRect.y - 5 + j + dstRect.h / 2, dstRect.x - 8,
     dstRect.y + 5 + j + dstRect.h / 2);
     }
}
```

7.9.3.4 goBack()

```
void HashTable::goBack ( )
```

7.9.3.5 goNext()

```
void HashTable::goNext ( )
```

7.9.3.6 goOff()

```
void HashTable::goOff ( )
```

7.9.3.7 goOn()

```
void HashTable::goOn ( )
```

7.9.3.8 highlight()

```
void HashTable::highlight (
              std::vector< int > l ) [protected]
```

Definition at line 23 of file step.cpp.

```
{
    if(isAnimate)
    {
        animate_mutex.lock();
        for(int i = 0; i < l.size(); i++)
        {
            currentScript->highlight(l[i]);
        }
        animate_mutex.unlock();
    }
}
```

7.9.3.9 init()

```
void HashTable::init (
              std::vector< int > v,
              int KEY )
```

Definition at line 6 of file init.cpp.

```
{
    if(table != nullptr)
    {
        for(int i = 0; i < HASH_KEY; i++)
        {
            delete table[i];
            table[i] = nullptr;
        }
        delete[] table;
    }
    HASH_KEY = key;
    table = new Head*[HASH_KEY];
    currentScript = scripts[DATA_STRUCTURES_OPERATOR::INIT];

    isAnimate = false;

    for(int i = 0; i < HASH_KEY; i++)
    {
        Sprite* spr = new Sprite(render);
        spr->setFont(font);
        spr->linking("hash-table/head");
        spr->setFontColor(fontColor);
        spr->coloring(SDL_Color{bgColor.r, bgColor.r, bgColor.r, 255});
        table[i] = new Head(spr);
    }
    for(int i : v)
    {
        int k = i % HASH_KEY;
        table[k]->root = insert(table[k]->root, i);
    }
    locating(table, 0, 0);
}
```

7.9.3.10 insert() [1/2]

```
void HashTable::insert (
            int key )
```

Definition at line 73 of file insert.cpp.

```
{
    if(table == nullptr) defaultSetting();
    currentScript = scripts[DATA_STRUCTURES_OPERATOR::INSERT];

    isAnimate = true;

    highlight({0});
    waitForStep();
    unhighlight({0});

    highlight({1, 2});
    waitForStep();
    unhighlight({1, 2});

    int k = key % HASH_KEY;

    if(isAnimate && table[k]->root == nullptr)
    {
        animate_mutex.lock();
        table[k]->sprite->highlight();
        animate_mutex.unlock();

        waitForStep();

        animate_mutex.lock();
        table[k]->sprite->unhighlight();
        animate_mutex.unlock();
    }

    table[k]->root = insert(table[k]->root, key);
    locating(table, 0, 0);
}
```

7.9.3.11 insert() [2/2]

```
HashTable::Node * HashTable::insert (
            Node * root,
            int k )  [protected]
```

Definition at line 4 of file insert.cpp.

```
{
    if(node == nullptr)
    {
        Sprite* spr = new Sprite(render);
        spr->setFont(font);
        spr->linking("hash-table/node");
        spr->setText(NUMBER::intToString(k));
        spr->setFontColor(fontColor);
        spr->coloring(nodeColor);
        highlight({1, 2});
        waitForStep();
        unhighlight({1, 2});
        return new Node(k, spr);
    }
    if(isAnimate)
    {
        animate_mutex.lock();
        node->sprite->highlight();
        animate_mutex.unlock();
        waitForStep();
    }
    if(node->pnext == nullptr)
    {
        if(isAnimate)
        {
            animate_mutex.lock();
            node->sprite->unhighlight();
            animate_mutex.unlock();
        }
        Sprite* spr = new Sprite(render);
        spr->setFont(font);
        spr->linking("hash-table/node");
        spr->setText(NUMBER::intToString(k));
        node->pnext = new Node(k, spr);

        if(isAnimate)
        {
            animate_mutex.lock();
            node->pnext->sprite->highlight();
            animate_mutex.unlock();
        }

        highlight({10});
        waitForStep();
        unhighlight({10});

        if(isAnimate)
        {
            animate_mutex.lock();
            node->pnext->sprite->unhighlight();
            animate_mutex.unlock();
        }
    }else
    {
        highlight({7, 8, 9});
        waitForStep();
        unhighlight({7, 8, 9});
        if(isAnimate)
        {
            animate_mutex.lock();
            node->sprite->unhighlight();
            animate_mutex.unlock();
        }
        node->pnext = insert(node->pnext, k);
    }
    return node;
}
```

7.9.3.12 isReceiveEvent()

```
bool HashTable::isReceiveEvent (
               SDL_Event & e )
```

Definition at line 3 of file event.cpp.

```
{
    switch(e.type)
    {
        case SDL_MOUSEBUTTONDOWN:
            if(currentScript != nullptr && currentScript->isReceiveEvent(e)) return true;
            if(e.motion.x < viewport.x || viewport.x + viewport.w < e.motion.x) return false;
             if(e.motion.y < viewport.y || viewport.y + viewport.h < e.motion.y) return false;
             if(e.button.button == SDL_BUTTON_LEFT) return false;
             if(table == nullptr) return false;
             return true;
             break;
         case SDL_MOUSEMOTION:
             if(isMoving) return true;
             if(currentScript == nullptr) return false;
             if(currentScript->isReceiveEvent(e)) return true;
             return false;
             break;
         default:
             return false;
             break;
    }
}
```

7.9.3.13 locating() [1/2]

```
int HashTable::locating (
            Head ** table,
            int shiftDown,
            int shiftRight )  [protected]
```

Definition at line 78 of file constructor.cpp.

```
{
    if(table == nullptr) return 0;

    for(int i = 0; i < HASH_KEY; i++)
    {
        if(table[i] == nullptr) continue;

        table[i]->sprite->locatingX(shiftX + shiftRight * distanceX);
        table[i]->sprite->locatingY(shiftY + (i + shiftDown) * distanceY);
        table[i]->sprite->aligning(HORIZONTAL_ALIGN::CENTER, VERTICAL_ALIGN::CENTER);
        if(table[i]->root == nullptr) table[i]->sprite->show();
        else
        {
            table[i]->sprite->hide();
            locating(table[i]->root, i + shiftDown, shiftRight);
        }
    }

    return 0;
}
```

7.9.3.14 locating() [2/2]

```
int HashTable::locating (
            Node * node,
            int shiftDown,
            int shiftRigh )  [protected]
```

Definition at line 66 of file constructor.cpp.

```
{
    if(node == nullptr) return 0;

    node->sprite->locatingX(shiftX + shiftRight * distanceX);
    node->sprite->locatingY(shiftY + shiftDown * distanceY);
    node->sprite->aligning(HORIZONTAL_ALIGN::CENTER, VERTICAL_ALIGN::CENTER);

    locating(node->pnext, shiftDown, shiftRight + 1);
    return 0;
}
```

7.9.3.15 react()

```
Button * HashTable::react (
            SDL_Event & e )
```

Definition at line 27 of file event.cpp.

```
{
    switch(e.type)
    {
        case SDL_MOUSEBUTTONDOWN:
            if(currentScript != nullptr && currentScript->isReceiveEvent(e))
            {
                return currentScript->react(e);
            }
            if(isMoving)
            {
                isMoving = false;
                int dx = e.motion.x - lastMousePressed.x;
                int dy = e.motion.y - lastMousePressed.y;
                shiftX += dx;
                shiftY += dy;
            }else
            {
                isMoving = true;
                lastMousePressed.x = e.motion.x;
                lastMousePressed.y = e.motion.y;
            }
            return nullptr;
            break;
        case SDL_MOUSEMOTION:
        {
            if(currentScript != nullptr && currentScript->isReceiveEvent(e))
                return currentScript->react(e);
            if(!isMoving) return nullptr;
            int dx = e.motion.x - lastMousePressed.x;
            int dy = e.motion.y - lastMousePressed.y;
            lastMousePressed.x = e.motion.x;
            lastMousePressed.y = e.motion.y;
            shiftX += dx;
            shiftY += dy;
            locating(table, 0, 0);
            return nullptr;
            break;
        }
        defaut:
            return nullptr;
            break;
    }
    return nullptr;
}
```

7.9.3.16 remove() [1/2]

```
void HashTable::remove (
                int key )
```

Definition at line 41 of file remove.cpp.

```
{
    if(table == nullptr) return;
    currentScript = scripts[DATA_STRUCTURES_OPERATOR::DELETE];
    isAnimate = true;

    highlight({0});
    waitForStep();
    unhighlight({0});

    highlight({1, 2});
    waitForStep();
    unhighlight({1, 2});

    if(table[key % HASH_KEY]->root == nullptr)
    {
        highlight({3, 4, 5});
        waitForStep();
        unhighlight({3, 4, 5});
        return;
    }

    if(table[key % HASH_KEY]->root->key == key)
    {
        if(isAnimate)
        {
            animate_mutex.lock();
            table[key % HASH_KEY]->root->sprite->highlight();
            animate_mutex.unlock();
        }
        highlight({6, 7, 8});
        waitForStep();
        unhighlight({6, 7, 8});
        if(isAnimate)
        {
            animate_mutex.lock();
            table[key % HASH_KEY]->root->sprite->unhighlight();
            animate_mutex.unlock();
        }

        isAnimate = false;
        table[key % HASH_KEY]->root = remove(table[key % HASH_KEY]->root, key);
        return ;
    }
    highlight({10});
    waitForStep();
    table[key % HASH_KEY]->root = remove(table[key % HASH_KEY]->root, key);
    unhighlight({10});
    locating(table[key % HASH_KEY]->root, key % HASH_KEY, 0);
}
```

7.9.3.17 remove() [2/2]

```
HashTable::Node * HashTable::remove (
              Node * root,
              int k )   [protected]
```

Definition at line 4 of file remove.cpp.

```
{
    if(node == nullptr)
    {
        return nullptr;
    }
    if(isAnimate)
    {
        animate_mutex.lock();
        node->sprite->highlight();
        animate_mutex.unlock();
        waitForStep();
    }
    if(node->key == key)
    {
        Node* tmp = node->pnext;
        node->pnext = nullptr;
        delete node;
        return tmp;
    }
    if(isAnimate)
    {
        animate_mutex.lock();
        node->sprite->unhighlight();
        animate_mutex.unlock();
    }
    if(node->pnext->key == key)
    {
        highlight({11, 12, 13, 14});
        waitForStep();
        unhighlight({11, 12, 13, 14});
    }
    node->pnext = remove(node->pnext, key);
    return node;
}
```

7.9.3.18 renderEdges()

```
void HashTable::renderEdges (
              Node * src,
              Node * dst )   [protected]
```

7.9.3.19 rendering()

```
void HashTable::rendering ( )
```

Definition at line 22 of file rendering.cpp.

```
{
    if(table == nullptr) return ;
    locating(table, 0, 0);
    SDL_RenderSetViewport(render, &viewport);

    for(int i = 0; i < HASH_KEY; i++)
    {
        if(table[i]->root == nullptr)
            table[i]->sprite->rendering();
        else
        {
            Node* current = table[i]->root;
            while(current != nullptr)
            {
                if(current->pnext != nullptr) drawEdge(current, current->pnext);
                current->sprite->rendering();
                current = current->pnext;
            }
        }
    }
    if(currentScript != nullptr)
    {
        SDL_RenderSetViewport(render, nullptr);
        currentScript->rendering();
    }
}
```

7.9.3.20 search() [1/2]

```
bool HashTable::search (
            int key )
```

Definition at line 33 of file search.cpp.

```
{
    if(table == nullptr) return false;
    isAnimate = true;
    currentScript = scripts[DATA_STRUCTURES_OPERATOR::SEARCH];

    highlight({0});
    waitForStep();
    unhighlight({0});

    highlight({1, 2});
    waitForStep();
    unhighlight({1, 2});

    if(isAnimate)
    {
        animate_mutex.lock();
        table[key % HASH_KEY]->sprite->highlight();
        animate_mutex.unlock();
    }
    highlight({3});
    waitForStep();
    if(isAnimate)
    {
        animate_mutex.lock();
        table[key % HASH_KEY]->sprite->unhighlight();
        animate_mutex.unlock();
    }
    search(table[key % HASH_KEY]->root, key);

    unhighlight({3});

    return true;
}
```

7.9.3.21 search() [2/2]

```
bool HashTable::search (
              Node * root,
              int k )  [protected]
```

Definition at line 4 of file search.cpp.

```
{
    if(node == nullptr)
    {
        highlight({9});
        waitForStep();
         unhighlight({9});
         return false;
     }
     if(isAnimate)
     {
         animate_mutex.lock();
         node->sprite->highlight();
         animate_mutex.unlock();
         waitForStep();
         animate_mutex.lock();
         node->sprite->unhighlight();
         animate_mutex.unlock();
     }
     if(node->key == key)
     {
         highlight({4, 5});
         waitForStep();
         unhighlight({4, 5});
         return true;
     }
     return search(node->pnext, key);
 }
```

7.9.3.22 setEdgesColor()

```
void HashTable::setEdgesColor (
              SDL_Color c )
```

7.9.3.23 setNodeColor()

```
void HashTable::setNodeColor (
              SDL_Color bg,
              SDL_Color fg )
```

7.9.3.24 setting()

```
void HashTable::setting (
              SDL_Color c1,
              SDL_Color c2,
              SDL_Color c3,
              SDL_Color c4 )
```

Definition at line 116 of file constructor.cpp.

```
{
    nodeColor = c2;
    fontColor = c3;
    edgesColor = c4;
    bgColor.r = nodeColor.r * 0.5;
    bgColor.g = nodeColor.g * 0.5;
    bgColor.b = nodeColor.b * 0.5;
    bgColor.a = 255;

    for(int i = 0; i < HASH_KEY; i++)
    {
        table[i]->sprite->coloring(bgColor);
        table[i]->sprite->setFontColor(fontColor);

        Node* current = table[i]->root;

        while(current != nullptr)
        {
            current->sprite->coloring(nodeColor);
            current->sprite->setFontColor(fontColor);
            current = current->pnext;
        }
    }
}
```

7.9.3.25 slowDown()

```
void HashTable::slowDown ( )
```

7.9.3.26 speedUp()

```
void HashTable::speedUp ( )
```

7.9.3.27 unhighlight()

```
void HashTable::unhighlight (
              std::vector< int > l ) [protected]
```

Definition at line 36 of file step.cpp.

```
{
    if(isAnimate)
    {
        animate_mutex.lock();
        for(int i = 0; i < l.size(); i++)
        {
            currentScript->unhighlight(l[i]);
        }
        animate_mutex.unlock();
    }
}
```

7.9.3.28 waitForStep()

```
void HashTable::waitForStep ( )  [protected]
```

Definition at line 2 of file step.cpp.

```
{
    if(isAnimate)
    {
        ds_mutex.unlock();
        std::this_thread::sleep_for(std::chrono::milliseconds(stepWait));
        ds_mutex.lock();
    }
     std::lock_guard<std::mutex> pause_lock(pause_mutex);
     if(isPause == false)
     {
         return ;
     }

     ds_mutex.unlock();
     std::unique_lock<std::mutex> lk(step_mutex);
     step_cv.wait(lk, [&]{return isQueue == true;});
     isQueue = false;
     ds_mutex.lock();
}
```

The documentation for this class was generated from the following files:

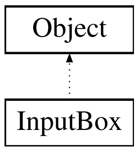
- include/data_structures/hash_table.hpp
- src/data_structures/event.cpp
- src/hash_table/constructor.cpp
- src/hash_table/destructor.cpp
- src/hash_table/operator/init.cpp
- src/hash_table/operator/insert.cpp
- src/hash_table/operator/remove.cpp
- src/hash_table/operator/search.cpp
- src/hash_table/rendering.cpp
- src/hash_table/step.cpp

7.10 InputBox Class Reference

Register for user keyboard input.

`#include <inputbox.hpp>`

Inheritance diagram for InputBox:



Public Member Functions

- InputBox (SDL_Renderer ∗render, TTF_Font ∗font)
- ∼InputBox ()
- void setDuplicate (int n, int m)
- void linking (std::string n)
- DATA_STRUCTURES_OPERATOR getOperator ()
- bool isReceiveEvent (SDL_Event &e)
- Button ∗ react (SDL_Event &e)
- std::string getText (int index)
- void setText (int index, std::string text)
- void setText (std::string text)
- void rendering ()

Protected Member Functions

- void initBackground (const json &mem)
- void initButtons (const json &mem)
- void initSprites (const json &mem)
- void initOperator (const json &mem)
- void importFromJson ()
- bool isButtonReceiveEvent (SDL_Event &event)
- bool isInputReceiveEvent (SDL_Event &event)

Protected Attributes

- int n
- int m

Private Member Functions

- void importFromJson (const json &mem)
- std::string getText ()

7.10.1 Detailed Description

Register for user keyboard input.

Drawable

Definition at line 18 of file inputbox.hpp.

7.10.2 Constructor & Destructor Documentation

7.10.2.1 InputBox()

```
InputBox::InputBox (
            SDL_Renderer * render,
            TTF_Font * font )
```

Definition at line 9 of file constructor.cpp.

```
                                                    : Object(render)
{
    this->font = font;
    this->render = render;
    n = 1;
    m = 1;
}
```

7.10.2.2 ∼InputBox()

```
InputBox::∼InputBox ( )
```

Definition at line 3 of file destructor.cpp.

```
{
    for(auto& i : buts) delete i;
    for(auto& i : inputs) delete i;
}
```

7.10.3 Member Function Documentation

7.10.3.1 getOperator()

```
DATA_STRUCTURES_OPERATOR InputBox::getOperator ( )
```

Definition at line 3 of file operator.cpp.

```
{
    return op;
}
```

7.10.3.2 getText()

```
std::string InputBox::getText (
                int index )
```

Definition at line 3 of file typing.cpp.

```
{
    if(index == -1 || index >= (int) inputs.size())
        return "";
    return inputs[index]->getText();
}
```

7.10.3.3 importFromJson()

```
void InputBox::importFromJson ( )  [protected]
```

Definition at line 39 of file constructor.cpp.

```
{
    json* mem = JSON::readFile(PATH::ATB::INPUTBOX_ + name + ".json");

    if(mem->contains("background"))
        initBackground((*mem)["background"]);

    if(mem->contains("buttons"))
        initButtons((*mem)["buttons"]);

    if(mem->contains("sprites"))
        initSprites((*mem)["sprites"]);
    if(mem->contains("operator"))
        initOperator((*mem)["operator"]);
    delete mem;
}
```

7.10.3.4 initBackground()

```
void InputBox::initBackground (
            const json & mem )  [protected]
```

Definition at line 56 of file constructor.cpp.

```
{
    Object::importFromJson(mem);
}
```

7.10.3.5 initButtons()

```
void InputBox::initButtons (
            const json & mem )  [protected]
```

Definition at line 61 of file constructor.cpp.

```
{
    for(auto& i : mem)
    {
        buts.push_back(new Button(render));
        if(i.contains("name")) buts.back()->linking(i["name"]);
        if(i.contains("dx")) buts.back()->move(i["dx"], i["dy"]);
    }
}
```

7.10.3.6 initOperator()

```
void InputBox::initOperator (
            const json & mem )  [protected]
```

Definition at line 23 of file constructor.cpp.

```
{
    std::string type = mem.get<std::string>();

    if(type == "INIT") op = DATA_STRUCTURES_OPERATOR::INIT;
    else if (type == "INSERT") op = DATA_STRUCTURES_OPERATOR::INSERT;
    else if (type == "DELETE") op = DATA_STRUCTURES_OPERATOR::DELETE;
    else if (type == "SEARCH") op = DATA_STRUCTURES_OPERATOR::SEARCH;
    else if (type == "TOP") op = DATA_STRUCTURES_OPERATOR::TOP;
    else if (type == "SIZE") op = DATA_STRUCTURES_OPERATOR::SIZE;
    else if (type == "SCC") op = DATA_STRUCTURES_OPERATOR::SCC;
    else if (type == "DIJKSTRA") op = DATA_STRUCTURES_OPERATOR::DIJKSTRA;
    else if (type == "MST") op = DATA_STRUCTURES_OPERATOR::MST;
    else if (type == "SETTING") op = DATA_STRUCTURES_OPERATOR::SETTING;
}
```

7.10.3.7 initSprites()

```
void InputBox::initSprites (
            const json & mem )  [protected]
```

Definition at line 71 of file constructor.cpp.

```
{
    for(auto& i : mem)
    {
        if(n * m >= 1 && n > 0 && m > 0 && i["name"].get<std::string>() == "graph/init/edge")
        {
            int dx = 35;
            int dy = 35;

            for(int u = 0; u < n; u++)
            {
                for(int v=  0; v < m; v++)
                {
                    inputs.push_back(new Sprite(render));
                    inputs.back()->setFont(font);
                    if(i.contains("name")) inputs.back()->linking(i["name"].get<std::string>());
                    inputs.back()->moveX(dx * u);
                    inputs.back()->moveY(dy * v);
                }
            }
        }else
        {
            inputs.push_back(new Sprite(render));
            inputs.back()->setFont(font);
            if(i.contains("name")) inputs.back()->linking(i["name"].get<std::string>());
        }
    }
}
```

7.10.3.8 isButtonReceiveEvent()

```
bool InputBox::isButtonReceiveEvent (
            SDL_Event & event )  [protected]
```

Definition at line 46 of file event.cpp.

```
{
    for(auto& but : buts)
    {
        if(but->isReceiveEvent(e))
        {
            return true;
        }
    }
    return false;
}
```

7.10.3.9 isInputReceiveEvent()

```
bool InputBox::isInputReceiveEvent (
            SDL_Event & event )  [protected]
```

Definition at line 19 of file event.cpp.

```
{
    switch(e.type)
    {
        case SDL_MOUSEBUTTONDOWN:
            for(auto& inp : inputs)
                if(inp->isLieInside(e.motion.x, e.motion.y))
                    return true;
            return false;
            break;
        case SDL_MOUSEMOTION:
            for(auto& inp : inputs)
                if(inp->isLieInside(e.motion.x, e.motion.y))
                    return true;
            return false;
            break;
        case SDL_KEYDOWN:
            if(typingIndex == -1)
                return false;
            return true;
            break;
        default:
            return false;
            break;
    }
}
```

7.10.3.10 isReceiveEvent()

```
bool InputBox::isReceiveEvent (
               SDL_Event & e )
```

Definition at line 4 of file event.cpp.

```
{
    switch(e.type)
    {
        case SDL_QUIT:
            return false;
            break;
        default:
            if(isButtonReceiveEvent(e) || isInputReceiveEvent(e))
                return true;
            return false;
            break;
    }
}
```

7.10.3.11 linking()

```
void InputBox::linking (
               std::string n )
```

Definition at line 17 of file constructor.cpp.

```
{
    name = n;
    importFromJson();
}
```

7.10.3.12 react()

```
Button * InputBox::react (
               SDL_Event & e )
```

Definition at line 58 of file event.cpp.

```
{
    switch(e.type)
    {
        case SDL_KEYDOWN:
        {
            if(typingIndex == -1)
```

```
                return nullptr;
            if(e.key.keysym.sym == SDLK_BACKSPACE)
            {
                inputs[typingIndex]->backspace();
            }
            if(SDLK_SPACE <= e.key.keysym.sym && e.key.keysym.sym <= SDLK_z)
            {
                inputs[typingIndex]->typing(e.key.keysym.sym);
            }
            return nullptr;
            break;
        }
        case SDL_MOUSEMOTION:
        {
            for(auto& but : buts)
                but->isHover(e.motion.x, e.motion.y);
            return nullptr;
            break;
        }
        case SDL_MOUSEBUTTONDOWN:
        {
            int i = 0;
            for(auto& inp : inputs)
            {
                if(inp->isLieInside(e.motion.x, e.motion.y))
                {
                    typingIndex = i;
                    return nullptr;
                }
                i++;
            }
            typingIndex = 0;
            for(auto& but : buts)
                if(but->isClicked(e.motion.x, e.motion.y))
                    return but;
            return nullptr;
            break;
        }
        default:
            return nullptr;
    }
}
```

7.10.3.13 rendering()

`void InputBox::rendering ( ) [virtual]`

Reimplemented from Object.

Definition at line 3 of file rendering.cpp.

```
{
    Object::rendering();
    for(auto& i : buts) i->rendering();
    for(auto& i : inputs) i->rendering();
}
```

7.10.3.14 setDuplicate()

```
void InputBox::setDuplicate (
              int n,
              int m )
```

Definition at line 103 of file constructor.cpp.

```
{
    this->n = n;
    this->m = m;
}
```

7.10.3.15 setText() [1/2]

```
void InputBox::setText (
             int index,
             std::string text )
```

Definition at line 10 of file typing.cpp.

```
{
    if (index != -1 && index < (int) inputs.size())
    {
        inputs[index]->setText(text);
    }
}
```

7.10.3.16 setText() [2/2]

```
void InputBox::setText (
             std::string text )
```

Definition at line 18 of file typing.cpp.

```
{
    if (typingIndex != -1)
    {
        inputs[typingIndex]->setText(text);
    }
}
```

7.10.4 Member Data Documentation

7.10.4.1 m

```
int InputBox::m  [protected]
```

Definition at line 38 of file inputbox.hpp.

7.10.4.2 n

```
int InputBox::n  [protected]
```

Definition at line 38 of file inputbox.hpp.

The documentation for this class was generated from the following files:

- include/inputbox.hpp
- src/inputbox/constructor.cpp
- src/inputbox/destructor.cpp
- src/inputbox/event.cpp
- src/inputbox/operator.cpp
- src/inputbox/rendering.cpp
- src/inputbox/typing.cpp

7.11 maxHeap Class Reference

```
#include <maxheap.hpp>
```

Public Member Functions

- maxHeap ()
- ∼maxHeap ()
- void init (std::vector< int > &value)
- void insert (int value)
- void pop ()
- int top ()
- int size ()

Protected Member Functions

- void heapify (int index)
- void swap (int index1, int index2)
- bool swapable (int index1, int index2)

7.11.1 Detailed Description

Definition at line 7 of file maxheap.hpp.

7.11.2 Constructor & Destructor Documentation

7.11.2.1 maxHeap()

```
maxHeap::maxHeap ( )
```

Definition at line 8 of file constructor.cpp.

```
{

}
```

7.11.2.2 ∼maxHeap()

```
maxHeap::∼maxHeap ( )
```

Definition at line 7 of file destructor.cpp.

```
{
    for(auto &i : value)
    {
        delete i;
    }
}
```

7.11.3 Member Function Documentation

7.11.3.1 heapify()

```
void maxHeap::heapify (
            int index )  [protected]
```

Definition at line 13 of file heapify.cpp.

```
{
    int left = 2 * index + 1;
    int right = 2 * index + 2;
    int largest = index;

    if(left < value.size() && swapable(index, left))
    {
        largest = left;
    }

    if(right < value.size() && swapable(largest, right))
    {
        largest = right;
    }

    if(largest != index)
    {
        swap(index, largest);
        heapify(largest);
    }
}
```

7.11.3.2 init()

```
void maxHeap::init (
            std::vector< int > & value )
```

Definition at line 4 of file init.cpp.

```
{
    for(auto &i : value)
    {
        insert(i);
    }
}
```

7.11.3.3 insert()

```
void maxHeap::insert (
            int value )
```

Definition at line 3 of file insert.cpp.

```
{
    Node* node = new Node(value);
    this->value.push_back(node);

    int index = this->value.size() - 1;

     if(index == 0) return ;

     do
     {
        int parent = (index - 1) / 2;

        if(swapable(parent, index))
        {
            swap(parent, index);
        }

        index = parent;
    }while(index !=0);

}
```

7.11.3.4 pop()

`void maxHeap::pop ( )`

Definition at line 3 of file remove.cpp.

```
{
    if(value.size() == 0)
    {
        return;
    }

    swap(0, value.size() - 1);
    value.pop_back();

    heapify(0);
}
```

7.11.3.5 size()

`int maxHeap::size ( )`

Definition at line 4 of file size.cpp.

```
{
    return value.size();
}
```

7.11.3.6 swap()

```
void maxHeap::swap (
              int index1,
              int index2 )   [protected]
```

Definition at line 3 of file heapify.cpp.

```
{
    std::swap(value[index1], value[index2]);
}
```

7.11.3.7 swapable()

```
bool maxHeap::swapable (
             int index1,
             int index2 )  [protected]
```

Definition at line 8 of file heapify.cpp.

```
{
    return value[index1]->value < value[index2]->value;
}
```

7.11.3.8 top()

```
int maxHeap::top ( )
```

Definition at line 3 of file getmax.cpp.

```
{
    return value[0]->value;
}
```

The documentation for this class was generated from the following files:

- include/data_structures/maxheap.hpp
- src/maxheap/constructor.cpp
- src/maxheap/destructor.cpp
- src/maxheap/operator/getmax.cpp
- src/maxheap/operator/heapify.cpp
- src/maxheap/operator/init.cpp
- src/maxheap/operator/insert.cpp
- src/maxheap/operator/remove.cpp
- src/maxheap/operator/size.cpp

7.12 minHeap Class Reference

heap class.

`#include <minheap.hpp>`

Public Member Functions

- minHeap (SDL_Renderer ∗render, std::mutex &m, TTF_Font ∗f, SDL_Rect v, int cap)
- ∼minHeap ()
- void setmax ()
- void setmin ()
- void init (std::vector< int > value)
- void insert (int value)
- void pop ()
- int top ()
- int size ()
- void closeScript ()
- bool isReceiveEvent (SDL_Event &e)
- Button ∗ react (SDL_Event &e)
- void rendering ()
- void setting (SDL_Color c1, SDL_Color c2, SDL_Color c3, SDL_Color c4)

Protected Member Functions

- void heapify (int index)
- void swap (int index1, int index2)
- bool swapable (int index1, int index2)
- void realInsert (int value)
- int locating (int id, int shiftDown, int shiftRight)
- void renderLine (Node ∗src, Node ∗dst)
- void waitForStep ()
- void highlight (std::vector< int > l)
- void unhighlight (std::vector< int > l)

7.12.1 Detailed Description

heap class.

Drawable heap.

Default is min heap.

Definition at line 25 of file minheap.hpp.

7.12.2 Constructor & Destructor Documentation

7.12.2.1 minHeap()

```
minHeap::minHeap (
               SDL_Renderer * render,
               std::mutex & m,
               TTF_Font * f,
               SDL_Rect v,
               int cap )
```

Definition at line 9 of file constructor.cpp.

```
                                                                                   : ds_mutex(m)
{
    render = r;
    font = f;
    viewport = v;
    capacity = cap;
    nodeColor = {20, 85, 185, 255};
    fontColor = {255, 255, 255, 255};
    edgesColor = {255, 255, 255, 255};
    shiftX = 20;
    shiftY = 20;
    distanceX = 60;
    distanceY = 80;
    isMoving = false;
    stepWait = 600;
    isAnimate = false;
    inverse = false;
    std::string fontpath = PATH::ASSETS::FONTS_ + "nimbus-sans-l/regular.otf";
    scriptFont = TTF_OpenFont(fontpath.c_str(), 18);

    currentScript = nullptr;

    Script* insert = new Script(render, scriptFont);
    insert->linking("minheap/insert");
    scripts[DATA_STRUCTURES_OPERATOR::INSERT] = insert;

    Script* remove = new Script(render, scriptFont);
    remove->linking("minheap/remove");
    scripts[DATA_STRUCTURES_OPERATOR::DELETE] = remove;

    Script* search = new Script(render, scriptFont);
    search->linking("minheap/top");
    scripts[DATA_STRUCTURES_OPERATOR::TOP] = search;

    Script* init = new Script(render, scriptFont);
    init->linking("minheap/init");

    scripts[DATA_STRUCTURES_OPERATOR::INIT] = init;

    Script* size = new Script(render, scriptFont);
    size->linking("minheap/size");
    scripts[DATA_STRUCTURES_OPERATOR::SIZE] = size;

}
```

7.12.2.2 ∼minHeap()

minHeap::∼minHeap ( )

Definition at line 7 of file destructor.cpp.

```
{
   for(auto &i : value)
    {
        delete i;
    }
    if(scriptFont != nullptr) TTF_CloseFont(scriptFont);
    for(auto & i : scripts)
        delete i.second;
    scripts.clear();
}
```

7.12.3 Member Function Documentation

7.12.3.1 closeScript()

void minHeap::closeScript ( )

Definition at line 74 of file event.cpp.

```
{
    currentScript = nullptr;
}
```

7.12.3.2 heapify()

```
void minHeap::heapify (
            int index )  [protected]
```

Definition at line 13 of file heapify.cpp.

```
{
    int left = 2 * index + 1;
    int right = 2 * index + 2;
    int largest = index;

    if(left < value.size() && swapable(index, left))
    {
        largest = left;
    }

    if(right < value.size() && swapable(largest, right))
    {
        largest = right;
    }

    if(largest != index)
    {
        if(isAnimate)
        {
            animate_mutex.lock();
            value[index]->sprite->highlight();
            value[largest]->sprite->highlight();
            animate_mutex.unlock();
            waitForStep();
        }
        swap(index, largest);
        if(isAnimate)
        {
            animate_mutex.lock();
            value[index]->sprite->unhighlight();
            value[largest]->sprite->unhighlight();
            animate_mutex.unlock();
        }
        heapify(largest);
    }
}
```

7.12.3.3 highlight()

```
void minHeap::highlight (
            std::vector< int > l )  [protected]
```

Definition at line 5 of file step.cpp.

```
{
    if(isAnimate)
    {
        animate_mutex.lock();
        for(int i = 0; i < l.size(); i++)
        {
            currentScript->highlight(l[i]);
        }
        animate_mutex.unlock();
    }
}
```

7.12.3.4 init()

```
void minHeap::init (
            std::vector< int > value )
```

Definition at line 4 of file init.cpp.

```
{
    currentScript = scripts[DATA_STRUCTURES_OPERATOR::INIT];
    this->value.clear();
    for(auto &i : value)
    {
        realInsert(i);
     }
}
```

7.12.3.5 insert()

```
void minHeap::insert (
            int value )
```

Definition at line 61 of file insert.cpp.

```
{
    shiftX = 20;
    shiftY = 20;
    currentScript = scripts[DATA_STRUCTURES_OPERATOR::INSERT];
    isAnimate = true;

    highlight({0});
    waitForStep();
    unhighlight({0});

    realInsert(value);
}
```

7.12.3.6 isReceiveEvent()

```
bool minHeap::isReceiveEvent (
            SDL_Event & e )
```

Definition at line 3 of file event.cpp.

```
{
    std::lock_guard<std::mutex> lk(animate_mutex);
    switch(e.type)
    {
        case SDL_MOUSEBUTTONDOWN:
            if(currentScript != nullptr && currentScript->isReceiveEvent(e)) return true;
             if(e.motion.x < viewport.x || viewport.x + viewport.w < e.motion.x) return false;
             if(e.motion.y < viewport.y || viewport.y + viewport.h < e.motion.y) return false;
             if(e.button.button == SDL_BUTTON_LEFT) return false;
             if(value.empty()) return false;
             return true;
             break;
         case SDL_MOUSEMOTION:
             if(isMoving) return true;
             if(currentScript == nullptr) return false;
             if(currentScript->isReceiveEvent(e)) return true;
             return false;
             break;
         default:
             return false;
             break;
     }
}
```

7.12.3.7 locating()

```
int minHeap::locating (
              int id,
              int shiftDown,
              int shiftRight )  [protected]
```

Definition at line 4 of file rendering.cpp.

```
{
    if(id < 0 || id >= value.size())
    {
        int shift = log2(value.size()) - shiftDown + 1;
        return std::max(0, (1 « shift) - 1);
     }

     int left = std::max(0, locating(id * 2 + 1, shiftDown + 1, shiftRight));
     Node* node = value[id];
     if(node->sprite != nullptr)
     {
         node->sprite->locatingX(shiftX + shiftRight * distanceX + left * distanceX);
         node->sprite->locatingY(shiftY + shiftDown * distanceY);
         node->sprite->aligning(HORIZONTAL_ALIGN::CENTER, VERTICAL_ALIGN::CENTER);
     }
     int right = locating(id * 2 + 2, shiftDown + 1, shiftRight + left + 1);

     int shift = log2(value.size()) - shiftDown + 1;
     return left + right + 1;
}
```

7.12.3.8 pop()

```
void minHeap::pop ( )
```

Definition at line 5 of file remove.cpp.

```
{
    if(!value.empty()) valuesReachedZero = false;
    if(valuesReachedZero) return ;
    currentScript = scripts[DATA_STRUCTURES_OPERATOR::DELETE];
     isAnimate = true;
     highlight({0});
     waitForStep();
     unhighlight({0});

     if(value.size() == 0)
     {
         valuesReachedZero = true;
         highlight({1, 2, 3});
         waitForStep();
         unhighlight({1, 2, 3});
         return;
     }
     if(isAnimate)
     {
         animate_mutex.lock();
         value[0]->sprite->highlight();
         value.back()->sprite->highlight();
         animate_mutex.unlock();
         waitForStep();
     }
     swap(0, value.size() - 1);

     if(isAnimate)
     {
         animate_mutex.lock();
         value[0]->sprite->unhighlight();
         value.back()->sprite->unhighlight();
         animate_mutex.unlock();
     }

     highlight({5});
     waitForStep();
     unhighlight({5});

     value.pop_back();

    highlight({6});
    waitForStep();

    heapify(0);

    unhighlight({6});
    waitForStep();
}
```

7.12.3.9 react()

```
Button * minHeap::react (
            SDL_Event & e )
```

Definition at line 28 of file event.cpp.

```
{
    std::lock_guard<std::mutex> lk(animate_mutex);
    switch(e.type)
    {
        case SDL_MOUSEBUTTONDOWN:
            if(currentScript != nullptr && currentScript->isReceiveEvent(e))
            {
                return currentScript->react(e);
            }
            if(isMoving)
            {
                isMoving = false;
                int dx = e.motion.x - lastMousePressed.x;
                int dy = e.motion.y - lastMousePressed.y;
                shiftX += dx;
                shiftY += dy;
            }else
            {
                isMoving = true;
                lastMousePressed.x = e.motion.x;
                lastMousePressed.y = e.motion.y;
            }
            return nullptr;
            break;
        case SDL_MOUSEMOTION:
        {
            if(currentScript != nullptr && currentScript->isReceiveEvent(e))
                return currentScript->react(e);
            if(!isMoving) return nullptr;
            int dx = e.motion.x - lastMousePressed.x;
            int dy = e.motion.y - lastMousePressed.y;
            lastMousePressed.x = e.motion.x;
            lastMousePressed.y = e.motion.y;
            shiftX += dx;
            shiftY += dy;
            return nullptr;
            break;
        }
        defaut:
            return nullptr;
            break;
    }
    return nullptr;
}
```

7.12.3.10 realInsert()

```
void minHeap::realInsert (
            int value )   [protected]
```

Definition at line 4 of file insert.cpp.

```
{
    if(inverse) value *= -1;
```

```
   Sprite* sprite = new Sprite(render);
   sprite->setFont(font);
   sprite->linking("AVL/node");
    sprite->setFontColor(fontColor);
    sprite->coloring(nodeColor);
    sprite->setText(NUMBER::intToString(value));
    Node* node = new Node(value, sprite);

    this->value.push_back(node);

    int index = this->value.size() - 1;
    highlight({1, 2, 3});
    waitForStep();
    unhighlight({1, 2, 3});

    if(index == 0)
    {
        highlight({5, 6, 7});
        waitForStep();
        unhighlight({5, 6, 7});
        return ;
    }
    highlight({9});
    do
    {
        int parent = (index - 1) / 2;

        if(isAnimate)
        {
            animate_mutex.lock();
            this->value[index]->sprite->highlight();
            this->value[parent]->sprite->highlight();
            animate_mutex.unlock();
            waitForStep();
        }
        if(swapable(parent, index))
        {
            highlight({11, 12, 13});
            waitForStep();
            swap(parent, index);
            unhighlight({11, 12, 13});
        }
        if(isAnimate)
        {
            animate_mutex.lock();
            this->value[index]->sprite->unhighlight();
            this->value[parent]->sprite->unhighlight();
            animate_mutex.unlock();
        }
        index = parent;
    }while(index !=0);
    unhighlight({9});
}
```

7.12.3.11 rendering()

```
void minHeap::rendering ( )
```

Definition at line 39 of file rendering.cpp.

```
{
    SDL_RenderSetViewport(render, &viewport);
    std::lock_guard<std::mutex> lock(animate_mutex);
    locating(0, 0, 0);

    for(int i = 1; i < value.size(); i++)
    {
        Node* node = value[i];
        if(node->sprite != nullptr)
        {
            Node* parent = value[(i - 1) / 2];
            renderLine(parent, node);
        }
    }
    for(auto& i : value)
    {
        if(i->sprite != nullptr)
            i->sprite->rendering();
    }
    if(currentScript != nullptr)
    {
        SDL_RenderSetViewport(render, nullptr);
        currentScript->rendering();
    }

}
```

7.12.3.12 renderLine()

```
void minHeap::renderLine (
            Node * src,
            Node * dst )  [protected]
```

Definition at line 26 of file rendering.cpp.

```
{
    SDL_Point psrc = {src->sprite->getX() + src->sprite->getW() / 2, src->sprite->getY() +
    src->sprite->getH() / 2};
    SDL_Point pdst = {dst->sprite->getX() + dst->sprite->getW() / 2, dst->sprite->getY() +
    dst->sprite->getH() / 2};

    SDL_SetRenderDrawColor(render, edgesColor.r, edgesColor.g, edgesColor.b, edgesColor.a);
    for(int i = -1; i <= 1; i++)
    {
        for(int j = -1; j <= 1; j++)
            SDL_RenderDrawLine(render, psrc.x + i, psrc.y + j, pdst.x + i, pdst.y + j);
    }
}
```

7.12.3.13 setmax()

```
void minHeap::setmax ( )
```

Definition at line 59 of file constructor.cpp.

```
{
    inverse = true;
}
```

7.12.3.14 setmin()

```
void minHeap::setmin ( )
```

Definition at line 54 of file constructor.cpp.

```
{
    inverse = false;
}
```

7.12.3.15 setting()

```
void minHeap::setting (
            SDL_Color c1,
            SDL_Color c2,
            SDL_Color c3,
            SDL_Color c4 )
```

Definition at line 64 of file constructor.cpp.

```
{
    bgColor = c1;
    nodeColor = c2;
    fontColor = c3;
    edgesColor = c4;

    for(auto i : value)
    {
        i->sprite->setFontColor(fontColor);
        i->sprite->coloring(nodeColor);
    }
}
```

7.12.3.16 size()

```
int minHeap::size ( )
```

Definition at line 4 of file size.cpp.

```
{
    currentScript = scripts[DATA_STRUCTURES_OPERATOR::SIZE];
    return value.size();
}
```

7.12.3.17 swap()

```
void minHeap::swap (
            int index1,
            int index2 )  [protected]
```

Definition at line 3 of file heapify.cpp.

```
{
    std::swap(value[index1], value[index2]);
}
```

7.12.3.18 swapable()

```
bool minHeap::swapable (
            int index1,
            int index2 )  [protected]
```

Definition at line 8 of file heapify.cpp.

```
{
    return value[index1]->value > value[index2]->value;
}
```

7.12.3.19 top()

```
int minHeap::top ( )
```

Definition at line 4 of file getmin.cpp.

```
{
    currentScript = scripts[DATA_STRUCTURES_OPERATOR::TOP];
    isAnimate = true;
    highlight({0, 1, 2, 3, 4, 5});
    waitForStep();
     unhighlight({0, 1, 2, 3, 4, 5});

     if(value.size() == 0)
         return INT_MAX;
     return value[0]->value;
}
```

7.12.3.20 unhighlight()

```
void minHeap::unhighlight (
            std::vector< int > l )  [protected]
```

Definition at line 18 of file step.cpp.

```
{
     if(isAnimate)
     {
         animate_mutex.lock();
         for(int i = 0; i < l.size(); i++)
         {
             currentScript->unhighlight(l[i]);
         }
         animate_mutex.unlock();
     }
}
```

7.12.3.21 waitForStep()

```
void minHeap::waitForStep ( )  [protected]
```

Definition at line 31 of file step.cpp.

```
{
     if(isAnimate)
     {
         ds_mutex.unlock();
         std::this_thread::sleep_for(std::chrono::milliseconds(stepWait));
         ds_mutex.lock();
     }
     std::lock_guard<std::mutex> pause_lock(pause_mutex);
     if(isPause == false)
     {
         return ;
     }

     ds_mutex.unlock();
     std::unique_lock<std::mutex> lk(step_mutex);
     step_cv.wait(lk, [&]{return isQueue == true;});
     isQueue = false;
     ds_mutex.lock();
}
```

The documentation for this class was generated from the following files:

- include/data_structures/minheap.hpp

- src/minheap/constructor.cpp
- src/minheap/destructor.cpp
- src/minheap/event.cpp
- src/minheap/operator/getmin.cpp
- src/minheap/operator/heapify.cpp
- src/minheap/operator/init.cpp
- src/minheap/operator/insert.cpp
- src/minheap/operator/remove.cpp
- src/minheap/operator/size.cpp
- src/minheap/rendering.cpp
- src/minheap/step.cpp

7.13 MyWindow Class Reference

Window class class that create a window and manage it.

```
#include <window.hpp>
```

Public Member Functions

- MyWindow ()
- ∼MyWindow ()
- bool isClosed ()
- bool isOpen ()
- void run ()

Protected Member Functions

- void initSDL2 ()
- void rendering ()
- void interacting ()
- void getEvent ()
- void react (Button ∗but)
- void setDisplay (std::string name)
- void setInputBox (std::string name)
- void setDataType (DATA_STRUCTURES_TYPE type)
- void runOperator ()
- void getDataFromFile (DATA_STRUCTURES_TYPE type)
- void getDataFromRandom (DATA_STRUCTURES_TYPE type)

7.13.1 Detailed Description

Window class class that create a window and manage it.

Finite state machine

handle thread

Definition at line 26 of file window.hpp.

7.13.2 Constructor & Destructor Documentation

7.13.2.1 MyWindow()

`MyWindow::MyWindow ( )`

Definition at line 28 of file constructor.cpp.

```
{
    HEIGHT = 635;
    WIDTH = 1200;
    status = WINDOW_STATUS::IS_CLOSED;
    FPS = 60;
    viewport = SDL_Rect({0, 0, WIDTH, HEIGHT});

    window = nullptr;
    render = nullptr;
    ds = nullptr;
    inputbox = nullptr;

    initSDL2();

    std::string fontpath = PATH::ASSETS::FONTS_ + "nimbus-sans-l/regular.otf";
    myfont = TTF_OpenFont(fontpath.c_str(), 24);

    display_pool["nullptr"] = nullptr;
    inputbox_pool["nullptr"] = nullptr;
    ds_pool[DATA_STRUCTURES_TYPE::NONE] = nullptr;

    setDisplay(DISPLAY::HOME_);
}
```

7.13.2.2 ∼MyWindow()

`MyWindow::∼MyWindow ( )`

Definition at line 5 of file destructor.cpp.

```
{
   if(render != nullptr)
   {
       SDL_DestroyRenderer(render);
        render = nullptr;
    }
    if(window != nullptr)
    {
        SDL_DestroyWindow(window);
        window = nullptr;
    }
    if(myfont != nullptr) TTF_CloseFont(myfont);

    for(auto &i : thread_pool)
    {
        if(i.joinable())
        {
            i.join();
        }
    }

    for(auto &i : display_pool)
    {
        delete i.second;
    }

    for(auto &i : ds_pool)
    {
        delete i.second;
    }

    for(auto &i : inputbox_pool)
    {
        delete i.second;
    }

    TTF_Quit();
    IMG_Quit();
    SDL_Quit();
}
```

7.13.3 Member Function Documentation

7.13.3.1 getDataFromFile()

```
void MyWindow::getDataFromFile (
             DATA_STRUCTURES_TYPE type )  [protected]
```

Definition at line 123 of file updating.cpp.

```
{
    switch(type)
    {
        case DATA_STRUCTURES_TYPE::AVL:{
                                            std::vector<std::string> mem = FILEE::readFile(PATH::SAVING::AVL_);
                                            if(mem.empty()) return ;
                                            renderMutex.lock();
                                            inputbox->setText(1, mem[0]);
                                            renderMutex.unlock();
                                            break;
                                        }
        case DATA_STRUCTURES_TYPE::TRIE:{
                                             std::vector<std::string> mem = FILEE::readFile(PATH::SAVING::TRIE_);
                                             if(mem.empty()) return ;
                                             std::string total = "";
                                             for(int i = 0; i < mem.size(); i++)
                                             {
                                                 total += mem[i];
                                                 if(i != mem.size() - 1) total += " ";
                                             }
                                             renderMutex.lock();
                                             inputbox->setText(1, total);
                                             renderMutex.unlock();
                                             break;
                                         }
        case DATA_STRUCTURES_TYPE::GRAPH:{
                                              std::ifstream fin(PATH::SAVING::GRAPH_);

                                              renderMutex.lock();

                                              fin » ds->capacity;

                                              renderMutex.unlock();

                                              setInputBox("graph/edges");

                                              renderMutex.lock();

                                              for(int i = 0; i < ds->capacity; i++)
                                                  for(int j = 0; j < ds->capacity; j++)
                                                  {
                                                      int x;
                                                      fin » x;
                                                      inputbox->setText(i * ds->capacity + j + 1, NUMBER::intToString(x));
                                                  }

                                              renderMutex.unlock();
                                              fin.close();
                                              break;
                                          }
        case DATA_STRUCTURES_TYPE::BTREE_4TH:{
                                                  std::vector<std::string> mem = FILEE::readFile(PATH::SAVING::BTREE_4TH_);
                                                  break;
                                              }
        case DATA_STRUCTURES_TYPE::MIN_HEAP:{
                                                 std::vector<std::string> mem = FILEE::readFile(PATH::SAVING::MIN_HEAP_);
                                                 renderMutex.lock();
                                                 inputbox->setText(1, mem[0]);
                                                 renderMutex.unlock();
                                                 break;
                                             }
        case DATA_STRUCTURES_TYPE::MAX_HEAP:{
                                                 std::vector<std::string> mem = FILEE::readFile(PATH::SAVING::MAX_HEAP_);
```

```
                                                renderMutex.lock();
                                                inputbox->setText(1, mem[0]);
                                                renderMutex.unlock();
                                                break;
                                            }
        case DATA_STRUCTURES_TYPE::HASH_TABLE:{
                                                  std::vector<std::string> mem =
    FILEE::readFile(PATH::SAVING::HASH_TABLE_);
                                                  renderMutex.lock();
                                                  inputbox->setText(1, mem[0]);
                                                  inputbox->setText(2, mem[1]);
                                                  renderMutex.unlock();
                                                  break;
                                              }
        case DATA_STRUCTURES_TYPE::NONE:{
                                            break;
                                        }
    }

}
```

7.13.3.2 getDataFromRandom()

```
void MyWindow::getDataFromRandom (
              DATA_STRUCTURES_TYPE type )  [protected]
```

Definition at line 207 of file updating.cpp.

```
{
    switch(type)
    {
        case DATA_STRUCTURES_TYPE::AVL:{
                                           int n = RANDOM::getInt(1, 32);
                                           std::string mem = RANDOM::getInt(n, 1, 999);
                                           renderMutex.lock();
                                           inputbox->setText(1, mem);
                                           renderMutex.unlock();
                                           break;
                                       }
        case DATA_STRUCTURES_TYPE::TRIE:{
                                            std::vector<std::string> mem =
    FILEE::readFile(PATH::SAVING::TRIE_);
                                            break;
                                        }
        case DATA_STRUCTURES_TYPE::GRAPH:{
                                             std::vector<std::string> mem =
    FILEE::readFile(PATH::SAVING::GRAPH_);
                                             break;
                                         }
        case DATA_STRUCTURES_TYPE::BTREE_4TH:{
                                                 std::vector<std::string> mem =
    FILEE::readFile(PATH::SAVING::BTREE_4TH_);
                                                 break;
                                             }
        case DATA_STRUCTURES_TYPE::MIN_HEAP:{
                                                std::vector<std::string> mem =
    FILEE::readFile(PATH::SAVING::MIN_HEAP_);
                                                break;
                                            }
        case DATA_STRUCTURES_TYPE::MAX_HEAP:{
                                                std::vector<std::string> mem =
    FILEE::readFile(PATH::SAVING::MAX_HEAP_);
                                                break;
                                            }
        case DATA_STRUCTURES_TYPE::HASH_TABLE:{
                                                  std::vector<std::string> mem =
    FILEE::readFile(PATH::SAVING::HASH_TABLE_);
                                                  break;
                                              }
        case DATA_STRUCTURES_TYPE::NONE:{
                                            break;
                                        }
    }
}
```

7.13.3.3 getEvent()

```
void MyWindow::getEvent ( )  [protected]
```

Definition at line 43 of file event.cpp.

```
{
    SDL_Event event;
    while(isOpen())
    {
        while(SDL_PollEvent(&event))
        {
            switch(event.type)
            {
                case SDL_QUIT:
                {
                    event_mutex.lock();

                    event_pool.push(event);

                    event_mutex.unlock();
                    break;
                }
                default:
                {

                    bool isds = false;
                    bool isdisplay = false;
                    bool isinputbox = false;

                    event_mutex.lock();

                    renderMutex.lock();

                    if(ds != nullptr && ds->isReceiveEvent(event))
                        isds = true;

                    if(current_display->isReceiveEvent(event))
                        isdisplay = true;

                    if (inputbox != nullptr && inputbox->isReceiveEvent(event))
                        isinputbox = true;

                    renderMutex.unlock();

                    if(isds || isdisplay || isinputbox)
                        event_pool.push(event);

                    event_mutex.unlock();

                    EVcond.notify_one();
                    std::this_thread::sleep_for(std::chrono::milliseconds(10));
                    break;
                }
            }
        }
    }
}
```

7.13.3.4 initSDL2()

```
void MyWindow::initSDL2 ( )  [protected]
```

Definition at line 5 of file constructor.cpp.

```
{
    SDL_Init(SDL_INIT_VIDEO | SDL_INIT_AUDIO);

    window = SDL_CreateWindow(
         "Dr Duck",
         SDL_WINDOWPOS_CENTERED,
         SDL_WINDOWPOS_CENTERED,
         WIDTH,
         HEIGHT,
         SDL_WINDOW_SHOWN
     );

     render = SDL_CreateRenderer(
        window,
        -1,
        SDL_RENDERER_ACCELERATED
    );

    IMG_Init(IMG_INIT_PNG | IMG_INIT_JPG);
    TTF_Init();
}
```

7.13.3.5 interacting()

```
void MyWindow::interacting ( )  [protected]
```

Definition at line 4 of file event.cpp.

```
{
    SDL_Event event;
    std::unique_lock<std::mutex> ul(event_mutex);
    while(isOpen())
    {
        EVcond.wait(ul, [&](){return !event_pool.empty();});
        event = event_pool.front();
        event_pool.pop();

        switch(event.type)
        {
            case SDL_QUIT:
                status_mutex.lock();
                status = WINDOW_STATUS::IS_CLOSED;
                status_mutex.unlock();
                step_cond.notify_all();
                break;
            default:
                Button* but = nullptr;
                renderMutex.lock();

                if(but == nullptr && inputbox != nullptr && inputbox->isReceiveEvent(event))
                    but = inputbox->react(event);

                if(ds != nullptr && ds->isReceiveEvent(event) && but == nullptr)
                    but = ds->react(event);

                if(but == nullptr && but == nullptr && current_display->isReceiveEvent(event))
                    but = current_display->react(event);

                renderMutex.unlock();

                react(but);
                break;
        }
        std::this_thread::sleep_for(std::chrono::milliseconds(5));
    }
}
```

7.13.3.6 isClosed()

```
bool MyWindow::isClosed ( )
```

Definition at line 3 of file status.cpp.

```
{
    std::lock_guard<std::mutex> lg(status_mutex);
    return status == WINDOW_STATUS::IS_CLOSED;
}
```

7.13.3.7 isOpen()

```
bool MyWindow::isOpen ( )
```

Definition at line 9 of file status.cpp.

```
{
    std::lock_guard<std::mutex> lg(status_mutex);
    return status == WINDOW_STATUS::IS_OPEN;
}
```

7.13.3.8 react()

```
void MyWindow::react (
            Button * but )  [protected]
```

Definition at line 249 of file updating.cpp.

```
{
    if(but == nullptr) return ;
    switch (but->getAction())
    {
        case BUTTON_ACTION::CHANGE_SCREEN:
            setDisplay(but->getNextScreen());
            if(but->getDataType() != DATA_STRUCTURES_TYPE::NONE)
                setDataType(but->getDataType());
            break;

        case BUTTON_ACTION::INIT:
            setInputBox(ds->getName() + "/init");
            break;

        case BUTTON_ACTION::SETTING:
            setInputBox("setting");
            break;

        case BUTTON_ACTION::INSERT:
            setInputBox(ds->getName() + "/insert");
            break;

        case BUTTON_ACTION::DELETE:
            setInputBox(ds->getName() + "/remove");
            break;

        case BUTTON_ACTION::SEARCH:
            setInputBox(ds->getName() + "/search");
            break;

        case BUTTON_ACTION::SIZE:
            setInputBox(ds->getName() + "/size");
            break;

        case BUTTON_ACTION::TOP:
            setInputBox(ds->getName() + "/top");
            break;

        case BUTTON_ACTION::CONNECTED_COMPONENTS:
            setInputBox(ds->getName() + "/scc");
            break;

        case BUTTON_ACTION::DIJKSTRA:
            setInputBox(ds->getName() + "/dijkstra");
            break;

        case BUTTON_ACTION::MST:
            setInputBox(ds->getName() + "/mst");
            break;

        case BUTTON_ACTION::GO_BACK:

            //ds->goBack();
            break;

        case BUTTON_ACTION::GO_NEXT:

            //ds->goNext();
            break;

        case BUTTON_ACTION::GO_ON:

            //ds->goOn();
            break;

        case BUTTON_ACTION::GO_OFF:

            //ds->goOff();
            break;

        case BUTTON_ACTION::SPEED_UP:

            //ds->speedUp();
            break;

        case BUTTON_ACTION::SLOW_DOWN:

            //ds->slowDown();
            break;

        case BUTTON_ACTION::DONE:

            step_mutex.lock();
            isQueuingStep = true;
            step_mutex.unlock();
            step_cond.notify_one();

            break;

        case BUTTON_ACTION::EDGES:

            int n;
            renderMutex.lock();

            n = NUMBER::stringToInt(inputbox->getText(1));
            n = std::min(n, 9);
            n = std::max(n, 2);
            if(ds != nullptr) ds->capacity = n;

            renderMutex.unlock();
            setInputBox("graph/edges");
            break;

        case BUTTON_ACTION::CLOSE:

            renderMutex.lock();
            ds->closeScript();
            renderMutex.unlock();
            break;

        case BUTTON_ACTION::RANDOM:{
                                                int n = RANDOM::getInt(1, 32);
                                                std::string mem = RANDOM::getInt(n, 1, 999);

                                                renderMutex.lock();
                                                inputbox->setText(1, mem);

                                                renderMutex.unlock();
                                                break;
                                            }
        case BUTTON_ACTION::RANDOM2: {

                                                  renderMutex.lock();
                                                  inputbox->setText(1, RANDOM::getInt(1, 1, 999));
                                                  renderMutex.unlock();
                                                  break;
                                              }
        case BUTTON_ACTION::RANDOM3:{
                                                 int n = RANDOM::getInt(1, 16);
                                                 int m = RANDOM::getInt(1, 16);
                                                 char upperbound = RANDOM::getInt(97 + 5, 97 + 25);
                                                 std::string mem;
                                                 mem = RANDOM::getString(m, 'a', upperbound);

                                                 for(int i = 1; i < n; i++)
                                                     mem += " " + RANDOM::getString(m, 'a', upperbound);

                                                 renderMutex.lock();
                                                 inputbox->setText(1, mem);
                                                 renderMutex.unlock();
                                                 break;
                                             }
        case BUTTON_ACTION::RANDOM4:
                                             {
                                                 int m = RANDOM::getInt(1, 16);
                                                 renderMutex.lock();
                                            inputbox->setText(1, RANDOM::getString(m, 'a', 'z'));
                                            renderMutex.unlock();
                                            break;
                                        }
            case BUTTON_ACTION::RANDOM5: {
                                            int n = RANDOM::getInt(1, 64);
                                            std::string mem = RANDOM::getInt(n, 1, 999);

                                            renderMutex.lock();
                                            inputbox->setText(2, mem);
                                            renderMutex.unlock();
                                            break;
                                        }
            case BUTTON_ACTION::RANDOM6: {
                                            renderMutex.lock();
                                            for(int i = 0; i < ds->capacity; i++)
                                                for(int j = 0; j < ds->capacity; j++)
                                                {
                                                    if(i == j) continue;
                                                    if(RANDOM::getInt(1, 100) <= 30)
                                                        inputbox->setText(i * ds->capacity + j + 1,
        RANDOM::getInt(1, 1, 99));
                                                    else
                                                        inputbox->setText(i * ds->capacity + j + 1, "0");
                                                }
                                            renderMutex.unlock();
                                            break;
                                        }
            case BUTTON_ACTION::RANDOM7: {
                                            renderMutex.lock();
                                            inputbox->setText(1, NUMBER::intToString(RANDOM::getInt(0,
        ds->capacity - 1)));
                                            renderMutex.unlock();
                                            break;
                                        }
            case BUTTON_ACTION::RANDOM8: {
                                            renderMutex.lock();
                                            inputbox->setText(2, NUMBER::intToString(RANDOM::getInt(0,
        ds->capacity - 1)));
                                            renderMutex.unlock();
                                            break;
                                        }

            case BUTTON_ACTION::FILE :{
                                        getDataFromFile(ds->getDataType());
                                        break;
                                    }

            default:
                                    break;
    }
}
```

7.13.3.9 rendering()

void MyWindow::rendering ( ) [protected]

Definition at line 4 of file rendering.cpp.

```
{
    while(isOpen())
    {
        if(!renderMutex.try_lock())
        {
            std::this_thread::sleep_for(std::chrono::milliseconds(5));
            continue;
         }

         SDL_RenderClear(render);

         current_display->rendering();

         if(ds != nullptr) ds->rendering();

         SDL_RenderSetViewport(render, &viewport);

         if(inputbox != nullptr) inputbox->rendering();

         SDL_RenderPresent(render);

        renderMutex.unlock();
        std::this_thread::sleep_for(std::chrono::milliseconds(5));
    }
}
```

7.13.3.10 run()

```
void MyWindow::run ( )
```

Definition at line 4 of file running.cpp.

```
{
    status = WINDOW_STATUS::IS_OPEN;
    thread_pool.push_back(std::thread(&MyWindow::rendering, this));
    thread_pool.push_back(std::thread(&MyWindow::getEvent, this));
    thread_pool.push_back(std::thread(&MyWindow::runOperator, this));

    interacting();

    for(auto& thread : thread_pool)
    {
        thread.join();
    }
}
```

7.13.3.11 runOperator()

```
void MyWindow::runOperator ( )  [protected]
```

Definition at line 60 of file updating.cpp.

```
{
    while(isOpen())
    {
        std::unique_lock<std::mutex> lock(step_mutex);
        step_cond.wait(lock, [&](){return isClosed() || isQueuingStep;});
        if(isClosed()) break;

        isQueuingStep = false;

        InputBox* temp;

        renderMutex.lock();
        if(inputbox == nullptr)
        {
            renderMutex.unlock();
            continue;
        }

        temp = inputbox;
        renderMutex.unlock();

        setInputBox("nullptr");

        renderMutex.lock();

        switch(temp->getOperator())
        {
            case DATA_STRUCTURES_OPERATOR::INIT:
                {
                    ds->init(temp);
                    break;
                }
            case DATA_STRUCTURES_OPERATOR::INSERT:
                ds->insert(temp);
                break;
            case DATA_STRUCTURES_OPERATOR::DELETE:
                ds->remove(temp);
                break;
            case DATA_STRUCTURES_OPERATOR::SEARCH:
                ds->search(temp);
                break;
            case DATA_STRUCTURES_OPERATOR::TOP:
                ds->top();
                break;
            case DATA_STRUCTURES_OPERATOR::SIZE:
                ds->size();
            case DATA_STRUCTURES_OPERATOR::SETTING:
                ds->setting(temp);
            case DATA_STRUCTURES_OPERATOR::DIJKSTRA:
                ds->dijkstra(temp);
                break;
            case DATA_STRUCTURES_OPERATOR::MST:
                ds->mst();
                break;
            case DATA_STRUCTURES_OPERATOR::SCC:
                ds->scc();
                break;
        }
        renderMutex.unlock();
    }
}
```

7.13.3.12 setDataType()

```
void MyWindow::setDataType (
              DATA_STRUCTURES_TYPE type )   [protected]
```

Definition at line 47 of file updating.cpp.

```
{
    renderMutex.lock();
    if(ds_pool.find(type) == ds_pool.end())
    {
        ds_pool[type] = new DataStructures(render, myfont, renderMutex);
        ds_pool[type]->setDataType(type);
    }
    ds = ds_pool[type];

    renderMutex.unlock();
}
```

7.13.3.13 setDisplay()

```
void MyWindow::setDisplay (
              std::string name )   [protected]
```

Definition at line 4 of file updating.cpp.

```
{
    setInputBox("nullptr");
    renderMutex.lock();

    ds = nullptr;
    if(display_pool.find(name) == display_pool.end())
    {
        display_pool[name] = new Display(render, viewport);
        display_pool[name]->linking(name);
    }
    current_display = display_pool[name];

    renderMutex.unlock();
}
```

7.13.3.14 setInputBox()

```
void MyWindow::setInputBox (
             std::string name )  [protected]
```

Definition at line 20 of file updating.cpp.

```
{
    renderMutex.lock();
    if(name == "graph/edges")
    {
        auto i = inputbox_pool.find("graph/edges");
        if(i != inputbox_pool.end())
        {
            auto t = i->second;
            inputbox_pool.erase(i);
            delete t;
        }
        {
            inputbox_pool["graph/edges"] = new InputBox(render, myfont);
            inputbox_pool["graph/edges"]->setDuplicate(ds->capacity, ds->capacity);
            inputbox_pool["graph/edges"]->linking("graph/edges");
        }
    }
    if(inputbox_pool.find(name) == inputbox_pool.end())
    {
        inputbox_pool[name] = new InputBox(render, myfont);
        inputbox_pool[name]->linking(name);
    }
    inputbox = inputbox_pool[name];
    renderMutex.unlock();
}
```

The documentation for this class was generated from the following files:

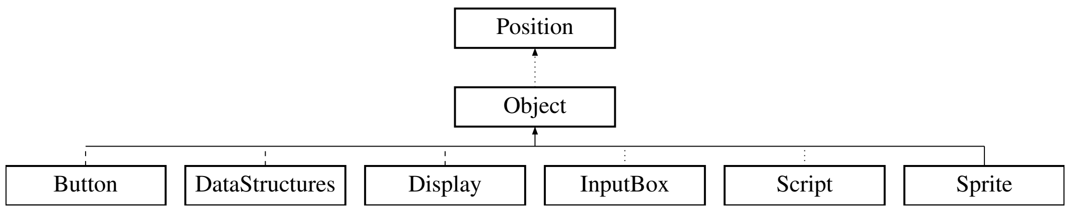
- include/window.hpp
- src/window/constructor.cpp
- src/window/destructor.cpp
- src/window/event.cpp
- src/window/rendering.cpp
- src/window/running.cpp
- src/window/status.cpp
- src/window/updating.cpp

7.14 Object Class Reference

Class that represent shape, image from files, text. Smallest drawable unit.

```
#include <object.hpp>
```

Inheritance diagram for Object:



Public Member Functions

- Object (SDL_Renderer ∗&r)
- ∼Object ()
- const SDL_Rect ∗ getCrop ()
- void cropping (int x, int y, int w, int h)
- void cropping (SDL_Rect c)
- void cropping (const json &mem)
- void noCropping ()
- const SDL_Rect ∗ getLocation ()
- int getX ()
- int getY ()
- int getW ()
- int getH ()
- virtual void locating (int x, int y, int w, int h)
- virtual void locating (SDL_Rect l)
- virtual void locating (const json &mem)
- virtual void locatingX (int x)
- virtual void locatingY (int y)
- virtual void locatingW (int w)
- virtual void locatingH (int h)
- virtual void moveX (int delta)
- virtual void moveY (int delta)
- virtual void zoomW (int delta)
- virtual void zoomH (int delta)
- virtual void zoom (double delta)
- virtual void zoomInMiddle (double delta)
- void fitTheTexture ()
- const SDL_Color ∗ getColor ()
- void coloring (int r, int g, int b, int a)
- void coloring (SDL_Color c)
- void coloring (const json &mem)
- void textureFromFile (std::string dir)
- void changeToCircle ()
- void changeToCircle (SDL_Point c)
- void changeToCircle (int x, int y)
- void changeToCircle (SDL_Point c, int r)
- void changeToCircle (int x, int y, int r)
- void changeToRectangle ()
- void setShape (const json &mem)
- bool isLieInside (int x, int y)
- bool isLieInside (SDL_Point p)
- bool isLieInside (SDL_Rect r)
- bool isLieInside (int x, int y, int w, int h)
- virtual void rendering ()
- void show ()
- void hide ()
- bool isVisible ()
- void importFromJson (const json &mem)
- void linking (std::string n)
- void setFont (TTF_Font ∗f)
- void setText (std::string t)
- void addText (std::string t)
- void addCharacter (char c)
- void removeCharacter ()
- void removeCharacter (int n)
- std::string getText ()
- int getSize ()

Protected Member Functions

- void fillWithColor ()
- void fillCircleByColor ()
- void fillRectangleByColor ()
- void textToTexture ()

7.14.1 Detailed Description

Class that represent shape, image from files, text. Smallest drawable unit.

Definition at line 28 of file object.hpp.

7.14.2 Constructor & Destructor Documentation

7.14.2.1 Object()

```
Object::Object (
              SDL_Renderer *& r )
```

Definition at line 7 of file constructor.cpp.

```
{
    render = r;
     location = nullptr;
     crop = nullptr;
     color = nullptr;
     texture = nullptr;
     font = nullptr;
     locating(0, 0, 0, 0);
     shapeType = SHAPE::NONE;
     visible = false;
 }
```

7.14.2.2 ∼Object()

```
Object::∼Object ( )
```

Definition at line 3 of file destructor.cpp.

```
{
    render = nullptr;
    if(crop != nullptr) delete crop;
    if(location != nullptr) delete location;
    if(color != nullptr) delete color;
    if(texture != nullptr) SDL_DestroyTexture(texture);
     texture = nullptr;

     crop = nullptr;
     location = nullptr;
     color = nullptr;
     texture = nullptr;
 }
```

7.14.3 Member Function Documentation

7.14.3.1 addCharacter()

```
void Object::addCharacter (
             char c )
```

Definition at line 22 of file font.cpp.

```
{
    text += c;
    textToTexture();
}
```

7.14.3.2 addText()

```
void Object::addText (
             std::string t )
```

Definition at line 16 of file font.cpp.

```
{
    text += t;
    textToTexture();
}
```

7.14.3.3 changeToCircle() [1/5]

```
void Object::changeToCircle ( )
```

Definition at line 5 of file shape.cpp.

```
{
    shapeType = SHAPE::CIRCLE;
    radius = std::min(getW(), getH()) / 2;

    center.x = getX() + getW() / 2;
    center.y = getY() + getH() / 2;
    fillCircleByColor();
}
```

7.14.3.4 changeToCircle() [2/5]

```
void Object::changeToCircle (
             int x,
             int y )
```

Definition at line 24 of file shape.cpp.

```
{
    changeToCircle({x, y});
}
```

7.14.3.5 changeToCircle() [3/5]

```
void Object::changeToCircle (
            int x,
            int y,
            int r )
```

Definition at line 37 of file shape.cpp.

```
{
    shapeType = SHAPE::CIRCLE;
    radius = r;
    center.x = x;
    center.y = y;
    fillCircleByColor();
}
```

7.14.3.6 changeToCircle() [4/5]

```
void Object::changeToCircle (
            SDL_Point c )
```

Definition at line 15 of file shape.cpp.

```
{
    shapeType = SHAPE::CIRCLE;
    center = c;
    radius = std::min(getW() - c.x, c.x - getX());
    radius = std::min(radius, std::min(getH() - c.y, c.y - getY()));
    fillCircleByColor();
}
```

7.14.3.7 changeToCircle() [5/5]

```
void Object::changeToCircle (
            SDL_Point c,
            int r )
```

Definition at line 29 of file shape.cpp.

```
{
    shapeType = SHAPE::CIRCLE;
    radius = r;
    center = c;
    fillCircleByColor();
}
```

7.14.3.8 changeToRectangle()

```
void Object::changeToRectangle ( )
```

Definition at line 46 of file shape.cpp.

```
{
    shapeType = SHAPE::RECTANGLE;
    fillRectangleByColor();
}
```

7.14.3.9 coloring() [1/3]

```
void Object::coloring (
              const json & mem )
```

Definition at line 30 of file coloring.cpp.

```
{
    if(mem.contains("r") && mem.contains("g") && mem.contains("b"))
    {
        if(mem.contains("a")) coloring(mem["r"], mem["g"], mem["b"], mem["a"]);
        else coloring(mem["r"], mem["g"], mem["b"], 255);
    }
}
```

7.14.3.10 coloring() [2/3]

```
void Object::coloring (
              int r,
              int g,
              int b,
              int a )
```

Definition at line 8 of file coloring.cpp.

```
{
    if(color == nullptr) color = new SDL_Color;
    color->r = r;
    color->g = g;
    color->b = b;
    color->a = a;

    fillWithColor();
}
```

7.14.3.11 coloring() [3/3]

```
void Object::coloring (
              SDL_Color c )
```

Definition at line 19 of file coloring.cpp.

```
{
    if(color == nullptr) color = new SDL_Color;
    color->r = c.r;
    color->g = c.g;
    color->b = c.b;
    color->a = c.a;

    fillWithColor();
}
```

7.14.3.12 cropping() [1/3]

```
void Object::cropping (
              const json & mem )
```

Definition at line 26 of file cropping.cpp.

```
{
    if(mem.contains("x") && mem.contains("y") && mem.contains("w") && mem.contains("h"))
        cropping(mem["x"], mem["y"], mem["w"], mem["h"]);
}
```

7.14.3.13 cropping() [2/3]

```
void Object::cropping (
               int x,
               int y,
               int w,
               int h )
```

Definition at line 8 of file cropping.cpp.

```
{
    if(crop == nullptr) crop = new SDL_Rect;
    crop->x = x;
    crop->y = y;
    crop->w = w;
    crop->h = h;
}
```

7.14.3.14 cropping() [3/3]

```
void Object::cropping (
               SDL_Rect c )
```

Definition at line 17 of file cropping.cpp.

```
{
    if(crop == nullptr) crop = new SDL_Rect;
    crop->x = c.x;
    crop->y = c.y;
    crop->w = c.w;
    crop->h = c.h;
}
```

7.14.3.15 fillCircleByColor()

```
void Object::fillCircleByColor ( )  [protected]
```

Definition at line 91 of file shape.cpp.

```
{
    if(location == nullptr) locating(0, 0, 0, 0);

    if(texture != nullptr) SDL_DestroyTexture(texture);
    texture = nullptr;

    Uint32 rmask, gmask, bmask, amask;
    Uint32 pixelColor;
#if SDL_BYTEORDER == SDL_BIG_ENDIAN
     rmask = 0xff000000;
     gmask = 0x00ff0000;
     bmask = 0x0000ff00;
     amask = 0x000000ff;
     pixelColor = (color->r « 24) | (color->g « 16) | (color->b « 8) | color->a;
#else
     rmask = 0x000000ff;
     gmask = 0x0000ff00;
     bmask = 0x00ff0000;
     amask = 0xff000000;
     pixelColor = (color->a « 24) | (color->b « 16) | (color->g « 8) | color->r;
#endif

     SDL_Surface *surf = SDL_CreateRGBSurface(0, getW(), getH(), 32, rmask, gmask, bmask, amask);
     SDL_SetSurfaceBlendMode(surf, SDL_BLENDMODE_BLEND);

     texture = SDL_CreateTextureFromSurface(render, surf);
     SDL_FreeSurface(surf);

     Uint32 *pixels = new Uint32[getW() * getH()];
    memset(pixels, 0, getW() * getH() * sizeof(Uint32));

    SDL_Point p = {getW() / 2, getH() / 2};
    center = p;

    if(radius > std::min(getW(), getH()) / 2) radius = std::min(getW(), getH()) / 2;

    for(int i = p.x - radius; i <= p.x + radius; i++)
        for(int j = p.y - radius; j <= p.y + radius; j++)
            if((i - p.x) * (i - p.x) + (j - p.y) * (j - p.y) <= radius * radius)
            {
                int index = i * getW() + j;
                if(index < 0 || index >= getW() * getH()) continue;
                pixels[index] = pixelColor;
            }

    SDL_UpdateTexture(texture, nullptr, pixels, getW() * sizeof(Uint32));
    delete[] pixels;
}
```

7.14.3.16 fillRectangleByColor()

`void Object::fillRectangleByColor ( )  [protected]`

Definition at line 74 of file shape.cpp.

```
{
    if(location == nullptr) locating(0, 0, 0, 0);

    if(texture != nullptr) SDL_DestroyTexture(texture);
    texture = nullptr;

    SDL_Surface* surf = SDL_CreateRGBSurfaceWithFormat(0, getW(), getH(), 32, SDL_PIXELFORMAT_RGBA32);

    SDL_SetSurfaceBlendMode(surf, SDL_BLENDMODE_BLEND);

    SDL_FillRect(surf, nullptr, SDL_MapRGBA(surf->format, color->r, color->g, color->b, color->a));

    texture = SDL_CreateTextureFromSurface(render, surf);

    SDL_FreeSurface(surf);
}
```

7.14.3.17 fillWithColor()

`void Object::fillWithColor ( )  [protected]`

Definition at line 39 of file coloring.cpp.

```
{
    if(shapeType == SHAPE::NONE) return fillRectangleByColor();
    if(shapeType == SHAPE::RECTANGLE) return fillRectangleByColor();
    if(shapeType == SHAPE::CIRCLE) return fillCircleByColor();
}
```

7.14.3.18 fitTheTexture()

`void Object::fitTheTexture ( )`

Definition at line 140 of file locating.cpp.

```
{
    if(texture == nullptr) return;
    SDL_QueryTexture(texture, nullptr, nullptr, &location->w, &location->h);
}
```

7.14.3.19 getColor()

```
const SDL_Color * Object::getColor ( )
```

Definition at line 3 of file coloring.cpp.

```
{
    return color;
}
```

7.14.3.20 getCrop()

```
const SDL_Rect * Object::getCrop ( )
```

Definition at line 3 of file cropping.cpp.

```
{
    return crop;
}
```

7.14.3.21 getH()

```
int Object::getH ( )
```

Definition at line 47 of file locating.cpp.

```
{
    return location->h;
}
```

7.14.3.22 getLocation()

```
const SDL_Rect * Object::getLocation ( )
```

Definition at line 27 of file locating.cpp.

```
{
    return location;
}
```

7.14.3.23 getSize()

```
int Object::getSize ( )
```

Definition at line 68 of file font.cpp.

```
{
    return text.size();
}
```

7.14.3.24 getText()

```
std::string Object::getText ( )
```

Definition at line 63 of file font.cpp.

```
{
    return text;
}
```

7.14.3.25 getW()

```
int Object::getW ( )
```

Definition at line 42 of file locating.cpp.

```
{
    return location->w;
}
```

7.14.3.26 getX()

```
int Object::getX ( )
```

Definition at line 32 of file locating.cpp.

```
{
    return location->x;
}
```

7.14.3.27 getY()

```
int Object::getY ( )
```

Definition at line 37 of file locating.cpp.

```
{
    return location->y;
}
```

7.14.3.28 hide()

```
void Object::hide ( )
```

Definition at line 8 of file visible.cpp.

```
{
    visible = false;
}
```

7.14.3.29 importFromJson()

```
void Object::importFromJson (
                const json & mem )
```

Definition at line 21 of file constructor.cpp.

```
{
    if(mem.contains("location"))
        locating(mem["location"]);

    if(mem.contains("crop"))
        cropping(mem["crop"]);

    if(mem.contains("color"))
        coloring(mem["color"]);

    if(mem.contains("shape"))
        setShape(mem["shape"]);

    if(mem.contains("visible"))
        visible = mem["visible"];

    if(mem.contains("image"))
        textureFromFile(PATH::ASSETS::GRAPHICS_ + mem["image"].get<std::string>());
    return ;
}
```

7.14.3.30 isLieInside() [1/4]

```
bool Object::isLieInside (
                int x,
                int y )
```

Definition at line 3 of file locating.cpp.

```
{
    if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
    return (x >= location->x && x < location->x + location->w && y >= location->y && y < location->y +
     location->h);
}
```

7.14.3.31 isLieInside() [2/4]

```
bool Object::isLieInside (
                int x,
                int y,
                int w,
                int h )
```

Definition at line 21 of file locating.cpp.

```
{
    if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
    return (x >= location->x && x + w <= location->x + location->w && y >= location->y && y + h <=
    location->y + location->h);
}
```

7.14.3.32 isLieInside() [3/4]

```
bool Object::isLieInside (
            SDL_Point p )
```

Definition at line 9 of file locating.cpp.

```
{
    if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
    return (p.x >= location->x && p.x < location->x + location->w && p.y >= location->y && p.y <
    location->y + location->h);
}
```

7.14.3.33 isLieInside() [4/4]

```
bool Object::isLieInside (
            SDL_Rect r )
```

Definition at line 15 of file locating.cpp.

```
{
    if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
    return (r.x >= location->x && r.x + r.w <= location->x + location->w && r.y >= location->y && r.y +
    r.h <= location->y + location->h);
}
```

7.14.3.34 isVisible()

```
bool Object::isVisible ( )
```

Definition at line 13 of file visible.cpp.

```
{
    return visible;
}
```

7.14.3.35 linking()

```
void Object::linking (
            std::string n )
```

Definition at line 43 of file constructor.cpp.

```
{
    name = n;
    json* mem = JSON::readFile(PATH::ATB::OBJECT_ + name + ".json");

    importFromJson(*mem);

    delete mem;
}
```

7.14.3.36 locating() [1/3]

```
void Object::locating (
             const json & mem )  [virtual]
```

Definition at line 70 of file locating.cpp.

```
{
    if(mem.contains("x") && mem.contains("y") && mem.contains("w") && mem.contains("h"))
        locating(mem["x"], mem["y"], mem["w"], mem["h"]);
}
```

7.14.3.37 locating() [2/3]

```
void Object::locating (
             int x,
             int y,
             int w,
             int h )  [virtual]
```

Reimplemented in Sprite.

Definition at line 52 of file locating.cpp.

```
{
    if(location == nullptr) location = new SDL_Rect;
    location->x = x;
    location->y = y;
    location->w = w;
    location->h = h;
}
```

7.14.3.38 locating() [3/3]

```
void Object::locating (
             SDL_Rect l )  [virtual]
```

Reimplemented in Sprite.

Definition at line 61 of file locating.cpp.

```
{
    if(location == nullptr) location = new SDL_Rect;
    location->x = l.x;
    location->y = l.y;
    location->w = l.w;
    location->h = l.h;
}
```

7.14.3.39 locatingH()

```
void Object::locatingH (
             int h )  [virtual]
```

Definition at line 94 of file locating.cpp.

```
{
    if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
    location->h = h;
}
```

7.14.3.40 locatingW()

```
void Object::locatingW (
             int w )  [virtual]
```

Definition at line 88 of file locating.cpp.

```
{
    if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
    location->w = w;
}
```

7.14.3.41 locatingX()

```
void Object::locatingX (
             int x )  [virtual]
```

Reimplemented in Sprite.

Definition at line 76 of file locating.cpp.

```
{
    if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
    location->x = x;
}
```

7.14.3.42 locatingY()

```
void Object::locatingY (
             int y )  [virtual]
```

Reimplemented in Sprite.

Definition at line 82 of file locating.cpp.

```
{
    if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
    location->y = y;
}
```

7.14.3.43 moveX()

```
void Object::moveX (
             int delta )  [virtual]
```

Reimplemented in Sprite.

Definition at line 100 of file locating.cpp.

```
{
    if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
    location->x += delta;
}
```

7.14.3.44 moveY()

```
void Object::moveY (
             int delta )  [virtual]
```

Reimplemented in Sprite.

Definition at line 106 of file locating.cpp.

```
{
    if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
    location->y += delta;
}
```

7.14.3.45 noCropping()

```
void Object::noCropping ( )
```

Definition at line 32 of file cropping.cpp.

```
{
    if(crop != nullptr) delete crop;
    crop = nullptr;
}
```

7.14.3.46 removeCharacter() [1/2]

```
void Object::removeCharacter ( )
```

Definition at line 28 of file font.cpp.

```
{
    if (text.size() > 0)
        text.pop_back();
    textToTexture();
}
```

7.14.3.47 removeCharacter() [2/2]

```
void Object::removeCharacter (
             int n )
```

Definition at line 35 of file font.cpp.

```
{
    if(n == 0) return ;
    if(text.size() <= n) text.clear();
    else text.erase(text.end() - n, text.end());
    textToTexture();
}
```

7.14.3.48 rendering()

```
void Object::rendering ( )  [virtual]
```

Reimplemented in Sprite, Script, InputBox, Display, DataStructures, and Button.

Definition at line 3 of file rendering.cpp.

```
{
    if(!visible) return ;
    if(texture == nullptr) return ;
    SDL_RenderCopy(render, texture, crop, location);
}
```

7.14.3.49 setFont()

```
void Object::setFont (
              TTF_Font * f )
```

Definition at line 4 of file font.cpp.

```
{
    font = f;
    textToTexture();
}
```

7.14.3.50 setShape()

```
void Object::setShape (
              const json & mem )
```

Definition at line 52 of file shape.cpp.

```
{
    if(mem["type"].get<std::string>() == "CIRCLE")
    {
        if(mem.contains("center"))
        {
            if(mem.contains("radius"))
                changeToCircle(mem["center"]["x"], mem["center"]["y"], mem["radius"]);
            else changeToCircle(mem["center"]["x"], mem["center"]["y"]);
        }else changeToCircle();

        return ;
    }

    if(mem["type"].get<std::string>() == "NONE" || mem["type"].get<std::string>() == "RECTANGLE")
    {
        changeToRectangle();
        return ;
    }

}
```

7.14.3.51 setText()

```
void Object::setText (
              std::string t )
```

Definition at line 10 of file font.cpp.

```
{
    text = t;
    textToTexture();
}
```

7.14.3.52 show()

```
void Object::show ( )
```

Definition at line 3 of file visible.cpp.

```
{
    visible = true;
}
```

7.14.3.53 textToTexture()

```
void Object::textToTexture ( )  [protected]
```

Definition at line 43 of file font.cpp.

```
{
    if(font == nullptr) return ;
    if(color == nullptr) return ;
    if(render == nullptr) return ;
    if(texture != nullptr)
    {
        SDL_DestroyTexture(texture);
    }
    texture = nullptr;

    SDL_Surface* surface = TTF_RenderText_Blended(font, text.c_str(), *color);

    if(surface == nullptr) return ;

    texture = SDL_CreateTextureFromSurface(render, surface);
    SDL_FreeSurface(surface);
    fitTheTexture();
}
```

7.14.3.54 textureFromFile()

```
void Object::textureFromFile (
              std::string dir )
```

Definition at line 4 of file external_storage.cpp.

```
{
    SDL_Surface *surface = IMG_Load(dir.c_str());

    texture = SDL_CreateTextureFromSurface(render, surface);
    SDL_FreeSurface(surface);
}
```

7.14.3.55 zoom()

```
void Object::zoom (
              double delta )  [virtual]
```

Definition at line 123 of file locating.cpp.

```
{
    if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
    location->w *= delta;
    location->h *= delta;
}
```

7.14.3.56 zoomH()

```
void Object::zoomH (
             int delta )  [virtual]
```

Definition at line 118 of file locating.cpp.

```
{
    if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
    location->h += delta;
}
```

7.14.3.57 zoomInMiddle()

```
void Object::zoomInMiddle (
             double delta )  [virtual]
```

Definition at line 130 of file locating.cpp.

```
{
    if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
    SDL_Point center = {location->x + location->w / 2, location->y + location->h / 2};
    location->w *= delta;
    location->h *= delta;
    location->x = center.x - location->w / 2;
    location->y = center.y - location->h / 2;
}
```

7.14.3.58 zoomW()

```
void Object::zoomW (
             int delta )  [virtual]
```

Definition at line 112 of file locating.cpp.

```
{
    if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
    location->w += delta;
}
```

The documentation for this class was generated from the following files:

- include/object.hpp
- src/object/coloring.cpp
- src/object/constructor.cpp
- src/object/cropping.cpp
- src/object/destructor.cpp
- src/object/external_storage.cpp
- src/object/font.cpp
- src/object/locating.cpp
- src/object/rendering.cpp
- src/object/shape.cpp
- src/object/visible.cpp

7.15 Point Struct Reference

Public Attributes

- double x
- double y

7.15.1 Detailed Description

Definition at line 3 of file rendering.cpp.

7.15.2 Member Data Documentation

7.15.2.1 x

```
double Point::x
```

Definition at line 5 of file rendering.cpp.

7.15.2.2 y

```
double Point::y
```

Definition at line 5 of file rendering.cpp.

The documentation for this struct was generated from the following file:

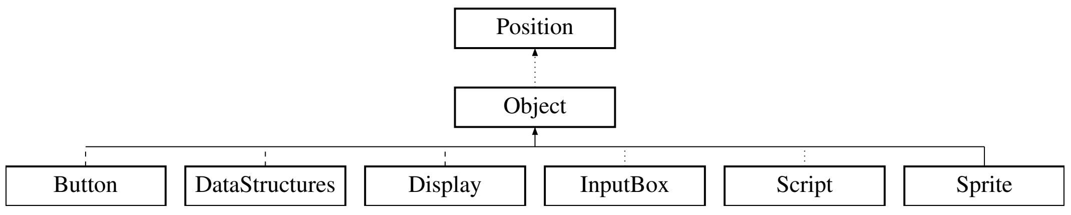
- src/graph/rendering.cpp

7.16 Position Struct Reference

location of an object in 2D coordinate

```
#include <object.hpp>
```

Inheritance diagram for Position:



Public Attributes

- SDL_Rect ∗ location
- SDL_Rect ∗ crop
- int radius
- SDL_Point center
- bool visible

7.16.1 Detailed Description

location of an object in 2D coordinate

Definition at line 16 of file object.hpp.

7.16.2 Member Data Documentation

7.16.2.1 center

```
SDL_Point Position::center
```

Definition at line 21 of file object.hpp.

7.16.2.2 crop

```
SDL_Rect* Position::crop
```

Definition at line 19 of file object.hpp.

7.16.2.3 location

```
SDL_Rect* Position::location
```

Definition at line 18 of file object.hpp.

7.16.2.4 radius

```
int Position::radius
```

Definition at line 20 of file object.hpp.

7.16.2.5 visible

```
bool Position::visible
```

Definition at line 22 of file object.hpp.

The documentation for this struct was generated from the following file:

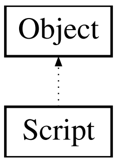
- include/object.hpp

7.17 Script Class Reference

Container that contains a pseudo-code.

```
#include <script.hpp>
```

Inheritance diagram for Script:



Public Member Functions

- Script (SDL_Renderer ∗render, TTF_Font ∗f)
- ∼Script ()
- void linking (std::string name)
- void highlight (int index)
- void unhighlight (int index)
- bool isReceiveEvent (SDL_Event &event)
- Button ∗ react (SDL_Event &event)
- void rendering ()

Protected Member Functions

- void initBackground (const json &mem)
- void initButtons (const json &mem)
- void importFromJson ()

Private Member Functions

- void importFromJson (const json &mem)

7.17.1 Detailed Description

Container that contains a pseudo-code.

Drawable

Definition at line 21 of file script.hpp.

7.17.2 Constructor & Destructor Documentation

7.17.2.1 Script()

```
Script::Script (
              SDL_Renderer * render,
              TTF_Font * f )
```

Definition at line 4 of file constructor.cpp.

```
                                               : Object(render)
{
    this->render = render;
    this->font = f;
    isMoving = false;
}
```

7.17.2.2 ∼Script()

```
Script::∼Script ( )
```

Definition at line 3 of file destructor.cpp.

```
{
    render = nullptr;
    font = nullptr;
    for(int i = 0; i < sprites.size(); i++)
        if(sprites[i] != nullptr) delete sprites[i];
    sprites.clear();
}
```

7.17.3 Member Function Documentation

7.17.3.1 highlight()

```
void Script::highlight (
              int index )
```

Definition at line 3 of file highlight.cpp.

```
{
    if(index < 0 || index >= sprites.size()) return ;
    sprites[index]->setTextBoxTransparent(180);
}
```

7.17.3.2 importFromJson()

```
void Script::importFromJson ( )  [protected]
```

Definition at line 17 of file constructor.cpp.

```
{
    json * mem = JSON::readFile(PATH::ATB::SCRIPT_ + name + ".json");
    if(mem == nullptr) return;

    if(mem->contains("background"))
        initBackground((*mem)["background"]);
    if(mem->contains("sprite"))
        spriteName = (*mem)["sprite"].get<std::string>();
    if(mem->contains("buttons"))
        initButtons((*mem)["buttons"]);

    if(mem->contains("script"))
    {
        auto lines = FILEE::readFile(PATH::ASSETS::SCRIPT_ + (*mem)["script"].get<std::string>());
        int j = 0;
        for(auto i : lines)
        {
            sprites.push_back(new Sprite(render));
            sprites.back()->setFont(font);
            sprites.back()->linking(spriteName);
            sprites.back()->setText(i);
            sprites.back()->locatingX(getX());
            sprites.back()->locatingY(getY() + j * 18);
            sprites.back()->aligning(HORIZONTAL_ALIGN::LEFT, VERTICAL_ALIGN::CENTER);
            j++;
        }
    }

}
```

7.17.3.3 initBackground()

```
void Script::initBackground (
            const json & mem )  [protected]
```

Definition at line 48 of file constructor.cpp.

```
{
    Object::importFromJson(mem);
}
```

7.17.3.4 initButtons()

```
void Script::initButtons (
            const json & mem )  [protected]
```

Definition at line 53 of file constructor.cpp.

```
{
    for(auto &i : mem)
    {
        Button* b = new Button(render);
        b->linking(i["name"].get<std::string>());
        buts.push_back(b);

    }
}
```

7.17.3.5 isReceiveEvent()

```
bool Script::isReceiveEvent (
              SDL_Event & event )
```

Definition at line 4 of file event.cpp.

```
{
    SDL_Rect viewport = {getX(), getY(), getW(), getH()};

    switch(e.type)
    {
        case SDL_MOUSEBUTTONDOWN:
            if(e.motion.x < viewport.x || viewport.x + viewport.w < e.motion.x) return false;
            if(e.motion.y < viewport.y || viewport.y + viewport.h < e.motion.y) return false;
            if(e.button.button == SDL_BUTTON_LEFT)
            {
                for(auto i : buts)
                {
                    if(i->isReceiveEvent(e))
                    {
                        return true;
                    }
                }
                return false;
            }
            return true;
            break;
        case SDL_MOUSEMOTION:
            if(isMoving) return true;
            for(auto i : buts)
            {
                if(i->isReceiveEvent(e))
                {
                    return true;
                }
            }
            return false;
            break;
        default:
            return false;
            break;
    }
}
```

7.17.3.6 linking()

```
void Script::linking (
              std::string name )
```

Definition at line 11 of file constructor.cpp.

```
{
    this->name = name;
    importFromJson();
}
```

7.17.3.7 react()

```
Button * Script::react (
              SDL_Event & event )
```

Definition at line 43 of file event.cpp.

```
{
    switch(e.type)
    {
        case SDL_MOUSEBUTTONDOWN:
            if(isMoving)
            {
                isMoving = false;
                int dx = e.motion.x - lastMousePressed.x;
                int dy = e.motion.y - lastMousePressed.y;
                moveX(dx);
                moveY(dy);
                for(auto i : buts)
                    i->move(dx, dy);
                for(auto i : sprites)
                {
                    i->moveX(dx);
                    i->moveY(dy);
                }

            }else
            {

                for(auto i : buts)
                {
                    if(i->isReceiveEvent(e))
                    {
                        return i;
                    }
                }
                isMoving = true;
                lastMousePressed.x = e.motion.x;
                lastMousePressed.y = e.motion.y;
            }
            return nullptr;
            break;
        case SDL_MOUSEMOTION:
            {
                if(!isMoving)
                {
                    for(auto i : buts)
                        i->isHover(e.motion.x, e.motion.y);
                    return nullptr;
                }
                int dx = e.motion.x - lastMousePressed.x;
                int dy = e.motion.y - lastMousePressed.y;
                lastMousePressed.x = e.motion.x;
                lastMousePressed.y = e.motion.y;
                moveX(dx);
                moveY(dy);
                for(auto i : buts)
                    i->move(dx, dy);
                for(auto i : sprites)
                {
                    i->moveX(dx);
                    i->moveY(dy);
                }

                return nullptr;
                break;
            }
        default:
            return nullptr;
            break;
    }
}
```

7.17.3.8 rendering()

`void Script::rendering ( )  [virtual]`

Reimplemented from Object.

Definition at line 3 of file rendering.cpp.

```
{
    Object::rendering();
    for(auto& i : sprites)
        i->rendering();
    for(auto& i : buts)
        i->rendering();
    return ;
}
```

7.17.3.9 unhighlight()

```
void Script::unhighlight (
             int index )
```

Definition at line 9 of file highlight.cpp.

```
{
    if(index < 0 || index >= sprites.size()) return ;
    sprites[index]->setTextBoxTransparent(0);
}
```

The documentation for this class was generated from the following files:

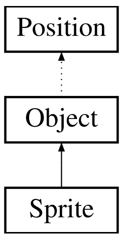
- include/script.hpp
- src/script/constructor.cpp
- src/script/destructor.cpp
- src/script/event.cpp
- src/script/highlight.cpp
- src/script/rendering.cpp

7.18 Sprite Class Reference

Object container Drawable.

`#include <sprite.hpp>`

Inheritance diagram for Sprite:



Public Member Functions

- Sprite (SDL_Renderer ∗r)
- ∼Sprite ()
- void linking (std::string n)
- void setFont (TTF_Font ∗f)
- void setFontColor (SDL_Color c)
- void setTextBox (TTF_Font ∗f)
- void setTextBoxTransparent (int a)
- void setText (std::string s)
- void typing (char c)
- void typing (std::string s)
- void backspace ()
- void backspace (int n)
- std::string getText ()
- void locating (int x, int y, int w, int h) override

- void locating (SDL_Rect r) override
- void locatingX (int x) override
- void locatingY (int y) override
- void moveX (int x) override
- void moveY (int y) override
- void aligning (HORIZONTAL_ALIGN h, VERTICAL_ALIGN v)
- void aligning (HORIZONTAL_ALIGN h)
- void aligning (VERTICAL_ALIGN v)
- void rendering () override
- void highlight ()
- void unhighlight ()
- const SDL_Rect ∗ getCrop ()
- void cropping (int x, int y, int w, int h)
- void cropping (SDL_Rect c)
- void cropping (const json &mem)
- void noCropping ()
- const SDL_Rect ∗ getLocation ()
- int getX ()
- int getY ()
- int getW ()
- int getH ()
- virtual void locating (const json &mem)
- virtual void locatingW (int w)
- virtual void locatingH (int h)
- virtual void zoomW (int delta)
- virtual void zoomH (int delta)
- virtual void zoom (double delta)
- virtual void zoomInMiddle (double delta)
- void fitTheTexture ()
- const SDL_Color ∗ getColor ()
- void coloring (int r, int g, int b, int a)
- void coloring (SDL_Color c)
- void coloring (const json &mem)
- void textureFromFile (std::string dir)
- void changeToCircle ()
- void changeToCircle (SDL_Point c)
- void changeToCircle (int x, int y)
- void changeToCircle (SDL_Point c, int r)
- void changeToCircle (int x, int y, int r)
- void changeToRectangle ()
- void setShape (const json &mem)
- bool isLieInside (int x, int y)
- bool isLieInside (SDL_Point p)
- bool isLieInside (SDL_Rect r)
- bool isLieInside (int x, int y, int w, int h)
- void show ()
- void hide ()
- bool isVisible ()
- void importFromJson (const json &mem)
- void addText (std::string t)
- void addCharacter (char c)
- void removeCharacter ()
- void removeCharacter (int n)
- int getSize ()

Protected Member Functions

- void initBackground (const json &mem)
- void initObjects (const json &mem)
- void initTextBox (const json &mem)
- void initInput (const json &mem)
- void importFromJson ()
- void aligning ()
- void fillWithColor ()
- void fillCircleByColor ()
- void fillRectangleByColor ()
- void textToTexture ()

7.18.1 Detailed Description

Object container Drawable.

Definition at line 20 of file sprite.hpp.

7.18.2 Constructor & Destructor Documentation

7.18.2.1 Sprite()

```
Sprite::Sprite (
              SDL_Renderer * r )
```

Definition at line 9 of file constructor.cpp.

```
                                : Object(r)
{
    render = r;
    textBox = nullptr;
    alignH = HORIZONTAL_ALIGN::CENTER;
    alignV = VERTICAL_ALIGN::CENTER;
    receiveDigit = true;
    receiveLetter = true;
    receiveSymbol = true;
    maxSize = 50;
}
```

7.18.2.2 ∼Sprite()

```
Sprite::∼Sprite ( )
```

Definition at line 3 of file destructor.cpp.

```
{
    for (auto& object : objects)
    {
        delete object;
    }
}
```

7.18.3 Member Function Documentation

7.18.3.1 addCharacter()

```
void Object::addCharacter (
             char c )  [inherited]
```

Definition at line 22 of file font.cpp.

```
{
    text += c;
    textToTexture();
}
```

7.18.3.2 addText()

```
void Object::addText (
             std::string t )  [inherited]
```

Definition at line 16 of file font.cpp.

```
{
    text += t;
    textToTexture();
}
```

7.18.3.3 aligning() [1/4]

```
void Sprite::aligning ( )  [protected]
```

Definition at line 4 of file aligning.cpp.

```
{
    if(textBox == nullptr) return ;
    int w = std::min(textBox->getW(), getW());
    int h = std::min(textBox->getH(), getH());

     textBox->cropping(textBox->getW() - w, textBox->getH() - h, w, h);
     textBox->locatingW(w);
     textBox->locatingH(h);
     switch(alignH)
     {
         case HORIZONTAL_ALIGN::LEFT:
             textBox->locatingX(getX());
             break;
         case HORIZONTAL_ALIGN::CENTER:
             textBox->locatingX(getX() + (getW() - w) / 2);
             break;
         case HORIZONTAL_ALIGN::RIGHT:
             textBox->locatingX(getX() + getW() - w);
             break;
     }
     switch(alignV)
     {
         case VERTICAL_ALIGN::TOP:
             textBox->locatingY(getY());
             break;
         case VERTICAL_ALIGN::CENTER:
             textBox->locatingY(getY() + (getH() - h) / 2);
             break;
         case VERTICAL_ALIGN::BOTTOM:
             textBox->locatingY(getY() + getH() - h);
             break;
     }
}
```

7.18.3.4 aligning() [2/4]

```
void Sprite::aligning (
            HORIZONTAL_ALIGN h )
```

Definition at line 46 of file aligning.cpp.

```
{
    alignH = h;
    aligning();
}
```

7.18.3.5 aligning() [3/4]

```
void Sprite::aligning (
            HORIZONTAL_ALIGN h,
            VERTICAL_ALIGN v )
```

Definition at line 39 of file aligning.cpp.

```
{
    alignH = h;
    alignV = v;
    aligning();
}
```

7.18.3.6 aligning() [4/4]

```
void Sprite::aligning (
            VERTICAL_ALIGN v )
```

Definition at line 52 of file aligning.cpp.

```
{
    alignV = v;
    aligning();
}
```

7.18.3.7 backspace() [1/2]

```
void Sprite::backspace ( )
```

Definition at line 52 of file textbox.cpp.

```
{
    if(textBox == nullptr) return ;
    textBox->removeCharacter();
    aligning();
}
```

7.18.3.8 backspace() [2/2]

```
void Sprite::backspace (
             int n )
```

Definition at line 59 of file textbox.cpp.

```
{
    if(textBox == nullptr) return ;
    textBox->removeCharacter(n);
    aligning();
}
```

7.18.3.9 changeToCircle() [1/5]

```
void Object::changeToCircle ( )  [inherited]
```

Definition at line 5 of file shape.cpp.

```
{
    shapeType = SHAPE::CIRCLE;
    radius = std::min(getW(), getH()) / 2;

    center.x = getX() + getW() / 2;
    center.y = getY() + getH() / 2;
    fillCircleByColor();
}
```

7.18.3.10 changeToCircle() [2/5]

```
void Object::changeToCircle (
             int x,
             int y )  [inherited]
```

Definition at line 24 of file shape.cpp.

```
{
    changeToCircle({x, y});
}
```

7.18.3.11 changeToCircle() [3/5]

```
void Object::changeToCircle (
             int x,
             int y,
             int r )  [inherited]
```

Definition at line 37 of file shape.cpp.

```
{
    shapeType = SHAPE::CIRCLE;
    radius = r;
    center.x = x;
    center.y = y;
    fillCircleByColor();
}
```

7.18.3.12 changeToCircle() [4/5]

```
void Object::changeToCircle (
            SDL_Point c )   [inherited]
```

Definition at line 15 of file shape.cpp.

```
{
    shapeType = SHAPE::CIRCLE;
    center = c;
    radius = std::min(getW() - c.x, c.x - getX());
    radius = std::min(radius, std::min(getH() - c.y, c.y - getY()));
    fillCircleByColor();
}
```

7.18.3.13 changeToCircle() [5/5]

```
void Object::changeToCircle (
            SDL_Point c,
            int r )   [inherited]
```

Definition at line 29 of file shape.cpp.

```
{
    shapeType = SHAPE::CIRCLE;
    radius = r;
    center = c;
    fillCircleByColor();
}
```

7.18.3.14 changeToRectangle()

```
void Object::changeToRectangle ( )   [inherited]
```

Definition at line 46 of file shape.cpp.

```
{
    shapeType = SHAPE::RECTANGLE;
    fillRectangleByColor();
}
```

7.18.3.15 coloring() [1/3]

```
void Object::coloring (
            const json & mem )   [inherited]
```

Definition at line 30 of file coloring.cpp.

```
{
    if(mem.contains("r") && mem.contains("g") && mem.contains("b"))
    {
        if(mem.contains("a")) coloring(mem["r"], mem["g"], mem["b"], mem["a"]);
        else coloring(mem["r"], mem["g"], mem["b"], 255);
    }
}
```

7.18.3.16 coloring() [2/3]

```
void Object::coloring (
            int r,
            int g,
            int b,
            int a )  [inherited]
```

Definition at line 8 of file coloring.cpp.

```
{
    if(color == nullptr) color = new SDL_Color;
    color->r = r;
    color->g = g;
    color->b = b;
    color->a = a;

    fillWithColor();
}
```

7.18.3.17 coloring() [3/3]

```
void Object::coloring (
            SDL_Color c )  [inherited]
```

Definition at line 19 of file coloring.cpp.

```
{
    if(color == nullptr) color = new SDL_Color;
    color->r = c.r;
    color->g = c.g;
    color->b = c.b;
    color->a = c.a;

    fillWithColor();
}
```

7.18.3.18 cropping() [1/3]

```
void Object::cropping (
            const json & mem )  [inherited]
```

Definition at line 26 of file cropping.cpp.

```
{
    if(mem.contains("x") && mem.contains("y") && mem.contains("w") && mem.contains("h"))
        cropping(mem["x"], mem["y"], mem["w"], mem["h"]);
}
```

7.18.3.19 cropping() [2/3]

```
void Object::cropping (
            int x,
            int y,
            int w,
            int h )  [inherited]
```

Definition at line 8 of file cropping.cpp.

```
{
    if(crop == nullptr) crop = new SDL_Rect;
    crop->x = x;
    crop->y = y;
    crop->w = w;
    crop->h = h;
}
```

7.18.3.20 cropping() [3/3]

```
void Object::cropping (
             SDL_Rect c )  [inherited]
```

Definition at line 17 of file cropping.cpp.

```
{
    if(crop == nullptr) crop = new SDL_Rect;
    crop->x = c.x;
    crop->y = c.y;
    crop->w = c.w;
    crop->h = c.h;
}
```

7.18.3.21 fillCircleByColor()

```
void Object::fillCircleByColor ( )  [protected], [inherited]
```

Definition at line 91 of file shape.cpp.

```
{
    if(location == nullptr) locating(0, 0, 0, 0);

    if(texture != nullptr) SDL_DestroyTexture(texture);
    texture = nullptr;

    Uint32 rmask, gmask, bmask, amask;
    Uint32 pixelColor;
#if SDL_BYTEORDER == SDL_BIG_ENDIAN
     rmask = 0xff000000;
     gmask = 0x00ff0000;
     bmask = 0x0000ff00;
     amask = 0x000000ff;
     pixelColor = (color->r « 24) | (color->g « 16) | (color->b « 8) | color->a;
#else
     rmask = 0x000000ff;
     gmask = 0x0000ff00;
     bmask = 0x00ff0000;
     amask = 0xff000000;
     pixelColor = (color->a « 24) | (color->b « 16) | (color->g « 8) | color->r;
#endif

     SDL_Surface *surf = SDL_CreateRGBSurface(0, getW(), getH(), 32, rmask, gmask, bmask, amask);
     SDL_SetSurfaceBlendMode(surf, SDL_BLENDMODE_BLEND);

     texture = SDL_CreateTextureFromSurface(render, surf);
     SDL_FreeSurface(surf);

     Uint32 *pixels = new Uint32[getW() * getH()];
     memset(pixels, 0, getW() * getH() * sizeof(Uint32));

     SDL_Point p = {getW() / 2, getH() / 2};
     center = p;

     if(radius > std::min(getW(), getH()) / 2) radius = std::min(getW(), getH()) / 2;

     for(int i = p.x - radius; i <= p.x + radius; i++)
         for(int j = p.y - radius; j <= p.y + radius; j++)
             if((i - p.x) * (i - p.x) + (j - p.y) * (j - p.y) <= radius * radius)
             {
                 int index = i * getW() + j;
                 if(index < 0 || index >= getW() * getH()) continue;
                 pixels[index] = pixelColor;
             }

     SDL_UpdateTexture(texture, nullptr, pixels, getW() * sizeof(Uint32));
     delete[] pixels;
}
```

7.18.3.22 fillRectangleByColor()

`void Object::fillRectangleByColor ( ) [protected], [inherited]`

Definition at line 74 of file shape.cpp.

```
{
    if(location == nullptr) locating(0, 0, 0, 0);

    if(texture != nullptr) SDL_DestroyTexture(texture);
    texture = nullptr;

    SDL_Surface* surf = SDL_CreateRGBSurfaceWithFormat(0, getW(), getH(), 32, SDL_PIXELFORMAT_RGBA32);

    SDL_SetSurfaceBlendMode(surf, SDL_BLENDMODE_BLEND);

    SDL_FillRect(surf, nullptr, SDL_MapRGBA(surf->format, color->r, color->g, color->b, color->a));

    texture = SDL_CreateTextureFromSurface(render, surf);

    SDL_FreeSurface(surf);
}
```

7.18.3.23 fillWithColor()

`void Object::fillWithColor ( ) [protected], [inherited]`

Definition at line 39 of file coloring.cpp.

```
{
    if(shapeType == SHAPE::NONE) return fillRectangleByColor();
    if(shapeType == SHAPE::RECTANGLE) return fillRectangleByColor();
    if(shapeType == SHAPE::CIRCLE) return fillCircleByColor();
}
```

7.18.3.24 fitTheTexture()

`void Object::fitTheTexture ( ) [inherited]`

Definition at line 140 of file locating.cpp.

```
{
    if(texture == nullptr) return;
    SDL_QueryTexture(texture, nullptr, nullptr, &location->w, &location->h);
}
```

7.18.3.25 getColor()

`const SDL_Color * Object::getColor ( ) [inherited]`

Definition at line 3 of file coloring.cpp.

```
{
    return color;
}
```

7.18.3.26 getCrop()

```
const SDL_Rect * Object::getCrop ( )  [inherited]
```

Definition at line 3 of file cropping.cpp.

```
{
    return crop;
}
```

7.18.3.27 getH()

```
int Object::getH ( )  [inherited]
```

Definition at line 47 of file locating.cpp.

```
{
    return location->h;
}
```

7.18.3.28 getLocation()

```
const SDL_Rect * Object::getLocation ( )  [inherited]
```

Definition at line 27 of file locating.cpp.

```
{
    return location;
}
```

7.18.3.29 getSize()

```
int Object::getSize ( )  [inherited]
```

Definition at line 68 of file font.cpp.

```
{
    return text.size();
}
```

7.18.3.30 getText()

```
std::string Sprite::getText ( )
```

Definition at line 49 of file constructor.cpp.

```
{
    if(textBox == nullptr) return "";
    return textBox->getText();
}
```

7.18.3.31 getW()

```
int Object::getW ( ) [inherited]
```

Definition at line 42 of file locating.cpp.

```
{
    return location->w;
}
```

7.18.3.32 getX()

```
int Object::getX ( ) [inherited]
```

Definition at line 32 of file locating.cpp.

```
{
    return location->x;
}
```

7.18.3.33 getY()

```
int Object::getY ( ) [inherited]
```

Definition at line 37 of file locating.cpp.

```
{
    return location->y;
}
```

7.18.3.34 hide()

```
void Object::hide ( ) [inherited]
```

Definition at line 8 of file visible.cpp.

```
{
    visible = false;
}
```

7.18.3.35 highlight()

```
void Sprite::highlight ( )
```

Definition at line 4 of file coloring.cpp.

```
{
   const SDL_Color* c = Object::getColor();

   if(c == nullptr) return ;

    cacheColor = {c->r, c->g, c->b, c->a};
    int maxChannel = std::max(std::max(c->r, c->g), c->b);
    Object::coloring(maxChannel * 0.4, maxChannel * 0.4, maxChannel * 0.4, c->a);
}
```

7.18.3.36 importFromJson() [1/2]

```
void Sprite::importFromJson ( )  [protected]
```

Definition at line 105 of file constructor.cpp.

```
{
    json* mem = JSON::readFile(PATH::ATB::SPRITE_ + name + ".json");

    initBackground((*mem)["background"]);

    if(mem->contains("objects")) initObjects((*mem)["objects"]);
    if(mem->contains("text-box")) initTextBox((*mem)["text-box"]);
    if(mem->contains("input")) initInput((*mem)["input"]);
    delete mem;
}
```

7.18.3.37 importFromJson() [2/2]

```
void Object::importFromJson (
            const json & mem )  [inherited]
```

Definition at line 21 of file constructor.cpp.

```
{
    if(mem.contains("location"))
        locating(mem["location"]);

    if(mem.contains("crop"))
        cropping(mem["crop"]);

    if(mem.contains("color"))
        coloring(mem["color"]);

    if(mem.contains("shape"))
        setShape(mem["shape"]);

    if(mem.contains("visible"))
        visible = mem["visible"];

    if(mem.contains("image"))
        textureFromFile(PATH::ASSETS::GRAPHICS_ + mem["image"].get<std::string>());
    return ;
}
```

7.18.3.38 initBackground()

```
void Sprite::initBackground (
            const json & mem )  [protected]
```

Definition at line 55 of file constructor.cpp.

```
{
    Object::importFromJson(mem);
    if(Object::getColor() == nullptr) return ;
    cacheColor = *Object::getColor();
}
```

7.18.3.39 initInput()

```
void Sprite::initInput (
            const json & mem )  [protected]
```

Definition at line 94 of file constructor.cpp.

```
{
    if(textBox == nullptr) return;
    if(mem.contains("digit")) receiveDigit = mem["digit"].get<bool>();
    if(mem.contains("letter")) receiveLetter = mem["letter"].get<bool>();
    if(mem.contains("symbol")) receiveSymbol = mem["symbol"].get<bool>();
    if(mem.contains("lower")) numberLower = mem["lower"].get<std::string>();
    if(mem.contains("upper")) numberUpper = mem["upper"].get<std::string>();
    if(mem.contains("maxsize")) maxSize = mem["maxsize"].get<int>();
}
```

7.18.3.40 initObjects()

```
void Sprite::initObjects (
            const json & mem )  [protected]
```

Definition at line 62 of file constructor.cpp.

```
{
    objects.clear();
    for(auto& i : mem)
    {
        Object* obj = new Object(render);
        if(i.contains("name")) obj->linking(i["name"]);

        obj->importFromJson(i);

        obj->moveX(getX());
        obj->moveY(getY());

        objects.push_back(obj);
    }
}
```

7.18.3.41 initTextBox()

```
void Sprite::initTextBox (
            const json & mem )  [protected]
```

Definition at line 79 of file constructor.cpp.

```
{
    if(textBox == nullptr)
    {
        objects.push_back(new Object(render));
        textBox = objects.back();
    }
    textBox->importFromJson(mem);
    textBox->setFont(this->font);
    textBox->moveX(getX());
    textBox->moveY(getY());
    if(mem.contains("text")) textBox->setText(mem["text"].get<std::string>());
    aligning();
}
```

7.18.3.42 isLieInside() [1/4]

```
bool Object::isLieInside (
            int x,
            int y )  [inherited]
```

Definition at line 3 of file locating.cpp.

```
{
    if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
    return (x >= location->x && x < location->x + location->w && y >= location->y && y < location->y +
     location->h);
}
```

7.18.3.43 isLieInside() [2/4]

```
bool Object::isLieInside (
            int x,
            int y,
            int w,
            int h )  [inherited]
```

Definition at line 21 of file locating.cpp.

```
{
    if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
    return (x >= location->x && x + w <= location->x + location->w && y >= location->y && y + h <=
    location->y + location->h);
}
```

7.18.3.44 isLieInside() [3/4]

```
bool Object::isLieInside (
            SDL_Point p )  [inherited]
```

Definition at line 9 of file locating.cpp.

```
{
    if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
    return (p.x >= location->x && p.x < location->x + location->w && p.y >= location->y && p.y <
    location->y + location->h);
}
```

7.18.3.45 isLieInside() [4/4]

```
bool Object::isLieInside (
            SDL_Rect r )  [inherited]
```

Definition at line 15 of file locating.cpp.

```
{
    if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
    return (r.x >= location->x && r.x + r.w <= location->x + location->w && r.y >= location->y && r.y +
    r.h <= location->y + location->h);
}
```

7.18.3.46 isVisible()

```
bool Object::isVisible ( )  [inherited]
```

Definition at line 13 of file visible.cpp.

```
{
    return visible;
}
```

7.18.3.47 linking()

```
void Sprite::linking (
             std::string n )
```

Definition at line 21 of file constructor.cpp.

```
{
    name = n;
    importFromJson();
}
```

7.18.3.48 locating() [1/3]

```
void Object::locating (
             const json & mem )  [virtual], [inherited]
```

Definition at line 70 of file locating.cpp.

```
{
    if(mem.contains("x") && mem.contains("y") && mem.contains("w") && mem.contains("h"))
        locating(mem["x"], mem["y"], mem["w"], mem["h"]);
}
```

7.18.3.49 locating() [2/3]

```
void Sprite::locating (
             int x,
             int y,
             int w,
             int h )  [override], [virtual]
```

Reimplemented from Object.

Definition at line 3 of file locating.cpp.

```
{
    Object::locating(x, y, w, h);
    for (auto &i : objects)
    {
        i->locating(x, y, w, h);
    }
}
```

7.18.3.50 locating() [3/3]

```
void Sprite::locating (
            SDL_Rect r )  [override], [virtual]
```

Reimplemented from Object.

Definition at line 12 of file locating.cpp.

```
{
    Object::locating(r);
    for (auto &i : objects)
    {
        i->locating(r);
    }
}
```

7.18.3.51 locatingH()

```
void Object::locatingH (
            int h )  [virtual], [inherited]
```

Definition at line 94 of file locating.cpp.

```
{
    if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
    location->h = h;
}
```

7.18.3.52 locatingW()

```
void Object::locatingW (
            int w )  [virtual], [inherited]
```

Definition at line 88 of file locating.cpp.

```
{
    if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
    location->w = w;
}
```

7.18.3.53 locatingX()

```
void Sprite::locatingX (
            int x )  [override], [virtual]
```

Reimplemented from Object.

Definition at line 20 of file locating.cpp.

```
{
    Object::locatingX(x);
    for (auto &i : objects)
    {
        i->locatingX(x);
    }
}
```

7.18.3.54 locatingY()

```
void Sprite::locatingY (
              int y )  [override], [virtual]
```

Reimplemented from Object.

Definition at line 29 of file locating.cpp.

```
{
    Object::locatingY(y);
    for (auto &i : objects)
    {
        i->locatingY(y);
    }
}
```

7.18.3.55 moveX()

```
void Sprite::moveX (
              int x )  [override], [virtual]
```

Reimplemented from Object.

Definition at line 38 of file locating.cpp.

```
{
    Object::moveX(x);
    for (auto &i : objects)
    {
        i->moveX(x);
    }
}
```

7.18.3.56 moveY()

```
void Sprite::moveY (
              int y )  [override], [virtual]
```

Reimplemented from Object.

Definition at line 47 of file locating.cpp.

```
{
    Object::moveY(y);
    for (auto &i : objects)
    {
        i->moveY(y);
    }
}
```

7.18.3.57 noCropping()

```
void Object::noCropping ( )  [inherited]
```

Definition at line 32 of file cropping.cpp.

```
{
    if(crop != nullptr) delete crop;
    crop = nullptr;
}
```

7.18.3.58 removeCharacter() [1/2]

```
void Object::removeCharacter ( )  [inherited]
```

Definition at line 28 of file font.cpp.

```
{
    if (text.size() > 0)
        text.pop_back();
    textToTexture();
}
```

7.18.3.59 removeCharacter() [2/2]

```
void Object::removeCharacter (
              int n )  [inherited]
```

Definition at line 35 of file font.cpp.

```
{
    if(n == 0) return ;
    if(text.size() <= n) text.clear();
    else text.erase(text.end() - n, text.end());
    textToTexture();
}
```

7.18.3.60 rendering()

```
void Sprite::rendering ( )  [override], [virtual]
```

Reimplemented from Object.

Definition at line 3 of file rendering.cpp.

```
{
    Object::rendering();
    for(auto& obj : objects)
    {
        obj->rendering();
    }
}
```

7.18.3.61 setFont()

```
void Sprite::setFont (
              TTF_Font * f )
```

Definition at line 26 of file constructor.cpp.

```
{
    font = f;
    Object::setFont(f);
}
```

7.18.3.62 setFontColor()

```
void Sprite::setFontColor (
            SDL_Color c )
```

Definition at line 31 of file constructor.cpp.

```
{
    if(textBox != nullptr)
    {
        textBox->coloring(c);
        textBox->setText(textBox->getText());
    }
}
```

7.18.3.63 setShape()

```
void Object::setShape (
            const json & mem )  [inherited]
```

Definition at line 52 of file shape.cpp.

```
{
    if(mem["type"].get<std::string>() == "CIRCLE")
    {
        if(mem.contains("center"))
        {
            if(mem.contains("radius"))
                changeToCircle(mem["center"]["x"], mem["center"]["y"], mem["radius"]);
            else changeToCircle(mem["center"]["x"], mem["center"]["y"]);
        }else changeToCircle();

        return ;
    }

    if(mem["type"].get<std::string>() == "NONE" || mem["type"].get<std::string>() == "RECTANGLE")
    {
        changeToRectangle();
        return ;
    }

}
```

7.18.3.64 setText()

```
void Sprite::setText (
            std::string s )
```

Definition at line 12 of file textbox.cpp.

```
{
    if(textBox == nullptr) return ;
    for(char c : s)
    {
        if(!receiveDigit && NUMBER::isDigit(c)) return ;
        if(!receiveLetter && NUMBER::isLetter(c)) return ;
        if(!receiveSymbol && NUMBER::isSymbol(c)) return ;
    }
    if(s.size() > maxSize) s = s.substr(0, maxSize);
    textBox->setText(s);
    aligning();
}
```

7.18.3.65 setTextBox()

```
void Sprite::setTextBox (
              TTF_Font * f )
```

Definition at line 39 of file constructor.cpp.

```
{
    if(textBox == nullptr)
    {
        objects.push_back(new Object(render));
        textBox = objects.back();
    }
    textBox->setFont(f);
}
```

7.18.3.66 setTextBoxTransparent()

```
void Sprite::setTextBoxTransparent (
              int a )
```

Definition at line 4 of file textbox.cpp.

```
{
    if(textBox == nullptr) return ;
    SDL_Color c = *getColor();
    c.a = a;
    coloring(c);
}
```

7.18.3.67 show()

```
void Object::show ( )  [inherited]
```

Definition at line 3 of file visible.cpp.

```
{
    visible = true;
}
```

7.18.3.68 textToTexture()

```
void Object::textToTexture ( )  [protected], [inherited]
```

Definition at line 43 of file font.cpp.

```
{
    if(font == nullptr) return ;
    if(color == nullptr) return ;
    if(render == nullptr) return ;
    if(texture != nullptr)
    {
        SDL_DestroyTexture(texture);
    }
    texture = nullptr;

    SDL_Surface* surface = TTF_RenderText_Blended(font, text.c_str(), *color);

    if(surface == nullptr) return ;

    texture = SDL_CreateTextureFromSurface(render, surface);
    SDL_FreeSurface(surface);
    fitTheTexture();
}
```

7.18.3.69 textureFromFile()

```
void Object::textureFromFile (
            std::string dir )  [inherited]
```

Definition at line 4 of file external_storage.cpp.

```
{
    SDL_Surface *surface = IMG_Load(dir.c_str());

    texture = SDL_CreateTextureFromSurface(render, surface);
    SDL_FreeSurface(surface);
}
```

7.18.3.70 typing() [1/2]

```
void Sprite::typing (
            char c )
```

Definition at line 26 of file textbox.cpp.

```
{
    if(textBox == nullptr) return ;
    if(!receiveDigit && NUMBER::isDigit(c)) return ;
    if(!receiveLetter && NUMBER::isLetter(c)) return ;
    if(!receiveSymbol && NUMBER::isSymbol(c)) return ;
    if(textBox->getSize() >= maxSize) return ;
    textBox->addCharacter(c);
    aligning();
}
```

7.18.3.71 typing() [2/2]

```
void Sprite::typing (
            std::string s )
```

Definition at line 37 of file textbox.cpp.

```
{
    if(textBox == nullptr) return ;
    if(textBox->getText().size() + s.size() >= maxSize) return ;
    for(char c : s)
    {
        if(!receiveDigit && NUMBER::isDigit(c)) return ;
        if(!receiveLetter && NUMBER::isLetter(c)) return ;
        if(!receiveSymbol && NUMBER::isSymbol(c)) return ;
    }
    if(textBox->getSize() + s.size() > maxSize) s = s.substr(0, maxSize);
    textBox->addText(s);
    aligning();
}
```

7.18.3.72 unhighlight()

```
void Sprite::unhighlight ( )
```

Definition at line 15 of file coloring.cpp.

```
{
    Object::coloring(cacheColor);
}
```

7.18.3.73 zoom()

```
void Object::zoom (
              double delta )  [virtual], [inherited]
```

Definition at line 123 of file locating.cpp.

```
{
    if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
    location->w *= delta;
    location->h *= delta;
}
```

7.18.3.74 zoomH()

```
void Object::zoomH (
              int delta )  [virtual], [inherited]
```

Definition at line 118 of file locating.cpp.

```
{
    if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
    location->h += delta;
}
```

7.18.3.75 zoomInMiddle()

```
void Object::zoomInMiddle (
              double delta )  [virtual], [inherited]
```

Definition at line 130 of file locating.cpp.

```
{
    if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
    SDL_Point center = {location->x + location->w / 2, location->y + location->h / 2};
    location->w *= delta;
    location->h *= delta;
    location->x = center.x - location->w / 2;
    location->y = center.y - location->h / 2;
}
```

7.18.3.76 zoomW()

```
void Object::zoomW (
              int delta )  [virtual], [inherited]
```

Definition at line 112 of file locating.cpp.

```
{
    if(location == nullptr) location = new SDL_Rect({0, 0, 0, 0});
    location->w += delta;
}
```

The documentation for this class was generated from the following files:

- include/sprite.hpp
- src/sprite/aligning.cpp
- src/sprite/coloring.cpp
- src/sprite/constructor.cpp
- src/sprite/destructor.cpp
- src/sprite/locating.cpp
- src/sprite/rendering.cpp
- src/sprite/textbox.cpp

7.19 Trie Class Reference

Trie data structure.

`#include <trie.hpp>`

Public Member Functions

- Trie (SDL_Renderer ∗r, std::mutex &m, TTF_Font ∗f, SDL_Rect v, int capacity)
- ∼Trie ()
- void init (std::vector< std::string > words)
- bool insert (std::string word)
- bool search (std::string word)
- bool remove (std::string word)
- void setEdgesColor (SDL_Color c)
- void setNodeColor (SDL_Color bg, SDL_Color fg)
- void rendering ()
- bool isReceiveEvent (SDL_Event &e)
- Button ∗ react (SDL_Event &e)
- void closeScript ()
- void setting (SDL_Color c1, SDL_Color c2, SDL_Color c3, SDL_Color c4)

Protected Member Functions

- Node ∗ insert (Node ∗node, std::string word, int index)
- bool search (Node ∗node, std::string word, int index)
- Node ∗ remove (Node ∗node, std::string word, int index)
- int locating (Node ∗node, int shiftDown, int shiftRight)
- void drawEgdes (Node ∗u, Node ∗v)
- void waitForStep ()
- void highlight (std::vector< int > l)
- void unhighlight (std::vector< int > l)

7.19.1 Detailed Description

Trie data structure.

Definition at line 20 of file trie.hpp.

7.19.2 Constructor & Destructor Documentation

7.19.2.1 Trie()

```
Trie::Trie (
            SDL_Renderer * r,
            std::mutex & m,
            TTF_Font * f,
            SDL_Rect v,
            int capacity )
```

Definition at line 14 of file constructor.cpp.

```
                                                          : ds_mutex(m)
{
    render = r;
    font = f;
    viewport = v;
    capacity = cap;
    size = 0;
    root = nullptr;
    edgesColor = {255, 255, 255, 255};
    fontColor = {255, 255, 255, 255};
    nodeColor = {20, 85, 185, 255};
    shiftX = 20;
    shiftX = 20;
    distanceX = 60;
    distanceY = 80;
    isMoving = false;

    isQueue = false;
    isPause = false;
    stepWait = 600;
    isAnimate = false;

    std::string fontpath = PATH::ASSETS::FONTS_ + "nimbus-sans-l/regular.otf";
    scriptFont = TTF_OpenFont(fontpath.c_str(), 18);

    currentScript = nullptr;
    Script* insert = new Script(render, scriptFont);
    insert->linking("trie/insert");
    scripts[DATA_STRUCTURES_OPERATOR::INSERT] = insert;

    Script* remove = new Script(render, scriptFont);
    remove->linking("trie/remove");
    scripts[DATA_STRUCTURES_OPERATOR::DELETE] = remove;

    Script* search = new Script(render, scriptFont);
    search->linking("trie/search");
    scripts[DATA_STRUCTURES_OPERATOR::SEARCH] = search;

    Script* init = new Script(render, scriptFont);
    init->linking("trie/init");
    scripts[DATA_STRUCTURES_OPERATOR::INIT] = init;
}
```

7.19.2.2 ∼Trie()

```
Trie::∼Trie ( )
```

Definition at line 9 of file destructor.cpp.

```
{
    if(root != nullptr) delete root;

    for(auto i : scripts)
        delete i.second;
    TTF_CloseFont(scriptFont);
}
```

7.19.3 Member Function Documentation

7.19.3.1 closeScript()

```
void Trie::closeScript ( )
```

Definition at line 72 of file event.cpp.

```
{
    currentScript = nullptr;
}
```

7.19.3.2 drawEgdes()

```
void Trie::drawEgdes (
            Node * u,
            Node * v )  [protected]
```

Definition at line 4 of file rendering.cpp.

```
{
    SDL_Point psrc = {u->sprite->getX() + u->sprite->getW() / 2, u->sprite->getY() + u->sprite->getH() /
     2};
    SDL_Point pdst = {v->sprite->getX() + v->sprite->getW() / 2, v->sprite->getY() + v->sprite->getH() /
     2};

    SDL_SetRenderDrawColor(render, edgesColor.r, edgesColor.g, edgesColor.b, edgesColor.a);
     for(int i = -1; i <= 1; i++)
     {
         for(int j = -1; j <= 1; j++)
             SDL_RenderDrawLine(render, psrc.x + i, psrc.y + j, pdst.x + i, pdst.y + j);
     }
}
```

7.19.3.3 highlight()

```
void Trie::highlight (
            std::vector< int > l )  [protected]
```

Definition at line 26 of file step.cpp.

```
{
     if(isAnimate)
     {
         animate_mutex.lock();
         for(int i = 0; i < l.size(); i++)
         {
             currentScript->highlight(l[i]);
         }
         animate_mutex.unlock();
     }
}
```

7.19.3.4 init()

```
void Trie::init (
            std::vector< std::string > words )
```

Definition at line 4 of file init.cpp.

```
{
    if(root != nullptr) delete root;
    root = nullptr;
    currentScript = scripts[DATA_STRUCTURES_OPERATOR::INIT];
    isAnimate = false;
     for(auto word : words)
     {
         root = insert(root, word, 0);
     }
}
```

7.19.3.5 insert() [1/2]

```
Trie::Node * Trie::insert (
              Node * node,
              std::string word,
              int index )  [protected]
```

Definition at line 3 of file insert.cpp.

```
{
    if(node == nullptr)
    {
        Sprite* spr = new Sprite(render);
        spr->setFont(font);
        spr->linking("trie/node");
         spr->typing(' ');
         spr->setFontColor(fontColor);
         spr->coloring(nodeColor);

         node = new Node('\0', spr);
         if(root == nullptr) root = node;
         node->sprite->coloring(nodeColor);

         if(isAnimate)
         {
             highlight({1, 2, 3});
             waitForStep();
             unhighlight({1, 2, 3});
         }
     }
     if(isAnimate)
     {

         animate_mutex.lock();
         node->sprite->highlight();
         animate_mutex.unlock();

         waitForStep();

         animate_mutex.lock();
         node->sprite->unhighlight();
         animate_mutex.unlock();

     }

     highlight({4, 5, 6, 7, 8});
     waitForStep();
     unhighlight({4, 5, 6, 7, 8});

     node->numberOfWords++;
     if(index == (int) word.size())
     {
         node->endOfWords++;
         return node;
     }

     highlight({9, 10, 11, 12, 13, 14});
     waitForStep();
     unhighlight({9, 10, 11, 12, 13, 14});

     int key = word[index] - 'a';

     Node *& currentChild = node->childs[key];
     if(currentChild == nullptr)
     {
         Sprite* spr = new Sprite(render);
         spr->setFont(font);
         spr->linking("trie/node");
         spr->typing(word[index]);
         spr->setFontColor(fontColor);
         spr->coloring(nodeColor);

         currentChild = new Node(key, spr);
     }
     currentChild = insert(currentChild, word, index + 1);

     if(isAnimate)
     {
         animate_mutex.lock();
         node->sprite->highlight();
         animate_mutex.unlock();

         waitForStep();

         animate_mutex.lock();
         node->sprite->unhighlight();
         animate_mutex.unlock();
     }
     highlight({15});
     waitForStep();
     unhighlight({15});
     return node;
 }
```

7.19.3.6 insert() [2/2]

```
bool Trie::insert (
        std::string word )
```

Definition at line 89 of file insert.cpp.

```
{

    currentScript = scripts[DATA_STRUCTURES_OPERATOR::INSERT];
    isAnimate = true;


    highlight({0});
    waitForStep();
    unhighlight({0});

     root = insert(root, word, 0);
     return true;
}
```

7.19.3.7 isReceiveEvent()

```
bool Trie::isReceiveEvent (
        SDL_Event & e )
```

Definition at line 3 of file event.cpp.

```
{
    switch(e.type)
    {
        case SDL_MOUSEBUTTONDOWN:
            if(currentScript != nullptr && currentScript->isReceiveEvent(e)) return true;
            if(e.motion.x < viewport.x || viewport.x + viewport.w < e.motion.x) return false;
             if(e.motion.y < viewport.y || viewport.y + viewport.h < e.motion.y) return false;
             if(e.button.button == SDL_BUTTON_LEFT) return false;
             if(root == nullptr) return false;
             return true;
             break;
         case SDL_MOUSEMOTION:
             if(isMoving) return true;
             if(currentScript == nullptr) return false;
             if(currentScript->isReceiveEvent(e)) return true;
             return false;
             break;
         default:
             return false;
             break;
    }
}
```

7.19.3.8 locating()

```
int Trie::locating (
            Node * node,
            int shiftDown,
            int shiftRight )  [protected]
```

Definition at line 57 of file constructor.cpp.

```
{
    if(node == nullptr) return 0;

    int j = 0;
    for(int i = 0; i < 26; i++)
        if(node->childs[i] != nullptr) j++;
    j /= 2;

    int i = 0;

    for(; j > 0; i++)
    {
        if(node->childs[i] == nullptr)
            continue;
        j--;

        shiftRight = locating(node->childs[i], shiftDown + 1, shiftRight);
    }

    node->sprite->locatingX(shiftX + shiftRight * distanceX);
    node->sprite->locatingY(shiftY + shiftDown * distanceY);
    node->sprite->aligning(HORIZONTAL_ALIGN::CENTER, VERTICAL_ALIGN::CENTER);
    shiftRight++;

    for(; i < 26; i++)
    {
        if(node->childs[i] == nullptr)
            continue;
        shiftRight = locating(node->childs[i], shiftDown + 1, shiftRight);
    }

    return shiftRight;
}
```

7.19.3.9 react()

```
Button * Trie::react (
            SDL_Event & e )
```

Definition at line 27 of file event.cpp.

```
{
    switch(e.type)
    {
        case SDL_MOUSEBUTTONDOWN:
            if(currentScript != nullptr && currentScript->isReceiveEvent(e))
            {
                return currentScript->react(e);
            }
            if(isMoving)
            {
                isMoving = false;
                int dx = e.motion.x - lastMousePressed.x;
                int dy = e.motion.y - lastMousePressed.y;
                shiftX += dx;
                shiftY += dy;
            }else
            {
                isMoving = true;
                lastMousePressed.x = e.motion.x;
                lastMousePressed.y = e.motion.y;
            }
            return nullptr;
            break;
        case SDL_MOUSEMOTION:
        {
            if(currentScript != nullptr && currentScript->isReceiveEvent(e))
```

```
                return currentScript->react(e);
            if(!isMoving) return nullptr;
            int dx = e.motion.x - lastMousePressed.x;
            int dy = e.motion.y - lastMousePressed.y;
            lastMousePressed.x = e.motion.x;
            lastMousePressed.y = e.motion.y;
            shiftX += dx;
            shiftY += dy;
            return nullptr;
            break;
        }
        defaut:
            return nullptr;
            break;
    }
    return nullptr;
}
```

7.19.3.10 remove() [1/2]

```
Trie::Node * Trie::remove (
              Node * node,
              std::string word,
              int index )  [protected]
```

Definition at line 3 of file remove.cpp.

```
{
    if(node == nullptr)
    {
        highlight({1, 2, 3});
        waitForStep();
        unhighlight({1, 2, 3});
         return nullptr;
     }
     node->numberOfWords--;

     if(isAnimate)
     {
         animate_mutex.lock();
         node->sprite->highlight();
         animate_mutex.unlock();

         waitForStep();
     }

     if(index == (int) word.size())
     {
         highlight({4, 5});
         waitForStep();
         unhighlight({4, 5});

         node->endOfWords--;
     }else
     {
         highlight({6, 7, 8, 9, 10, 11, 12, 13});
         waitForStep();
         unhighlight({6, 7, 8, 9, 10, 11, 12, 13});

         if(isAnimate)
         {
             animate_mutex.lock();
             node->sprite->unhighlight();
             animate_mutex.unlock();
         }

         int key = word[index] - 'a';
         node->childs[key] = remove(node->childs[key], word, index + 1);
     }
     if(isAnimate)
     {
         animate_mutex.lock();
         node->sprite->highlight();
         animate_mutex.unlock();

         waitForStep();
         animate_mutex.lock();
         node->sprite->unhighlight();
         animate_mutex.unlock();
     }

     if(node->numberOfWords == 0)
     {
         highlight({14});
         waitForStep();
         unhighlight({14});
         delete node;
         node = nullptr;
     }

     return node;
 }
```

7.19.3.11 remove() [2/2]

```
bool Trie::remove (
              std::string word )
```

Definition at line 70 of file remove.cpp.

```
{
    currentScript = scripts[DATA_STRUCTURES_OPERATOR::DELETE];

    isAnimate = false;
    if(!search(root, word, 0))
    {
        isAnimate = true;
        highlight({1, 2, 3});
        waitForStep();
        unhighlight({1, 2, 3});

        return false;
    }
    isAnimate = true;

    highlight({0});
    waitForStep();
    unhighlight({0});

    root = remove(root, word, 0);
    return true;
}
```

7.19.3.12 rendering()

```
void Trie::rendering ( )
```

Definition at line 17 of file rendering.cpp.

```
{
    if(root == nullptr) return ;
    SDL_RenderSetViewport(render, &viewport);
    locating(root, 0, 0);

    std::queue<Node*> q;
    q.push(root);

    while(!q.empty())
    {
        Node* u = q.front();
        q.pop();

        for(int i = 0; i < 26; i++)
        {
            if(u->childs[i] != nullptr)
            {
                q.push(u->childs[i]);
                drawEgdes(u, u->childs[i]);
            }
        }
        u->sprite->rendering();
    }
    if(currentScript != nullptr)
    {
        SDL_RenderSetViewport(render, nullptr);
        currentScript->rendering();
    }
}
```

7.19.3.13 search() [1/2]

```
bool Trie::search (
            Node * node,
            std::string word,
            int index )  [protected]
```

Definition at line 3 of file search.cpp.

```
{
    if(node == nullptr)
    {
        highlight({1, 2, 3});
        waitForStep();
        unhighlight({1, 2, 3});
        return false;
    }
    if(isAnimate)
    {
        animate_mutex.lock();
        node->sprite->highlight();
        animate_mutex.unlock();

        waitForStep();

    }
    if(index == (int) word.size())
    {
        highlight({4, 5, 6});
        waitForStep();
        unhighlight({4, 5, 6});
        if(isAnimate)
        {
            animate_mutex.lock();
            node->sprite->unhighlight();
            animate_mutex.unlock();
        }
        return node->endOfWords > 0;
    }
    int key = word[index] - 'a';
    if(isAnimate)
    {
        animate_mutex.lock();
        node->sprite->unhighlight();
        animate_mutex.unlock();
    }

    highlight({7, 8, 9, 10, 11, 12});
    waitForStep();
    unhighlight({7, 8, 9, 10, 11, 12});

    return search(node->childs[key], word, index + 1);
}
```

7.19.3.14 search() [2/2]

```
bool Trie::search (
            std::string word )
```

Definition at line 49 of file search.cpp.

```
{
    currentScript = scripts[DATA_STRUCTURES_OPERATOR::SEARCH];
    isAnimate = true;

    highlight({0});
    waitForStep();
    unhighlight({0});

    return search(root, word, 0);
}
```

7.19.3.15 setEdgesColor()

```
void Trie::setEdgesColor (
             SDL_Color c )
```

7.19.3.16 setNodeColor()

```
void Trie::setNodeColor (
             SDL_Color bg,
             SDL_Color fg )
```

7.19.3.17 setting()

```
void Trie::setting (
             SDL_Color c1,
             SDL_Color c2,
             SDL_Color c3,
             SDL_Color c4 )
```

Definition at line 92 of file constructor.cpp.

```
{
    bgColor = c1;
    nodeColor = c2;
    fontColor = c3;
    edgesColor = c4;

    std::queue<Node*> q;
    if(root != nullptr)
        q.push(root);

    while(!q.empty())
    {
        Node* node = q.front();
        q.pop();

        node->sprite->coloring(nodeColor);
        node->sprite->setFontColor(fontColor);
        node->sprite->coloring(nodeColor);

        for(int i = 0; i < 26; i++)
            if(node->childs[i] != nullptr)
                q.push(node->childs[i]);
    }
}
```

7.19.3.18 unhighlight()

```
void Trie::unhighlight (
            std::vector< int > l )  [protected]
```

Definition at line 39 of file step.cpp.

```
{
    if(isAnimate)
    {
        animate_mutex.lock();
        for(int i = 0; i < l.size(); i++)
        {
            currentScript->unhighlight(l[i]);
        }
        animate_mutex.unlock();
    }
}
```

7.19.3.19 waitForStep()

```
void Trie::waitForStep ( )  [protected]
```

Definition at line 5 of file step.cpp.

```
{
    if(isAnimate)
    {
        ds_mutex.unlock();
         std::this_thread::sleep_for(std::chrono::milliseconds(stepWait));
         ds_mutex.lock();
     }
     std::lock_guard<std::mutex> pause_lock(pause_mutex);
     if(isPause == false)
     {
         return ;
     }

     ds_mutex.unlock();
     std::unique_lock<std::mutex> lk(step_mutex);
     step_cv.wait(lk, [&]{return isQueue == true;});
     isQueue = false;
     ds_mutex.lock();
}
```

The documentation for this class was generated from the following files:

- include/data_structures/trie.hpp
- src/trie/constructor.cpp
- src/trie/destructor.cpp
- src/trie/event.cpp
- src/trie/operator/init.cpp
- src/trie/operator/insert.cpp
- src/trie/operator/remove.cpp
- src/trie/operator/search.cpp
- src/trie/rendering.cpp
- src/trie/step.cpp

Chapter 8

File Documentation

8.1 include/button.hpp File Reference

```
#include <vector>
#include <string>
#include <object.hpp>
#include <sprite.hpp>
#include <GLOBAL.hpp>
```

Classes

- class Button

  Button class that interact with user input.

8.2 include/data_structures.hpp File Reference

```
#include <iostream>
#include <vector>
#include <string>
#include <mutex>
#include <SDL2/SDL.h>
#include <GLOBAL.hpp>
#include <object.hpp>
#include <sprite.hpp>
#include <services.hpp>
#include <display.hpp>
#include <inputbox.hpp>
#include <data_structures/AVL.hpp>
#include <data_structures/trie.hpp>
#include <data_structures/hash_table.hpp>
#include <data_structures/minheap.hpp>
#include <data_structures/maxheap.hpp>
#include <data_structures/graph.hpp>
```

Classes

- class DataStructures

  Container that contains all data structures.

8.3 include/data_structures/AVL.hpp File Reference

```
#include <iostream>
#include <cmath>
#include <mutex>
#include <condition_variable>
#include <SDL2/SDL.h>
#include <SDL2/SDL_ttf.h>
#include <sprite.hpp>
#include <script.hpp>
```

Classes

- class AVL

  AVL class.

8.4 include/data_structures/btree4th.hpp File Reference

```
#include <vector>
#include <string>
#include <iostream>
```

Classes

- class BTree4th

8.5 include/data_structures/graph.hpp File Reference

```
#include <vector>
#include <string>
#include <queue>
#include <stack>
#include <iostream>
#include <mutex>
#include <thread>
#include <condition_variable>
#include <SDL2/SDL.h>
#include <SDL2/SDL_ttf.h>
#include <sprite.hpp>
#include <button.hpp>
```

Classes

- class Graph

  Graph class.

8.6 include/data_structures/hash_table.hpp File Reference

```
#include <iostream>
#include <vector>
#include <map>
#include <mutex>
#include <condition_variable>
#include <thread>
#include <SDL2/SDL.h>
#include <SDL2/SDL_ttf.h>
#include <sprite.hpp>
#include <GLOBAL.hpp>
#include <script.hpp>
```

Classes

- class HashTable

  HashTable class.

8.7 include/data_structures/maxheap.hpp File Reference

```
#include <iostream>
#include <vector>
#include <string>
```

Classes

- class maxHeap

8.8 include/data_structures/minheap.hpp File Reference

```
#include <iostream>
#include <vector>
#include <string>
#include <mutex>
#include <thread>
#include <condition_variable>
#include <SDL2/SDL.h>
#include <SDL2/SDL_ttf.h>
#include <sprite.hpp>
#include <button.hpp>
#include <script.hpp>
```

Classes

- class minHeap

  heap class.

8.9 include/data_structures/trie.hpp File Reference

```
#include <iostream>
#include <string>
#include <vector>
#include <mutex>
#include <condition_variable>
#include <SDL2/SDL.h>
#include <SDL2/SDL_ttf.h>
#include <script.hpp>
#include <sprite.hpp>
```

Classes

- class Trie

  Trie data structure.

8.10 include/display.hpp File Reference

```
#include <vector>
#include <string>
#include <SDL2/SDL.h>
#include <services.hpp>
#include <GLOBAL.hpp>
#include <object.hpp>
#include <button.hpp>
```

Classes

- class Display

  container of button intermediate between button and user input, window

8.11 include/GLOBAL.hpp File Reference

```
#include <string>
#include <random>
#include <fstream>
#include <nlohmann/json.hpp>
```

Namespaces

- DISPLAY

  Name of display.
- PATH

  Path to assets, atributes, and saving files.
- PATH::ASSETS
- PATH::ATB
- PATH::SAVING

Typedefs

- using json = nlohmann::json

Enumerations

- enum class WINDOW_STATUS { IS_OPEN , IS_CLOSED }

  Status of window.
- enum class SHAPE {
  NONE , RECTANGLE , CIRCLE , TRIANGLE ,
  LINE , POLYGON }

  Kind of simple shape.
- enum class BUTTON_STATUS { NONE , HOVER , RELEASED , CLICKED }

  Status of button.
- enum class BUTTON_ACTION {
  CHANGE_SCREEN , INSERT , DELETE , INIT ,
  SEARCH , SETTING , DONE , EDGES ,
  GO_BACK , GO_NEXT , GO_ON , GO_OFF ,
  SPEED_UP , SLOW_DOWN , CLOSE , TOP ,
  SIZE , CONNECTED_COMPONENTS , MST , DIJKSTRA ,
  RANDOM , RANDOM2 , RANDOM3 , RANDOM4 ,
  RANDOM5 , RANDOM6 , RANDOM7 , RANDOM8 ,
  RANDOM9 , RANDOM10 , RANDOM11 , RANDOM12 ,
  RANDOM13 , RANDOM14 , RANDOM15 , RANDOM16 ,
  FILE , NONE }

  Type of button.
- enum class DATA_STRUCTURES_TYPE {
  NONE , AVL , HASH_TABLE , GRAPH ,
  TRIE , MIN_HEAP , MAX_HEAP , BTREE_4TH }

  Type of data structures.
- enum class DATA_STRUCTURES_OPERATOR {
  INIT , INSERT , DELETE , SEARCH ,
  TOP , SIZE , SCC , MST ,
  DIJKSTRA , SETTING }

  Type of data structures operator.
- enum class HORIZONTAL_ALIGN { LEFT , CENTER , RIGHT }

  Align in horizon axis.
- enum class VERTICAL_ALIGN { TOP , CENTER , BOTTOM }

  Align in vertical axis.
- enum class INPUT_TYPE {
  NONE , INT , ARRAY , STRING ,
  STRINGS }

  Type of input.

Variables

- const std::string DISPLAY::HOME_ = "home"
- const std::string DISPLAY::WORKING_ = "working"
- const std::string PATH::ASSETS_ = "assets/"
- const std::string PATH::ASSETS::GRAPHICS_ = "assets/graphics/"
- const std::string PATH::ASSETS::FONTS_ = "assets/fonts/"
- const std::string PATH::ASSETS::SCRIPT_ = "assets/script/"
- const std::string PATH::ATTRIBUTE_ = "atb/"
- const std::string PATH::ATB::SPRITE_ = "atb/sprite/"
- const std::string PATH::ATB::OBJECT_ = "atb/object/"
- const std::string PATH::ATB::DISPLAY_ = "atb/display/"
- const std::string PATH::ATB::BUTTON_ = "atb/button/"
- const std::string PATH::ATB::DATA_STRUCTURES_ = "atb/data_structures/"
- const std::string PATH::ATB::INPUTBOX_ = "atb/input/"
- const std::string PATH::ATB::SCRIPT_ = "atb/script/"
- const std::string PATH::SAVING_ = "saving/"
- const std::string PATH::SAVING::AVL_ = "saving/AVL.txt"
- const std::string PATH::SAVING::HASH_TABLE_ = "saving/HASH_TABLE.txt"
- const std::string PATH::SAVING::GRAPH_ = "saving/GRAPH.txt"
- const std::string PATH::SAVING::TRIE_ = "saving/TRIE.txt"
- const std::string PATH::SAVING::MIN_HEAP_ = "saving/MIN_HEAP.txt"
- const std::string PATH::SAVING::MAX_HEAP_ = "saving/MAX_HEAP.txt"
- const std::string PATH::SAVING::BTREE_4TH_ = "saving/BTREE_4TH.txt"

8.11.1 Typedef Documentation

8.11.1.1 json

```
using json = nlohmann::json
```

Definition at line 10 of file GLOBAL.hpp.

8.11.2 Enumeration Type Documentation

Enumerator

8.11.2.1 BUTTON_ACTION

```
enum BUTTON_ACTION  [strong]
```

Type of button.

Enumerator

CHANGE_SCREEN	
INSERT	
DELETE	
INIT	
SEARCH	
SETTING	
DONE	
EDGES	
GO_BACK	
GO_NEXT	
GO_ON	
GO_OFF	
SPEED_UP	
SLOW_DOWN	
CLOSE	
TOP	
SIZE	
CONNECTED_COMPONENTS	
MST	
DIJKSTRA	
RANDOM	
RANDOM2	
RANDOM3	
RANDOM4	
RANDOM5	
RANDOM6	
RANDOM7	
RANDOM8	
RANDOM9	
RANDOM10	
RANDOM11	
RANDOM12	
RANDOM13	
RANDOM14	
RANDOM15	
RANDOM16	
FILE	
NONE	

Definition at line 44 of file GLOBAL.hpp.

```
{
    CHANGE_SCREEN,
    INSERT,
    DELETE,
    INIT,
    SEARCH,
    SETTING,
    DONE,
    EDGES,
    GO_BACK,
    GO_NEXT,
    GO_ON,
    GO_OFF,
    SPEED_UP,
    SLOW_DOWN,
    CLOSE,
    TOP,
    SIZE,
    CONNECTED_COMPONENTS,
    MST,
    DIJKSTRA,
    RANDOM,
    RANDOM2,
    RANDOM3,
    RANDOM4,
    RANDOM5,
    RANDOM6,
    RANDOM7,
    RANDOM8,
    RANDOM9,
    RANDOM10,
    RANDOM11,
    RANDOM12,
    RANDOM13,
    RANDOM14,
    RANDOM15,
    RANDOM16,
    FILE,
    NONE,
};
```

8.11.2.2 BUTTON_STATUS

`enum BUTTON_STATUS [strong]`

Status of button.

Enumerator

NONE	
HOVER	
RELEASED	
CLICKED	

Definition at line 34 of file GLOBAL.hpp.

```
{
    NONE,
    HOVER,
    RELEASED,
    CLICKED
};
```

8.11.2.3 DATA_STRUCTURES_OPERATOR

`enum DATA_STRUCTURES_OPERATOR [strong]`

Type of data structures operator.

Enumerator

INIT	
INSERT	
DELETE	
SEARCH	
TOP	
SIZE	
SCC	
MST	
DIJKSTRA	
SETTING	

Definition at line 103 of file GLOBAL.hpp.

```
{
    INIT,
    INSERT,
    DELETE,
    SEARCH,
    TOP,
    SIZE,
    SCC,
    MST,
    DIJKSTRA,
    SETTING
};
```

8.11.2.4 DATA_STRUCTURES_TYPE

```
enum DATA_STRUCTURES_TYPE  [strong]
```

Type of data structures.

Enumerator

NONE	
AVL	
HASH_TABLE	
GRAPH	
TRIE	
MIN_HEAP	
MAX_HEAP	
BTREE_4TH	

Definition at line 88 of file GLOBAL.hpp.

```
{
    NONE,
    AVL,
    HASH_TABLE,
    GRAPH,
    TRIE,
    MIN_HEAP,
    MAX_HEAP,
    BTREE_4TH
};
```

8.11.2.5 HORIZONTAL_ALIGN

```
enum HORIZONTAL_ALIGN  [strong]
```

Align in horizon axis.

Enumerator

LEFT	
CENTER	
RIGHT	

Definition at line 119 of file GLOBAL.hpp.

```
{
    LEFT,
    CENTER,
    RIGHT
};
```

8.11.2.6 INPUT_TYPE

```
enum INPUT_TYPE  [strong]
```

Type of input.

Enumerator

NONE	
INT	
ARRAY	
STRING	
STRINGS	

Definition at line 138 of file GLOBAL.hpp.

```
{
    NONE,
    INT,
    ARRAY,
    STRING,
    STRINGS
};
```

8.11.2.7 SHAPE

```
enum SHAPE  [strong]
```

Kind of simple shape.

Enumerator

NONE	

Enumerator

RECTANGLE	
CIRCLE	
TRIANGLE	
LINE	
POLYGON	

Definition at line 22 of file GLOBAL.hpp.
```
{
    NONE,
    RECTANGLE,
    CIRCLE,
    TRIANGLE,
    LINE,
    POLYGON
};
```

8.11.2.8 VERTICAL_ALIGN

`enum VERTICAL_ALIGN [strong]`

Align in vertical axis.

Enumerator

TOP	
CENTER	
BOTTOM	

Definition at line 128 of file GLOBAL.hpp.
```
{
    TOP,
    CENTER,
    BOTTOM
};
```

8.11.2.9 WINDOW_STATUS

`enum WINDOW_STATUS [strong]`

Status of window.

Enumerator

IS_OPEN	
IS_CLOSED	

Definition at line 14 of file GLOBAL.hpp.
```
{
    IS_OPEN,
    IS_CLOSED
};
```

8.12 include/inputbox.hpp File Reference

```
#include <string>
#include <vector>
#include <SDL2/SDL.h>
#include <object.hpp>
#include <sprite.hpp>
#include <button.hpp>
```

Classes

- class InputBox

  Register for user keyboard input.

8.13 include/object.hpp File Reference

```
#include <string>
#include <SDL2/SDL.h>
#include <SDL2/SDL_ttf.h>
#include <nlohmann/json.hpp>
#include <GLOBAL.hpp>
```

Classes

- struct Position

  location of an object in 2D coordinate
- class Object

  Class that represent shape, image from files, text. Smallest drawable unit.

Typedefs

- using json = nlohmann::json

8.13.1 Typedef Documentation

8.13.1.1 json

```
using json = nlohmann::json
```

Definition at line 12 of file object.hpp.

8.14 include/script.hpp File Reference

```
#include <iostream>
#include <string>
#include <vector>
#include <SDL2/SDL.h>
#include <SDL2/SDL_ttf.h>
#include <object.hpp>
#include <sprite.hpp>
#include <button.hpp>
```

Classes

- class Script

  Container that contains a pseudo-code.

8.15 include/services.hpp File Reference

```
#include <atomic>
#include <string>
#include <math.h>
#include <climits>
#include <random>
#include <chrono>
#include <nlohmann/json.hpp>
```

Namespaces

- JSON

  Interact with ∗.json files.
- FILEE

  Interact with text files.
- NUMBER

  Convert between string and interger.
- SIUSTRING

  Features for std::string.
- RANDOM

  Random intergers, doubles, strings generator.

Typedefs

- using json = nlohmann::json

Functions

- json ∗ JSON::readFile (std::string path)
- void JSON::saveFile (std::string path, json ∗data)
- std::vector< std::string > FILEE::readFile (std::string path)
- int64_t NUMBER::stringToInt (std::string s)
- std::string NUMBER::intToString (int64_t n)
- std::vector< int > NUMBER::stringToArray (std::string s)
- bool NUMBER::isDigit (char c)
- bool NUMBER::isLetter (char c)
- bool NUMBER::isSymbol (char c)
- bool NUMBER::isSign (char c)
- bool NUMBER::isOperator (char c)
- std::string NUMBER::removeLeadingZero (std::string s)
- bool NUMBER::isNumber (std::string s)
- bool NUMBER::isInInterval (std::string s, int64_t a, int64_t b)
- bool SIUSTRING::isSeparator (char c)
- std::vector< std::string > SIUSTRING::split (std::string s)
- int RANDOM::getInt (int a, int b)
- std::string RANDOM::getInt (int length, int a, int b)
- long long RANDOM::getLongLong (long long a, long long b)
- float RANDOM::getFloat (float a, float b)
- double RANDOM::getDouble (double a, double b)
- char RANDOM::getChar (char a, char b)
- char RANDOM::getChar ()
- std::string RANDOM::getString (int length)
- std::string RANDOM::getString (int length, char a, char b)
- bool RANDOM::flipCoin ()

Variables

- const int64_t NUMBER::INF = LLONG_MAX
- std::mt19937 RANDOM::rng = std::mt19937(std::chrono::steady_clock::now().time_since_epoch().count())

8.15.1 Typedef Documentation

8.15.1.1 json

```
using json = nlohmann::json
```

Definition at line 13 of file services.hpp.

8.16 include/sprite.hpp File Reference

```
#include <vector>
#include <SDL2/SDL.h>
#include <nlohmann/json.hpp>
#include <SDL2/SDL_ttf.h>
#include <object.hpp>
```

Classes

- class Sprite

  Object container Drawable.

Typedefs

- using json = nlohmann::json

8.16.1 Typedef Documentation

8.16.1.1 json

```
using json = nlohmann::json
```

Definition at line 15 of file sprite.hpp.

8.17 include/window.hpp File Reference

```
#include <vector>
#include <thread>
#include <mutex>
#include <condition_variable>
#include <map>
#include <queue>
#include <SDL2/SDL.h>
#include <GLOBAL.hpp>
#include <display.hpp>
#include <data_structures.hpp>
#include <inputbox.hpp>
```

Classes

- class MyWindow

  Window class class that create a window and manage it.

8.18 README.md File Reference

8.19 src/AVL/constructor.cpp File Reference

```
#include <algorithm>
#include <queue>
#include <data_structures/AVL.hpp>
```

8.20 src/btree4th/constructor.cpp File Reference

```
#include <data_structures/btree4th.hpp>
```

8.21 src/button/constructor.cpp File Reference

```
#include <iostream>
#include <button.hpp>
#include <GLOBAL.hpp>
#include <services.hpp>
```

8.22 src/data_structures/constructor.cpp File Reference

```
#include <data_structures.hpp>
```

8.23 src/display/constructor.cpp File Reference

```
#include <display.hpp>
#include <iostream>
```

8.24 src/graph/constructor.cpp File Reference

```
#include <data_structures/graph.hpp>
```

8.25 src/hash_table/constructor.cpp File Reference

```
#include <data_structures/hash_table.hpp>
#include <iostream>
#include <services.hpp>
```

8.26 src/inputbox/constructor.cpp File Reference

```
#include "GLOBAL.hpp"
#include <iostream>
#include <SDL2/SDL.h>
#include <inputbox.hpp>
#include <services.hpp>
```

8.27 src/maxheap/constructor.cpp File Reference

```
#include <data_structures/maxheap.hpp>
```

8.28 src/minheap/constructor.cpp File Reference

```
#include <data_structures/minheap.hpp>
```

8.29 src/object/constructor.cpp File Reference

```
#include <iostream>
#include <object.hpp>
#include <GLOBAL.hpp>
#include <services.hpp>
```

8.30 src/script/constructor.cpp File Reference

```
#include <script.hpp>
#include <services.hpp>
```

8.31 src/sprite/constructor.cpp File Reference

```
#include <fstream>
#include <iostream>
#include <sprite.hpp>
#include <GLOBAL.hpp>
#include <services.hpp>
```

8.32 src/trie/constructor.cpp File Reference

```
#include <data_structures/trie.hpp>
#include <queue>
```

8.33 src/window/constructor.cpp File Reference

```
#include <window.hpp>
#include <SDL2/SDL_image.h>
#include <SDL2/SDL_ttf.h>
```

8.34 src/AVL/destructor.cpp File Reference

```
#include <data_structures/AVL.hpp>
```

8.35 src/btree4th/destructor.cpp File Reference

```
#include <data_structures/btree4th.hpp>
```

8.36 src/button/destructor.cpp File Reference

```
#include <button.hpp>
```

8.37 src/data_structures/destructor.cpp File Reference

```
#include <data_structures.hpp>
```

8.38 src/display/destructor.cpp File Reference

```
#include <display.hpp>
```

8.39 src/graph/destructor.cpp File Reference

```
#include <data_structures/graph.hpp>
```

8.40 src/hash_table/destructor.cpp File Reference

```
#include <data_structures/hash_table.hpp>
```

8.41 src/inputbox/destructor.cpp File Reference

```
#include <inputbox.hpp>
```

8.42 src/maxheap/destructor.cpp File Reference

```
#include <data_structures/maxheap.hpp>
```

8.43 src/minheap/destructor.cpp File Reference

```
#include <data_structures/minheap.hpp>
```

8.44 src/object/destructor.cpp File Reference

```
#include <object.hpp>
```

8.45 src/script/destructor.cpp File Reference

```
#include <script.hpp>
```

8.46 src/sprite/destructor.cpp File Reference

```
#include <sprite.hpp>
```

8.47 src/trie/destructor.cpp File Reference

```
#include <data_structures/trie.hpp>
```

8.48 src/window/destructor.cpp File Reference

```
#include <window.hpp>
#include <SDL2/SDL_ttf.h>
#include <SDL2/SDL_image.h>
```

8.49 src/AVL/event.cpp File Reference

```
#include <data_structures/AVL.hpp>
```

8.50 src/button/event.cpp File Reference

```
#include <button.hpp>
```

8.51 src/data_structures/event.cpp File Reference

```
#include "GLOBAL.hpp"
#include <data_structures.hpp>
```

8.52 src/display/event.cpp File Reference

```
#include <display.hpp>
```

8.53 src/graph/event.cpp File Reference

```
#include <data_structures/graph.hpp>
```

8.54 src/hash_table/event.cpp File Reference

```
#include <data_structures/hash_table.hpp>
```

8.55 src/inputbox/event.cpp File Reference

```
#include <inputbox.hpp>
#include <iostream>
```

8.56 src/minheap/event.cpp File Reference

```
#include <data_structures/minheap.hpp>
```

8.57 src/script/event.cpp File Reference

```
#include <script.hpp>
```

8.58 src/trie/event.cpp File Reference

```
#include <data_structures/trie.hpp>
```

8.59 src/window/event.cpp File Reference

```
#include <window.hpp>
#include <services.hpp>
```

8.60 src/AVL/operator/delete.cpp File Reference

```
#include <data_structures/AVL.hpp>
#include <services.hpp>
```

8.61 src/AVL/operator/init.cpp File Reference

```
#include <data_structures/AVL.hpp>
#include <services.hpp>
```

8.62 src/btree4th/operator/init.cpp File Reference

```
#include <data_structures/btree4th.hpp>
```

8.63 src/data_structures/operator/AVL/init.cpp File Reference

```
#include <data_structures.hpp>
```

8.64 src/data_structures/operator/graph/init.cpp File Reference

```
#include <data_structures.hpp>
```

8.65 src/data_structures/operator/hash_table/init.cpp File Reference

```
#include <data_structures.hpp>
```

8.66 src/data_structures/operator/minheap/init.cpp File Reference

```
#include <data_structures.hpp>
```

8.67 src/data_structures/operator/trie/init.cpp File Reference

```
#include <data_structures.hpp>
```

8.68 src/graph/operator/init.cpp File Reference

```
#include <data_structures/graph.hpp>
#include <services.hpp>
```

Functions

- double sqr (double x)

8.68.1 Function Documentation

8.68.1.1 sqr()

```
double sqr (
            double x )
```

Definition at line 4 of file init.cpp.

```
{
    return x * x;
}
```

8.69 src/hash_table/operator/init.cpp File Reference

```
#include <iostream>
#include <data_structures/hash_table.hpp>
#include <services.hpp>
```

8.70 src/maxheap/operator/init.cpp File Reference

```
#include <data_structures/maxheap.hpp>
```

8.71 src/minheap/operator/init.cpp File Reference

```
#include <data_structures/minheap.hpp>
```

8.72 src/trie/operator/init.cpp File Reference

```
#include <data_structures/trie.hpp>
```

8.73 src/AVL/operator/insert.cpp File Reference

```
#include <data_structures.hpp>
#include <services.hpp>
```

8.74 src/btree4th/operator/insert.cpp File Reference

```
#include <data_structures/btree4th.hpp>
```

8.75 src/data_structures/operator/AVL/insert.cpp File Reference

```
#include <data_structures.hpp>
```

8.76 src/data_structures/operator/hash_table/insert.cpp File Reference

```
#include <data_structures.hpp>
```

8.77 src/data_structures/operator/minheap/insert.cpp File Reference

```
#include <data_structures.hpp>
```

8.78 src/data_structures/operator/trie/insert.cpp File Reference

```
#include <data_structures.hpp>
```

8.79 src/hash_table/operator/insert.cpp File Reference

```
#include <data_structures/hash_table.hpp>
#include <services.hpp>
```

8.80 src/maxheap/operator/insert.cpp File Reference

```
#include <data_structures/maxheap.hpp>
```

8.81 src/minheap/operator/insert.cpp File Reference

```
#include <data_structures/minheap.hpp>
#include <services.hpp>
```

8.82 src/trie/operator/insert.cpp File Reference

```
#include <data_structures/trie.hpp>
```

8.83 src/AVL/operator/search.cpp File Reference

```
#include <data_structures/AVL.hpp>
#include <services.hpp>
```

8.84 src/btree4th/operator/search.cpp File Reference

```
#include <data_structures/btree4th.hpp>
```

8.85 src/data_structures/operator/AVL/search.cpp File Reference

```
#include <data_structures.hpp>
```

8.86 src/data_structures/operator/hash_table/search.cpp File Reference

```
#include <data_structures.hpp>
```

8.87 src/data_structures/operator/trie/search.cpp File Reference

```
#include <data_structures.hpp>
```

8.88 src/hash_table/operator/search.cpp File Reference

```
#include <data_structures/hash_table.hpp>
```

8.89 src/trie/operator/search.cpp File Reference

```
#include <data_structures/trie.hpp>
```

8.90 src/AVL/rendering.cpp File Reference

```
#include <data_structures/AVL.hpp>
#include <queue>
```

8.91 src/button/rendering.cpp File Reference

```
#include <button.hpp>
#include <iostream>
```

8.92 src/data_structures/rendering.cpp File Reference

```
#include <data_structures.hpp>
```

8.93 src/display/rendering.cpp File Reference

```
#include <display.hpp>
#include <SDL2/SDL.h>
```

8.94 src/graph/rendering.cpp File Reference

```
#include <data_structures/graph.hpp>
```

Classes

- struct Point

8.95 src/hash_table/rendering.cpp File Reference

```
#include <data_structures/hash_table.hpp>
```

8.96 src/inputbox/rendering.cpp File Reference

```
#include <inputbox.hpp>
```

8.97 src/maxheap/rendering.cpp File Reference

8.98 src/minheap/rendering.cpp File Reference

```
#include <data_structures/minheap.hpp>
#include <cmath>
```

8.99 src/object/rendering.cpp File Reference

```
#include <object.hpp>
```

8.100 src/script/rendering.cpp File Reference

```
#include <script.hpp>
```

8.101 src/sprite/rendering.cpp File Reference

```
#include <sprite.hpp>
```

8.102 src/trie/rendering.cpp File Reference

```
#include <data_structures/trie.hpp>
#include <queue>
```

8.103 src/window/rendering.cpp File Reference

```
#include <window.hpp>
#include <services.hpp>
```

8.104 src/AVL/rotate.cpp File Reference

```
#include <data_structures/AVL.hpp>
```

8.105 src/AVL/step.cpp File Reference

```
#include <data_structures/AVL.hpp>
#include <chrono>
#include <thread>
```

8.106 src/data_structures/step.cpp File Reference

```
#include <data_structures.hpp>
```

8.107 src/hash_table/step.cpp File Reference

```
#include <data_structures/hash_table.hpp>
```

8.108 src/minheap/step.cpp File Reference

```
#include <data_structures/minheap.hpp>
#include <chrono>
#include <thread>
```

8.109 src/trie/step.cpp File Reference

```
#include <data_structures/trie.hpp>
#include <thread>
```

8.110 src/btree4th/node.cpp File Reference

```
#include <data_structures/btree4th.hpp>
```

8.111 src/btree4th/operator/addRecord.cpp File Reference

```
#include <data_structures/btree4th.hpp>
```

8.112 src/btree4th/operator/split.cpp File Reference

```
#include <data_structures/btree4th.hpp>
```

8.113 src/button/action.cpp File Reference

```
#include <button.hpp>
```

8.114 src/button/mouse_action.cpp File Reference

```
#include <button.hpp>
#include <iostream>
```

8.115 src/data_structures/AVL.cpp File Reference

8.116 src/data_structures/btree4th.cpp File Reference

8.117 src/data_structures/graph.cpp File Reference

8.118 src/data_structures/hash_table.cpp File Reference

8.119 src/data_structures/heap.cpp File Reference

8.120 src/data_structures/operator.cpp File Reference

```
#include "GLOBAL.hpp"
#include <data_structures.hpp>
```

8.121 src/inputbox/operator.cpp File Reference

```
#include <inputbox.hpp>
```

8.122 src/data_structures/operator/AVL/remove.cpp File Reference

```
#include <data_structures.hpp>
```

8.123 src/data_structures/operator/hash_table/remove.cpp File Reference

```
#include <data_structures.hpp>
```

8.124 src/data_structures/operator/trie/remove.cpp File Reference

```
#include <data_structures.hpp>
```

8.125 src/hash_table/operator/remove.cpp File Reference

```
#include <data_structures/hash_table.hpp>
```

8.126 src/maxheap/operator/remove.cpp File Reference

```
#include <data_structures/maxheap.hpp>
```

8.127 src/minheap/operator/remove.cpp File Reference

```
#include <data_structures/minheap.hpp>
```

Variables

- bool valuesReachedZero = false

8.127.1 Variable Documentation

8.127.1.1 valuesReachedZero

```
bool valuesReachedZero = false
```

Definition at line 3 of file remove.cpp.

8.128 src/trie/operator/remove.cpp File Reference

```
#include <data_structures/trie.hpp>
```

8.129 src/data_structures/operator/AVL/setting.cpp File Reference

```
#include <data_structures.hpp>
#include <services.hpp>
```

8.130 src/data_structures/operator/graph/setting.cpp File Reference

```
#include <data_structures.hpp>
#include <services.hpp>
```

8.131 src/data_structures/operator/hash_table/setting.cpp File Reference

```
#include <data_structures.hpp>
#include <services.hpp>
```

8.132 src/data_structures/operator/minheap/setting.cpp File Reference

```
#include <data_structures.hpp>
#include <services.hpp>
```

8.133 src/data_structures/operator/trie/setting.cpp File Reference

```
#include <data_structures.hpp>
#include <services.hpp>
```

8.134 src/data_structures/operator/graph/dijkstra.cpp File Reference

```
#include <data_structures.hpp>
```

8.135 src/graph/operator/dijkstra.cpp File Reference

```
#include <data_structures/graph.hpp>
```

Classes

- struct distanceHeap

8.136 src/data_structures/operator/graph/mst.cpp File Reference

```
#include <data_structures.hpp>
```

8.137 src/graph/operator/mst.cpp File Reference

```
#include <data_structures/graph.hpp>
#include <algorithm>
```

Classes

- struct DSU

8.138 src/data_structures/operator/graph/scc.cpp File Reference

```
#include <data_structures.hpp>
```

8.139 src/graph/operator/scc.cpp File Reference

```
#include <data_structures/graph.hpp>
#include <services.hpp>
```

8.140 src/data_structures/operator/minheap/pop.cpp File Reference

```
#include <data_structures.hpp>
```

8.141 src/data_structures/operator/minheap/size.cpp File Reference

8.142 src/maxheap/operator/size.cpp File Reference

```
#include <data_structures/maxheap.hpp>
```

8.143 src/minheap/operator/size.cpp File Reference

```
#include <data_structures/minheap.hpp>
```

8.144 src/data_structures/operator/minheap/top.cpp File Reference

8.145 src/data_structures/trie.cpp File Reference

8.146 src/graph/operator/repair.cpp File Reference

```
#include <data_structures/graph.hpp>
```

8.147 src/inputbox/typing.cpp File Reference

```
#include <inputbox.hpp>
```

8.148 src/main.cpp File Reference

```
#include <iostream>
#include <window.hpp>
```

Functions

- signed main ()

8.148.1 Function Documentation

8.148.1.1 main()

```
signed main ( )
```

Definition at line 4 of file main.cpp.

```
{
    MyWindow window;
    window.run();

    return 0;
}
```

8.149 src/maxheap/operator/getmax.cpp File Reference

```
#include <data_structures/maxheap.hpp>
```

8.150 src/maxheap/operator/heapify.cpp File Reference

```
#include <data_structures/maxheap.hpp>
```

8.151 src/minheap/operator/heapify.cpp File Reference

```
#include <data_structures/minheap.hpp>
```

8.152 src/minheap/operator/getmin.cpp File Reference

```
#include <climits>
#include <data_structures/minheap.hpp>
```

8.153 src/object/coloring.cpp File Reference

```
#include <object.hpp>
```

8.154 src/sprite/coloring.cpp File Reference

```
#include <sprite.hpp>
#include <iostream>
```

8.155 src/object/cropping.cpp File Reference

```
#include <object.hpp>
```

8.156 src/object/external_storage.cpp File Reference

```
#include <object.hpp>
#include <SDL2/SDL_image.h>
```

8.157 src/object/font.cpp File Reference

```
#include <object.hpp>
#include <iostream>
```

8.158 src/object/locating.cpp File Reference

```
#include <object.hpp>
```

8.159 src/sprite/locating.cpp File Reference

```
#include <sprite.hpp>
```

8.160 src/object/shape.cpp File Reference

```
#include <object.hpp>
#include <algorithm>
#include <iostream>
```

8.161 src/object/visible.cpp File Reference

```
#include <object.hpp>
```

8.162 src/script/highlight.cpp File Reference

```
#include <script.hpp>
```

8.163 src/services/file.cpp File Reference

```
#include <services.hpp>
#include <fstream>
#include <iostream>
```

8.164 src/services/json.cpp File Reference

```
#include <services.hpp>
#include <fstream>
#include <iostream>
```

8.165 src/services/number.cpp File Reference

```
#include <services.hpp>
#include <iostream>
```

Functions

- std::string removeLeadingZero (std::string s)

8.165.1 Function Documentation

8.165.1.1 removeLeadingZero()

```
std::string removeLeadingZero (
              std::string s )
```

Definition at line 115 of file number.cpp.

```
{
    if(!NUMBER::isNumber(s)) return "0";

    bool isNegative = (s[0] == '-');
    bool isSign = NUMBER::isSign(s[0]);

    int i = isSign;
    while(s[i] == '0') i++;

    if(i == (int) s.size()) return "0";

    std::string result = "";
    if(isNegative) result += '-';
    for(; i < (int) s.size(); i++) result += s[i];

    return result;
}
```

8.166 src/services/random.cpp File Reference

```
#include <services.hpp>
```

8.167 src/services/string.cpp File Reference

```
#include <services.hpp>
```

8.168 src/sprite/aligning.cpp File Reference

```
#include <sprite.hpp>
#include <iostream>
```

8.169 src/sprite/textbox.cpp File Reference

```
#include <sprite.hpp>
#include <services.hpp>
```

8.170 src/window/running.cpp File Reference

```
#include <window.hpp>
#include <services.hpp>
```

8.171 src/window/status.cpp File Reference

```
#include <window.hpp>
```

8.172 src/window/updating.cpp File Reference

```
#include <window.hpp>
#include <sstream>
```

Index