LSystems

v1.0.1

Generated by Doxygen 1.9.6

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

File Index

2.1 File List

Here is a list of all documented files with brief descriptions:

Chapter 3

Class Documentation

3.1 Drawer Class Reference

A class for drawing LSystems, processes and gives commands to Renderer.

```
#include <drawer.hpp>
```

Classes

- struct Node

    Node is position of Drawer on the plane, it is characterized by double coordinates and the angle of rotation (relative to the vertical)

Public Member Functions

- void turn_node (bool direction)

    Change rotation angle of stack's top.
- void move_node (Renderer &render, bool should_draw=true)

    Make a new Node, move to it and instructs Renderer to draw it directly with the top one.
- Drawer (std::string l_system_string, double line_length=10., double rotation_angle=90.)
- void Draw (Renderer &render)

    Processes the string and makes Renderer work: draws a segment (line) between top and new node.

Public Attributes

- std::string **l_system_**

    String generated by LSystem.
- std::stack< Node > **nodes_stack** = {}

    Stack of nodes.
- double **line_length_**

    Length of each segment.
- double **rotation_angle_**

    Angle of rotation (relative to the vertical)

3.1.1 Detailed Description

A class for drawing LSystems, processes and gives commands to Renderer.

3.1.2 Constructor & Destructor Documentation

3.1.2.1 Drawer()

```
Drawer::Drawer (
             std::string l_system_string,
             double line_length = 10.,
             double rotation_angle = 90.  )  [explicit]
```

Parameters

l_system_string	The out string of LSystem
line_length	The length of segment connecting two nodes; default is 5
rotation_angle	The angle of rotation (relative to the vertical); default is 0

3.1.3 Member Function Documentation

3.1.3.1 Draw()

```
void Drawer::Draw (
             Renderer & render )
```

Processes the string and makes Renderer work: draws a segment (line) between top and new node.

Parameters

render	Instance of class Renderer

Here is the call graph for this function:

3.2 finder_type Class Reference

A class for generating LSystem's string (method LSystem::GetString()); realize function find(), searches for matches in the range. See boost::find_format_all.

```
#include <lsystem.hpp>
```

Public Member Functions

- **finder_type** (LSystem::rules_container &rules)
- template< class iterator_type >
  boost::iterator_range< iterator_type > **operator()** (iterator_type begin, iterator_type end)

Private Attributes

- LSystem::rules_container & **rules_**

3.2.1 Detailed Description

A class for generating LSystem's string (method LSystem::GetString()); realize function find(), searches for matches in the range. See boost::find_format_all.

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

- include/lsystem/lsystem.hpp

3.3 formatter_type Class Reference

A class for generating LSystem's string (method LSystem::GetString()); realize function replace(), formats matches in the range. See boost::find_format_all.

```
#include <lsystem.hpp>
```

Public Member Functions

- **formatter_type** (LSystem::rules_container &rules)
- template< class iterator_type >
  boost::iterator_range< iterator_type > **operator()** (const boost::iterator_range< iterator_type > &range) const

Private Attributes

- LSystem::rules_container & **rules_**

3.3.1 Detailed Description

A class for generating LSystem's string (method LSystem::GetString()); realize function replace(), formats matches in the range. See boost::find_format_all.

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

- include/lsystem/lsystem.hpp

3.4 LSystem Class Reference

A class for LSystem, contains rules, axiom and number of generations.

```
#include <lsystem.hpp>
```

Public Types

- using **rules_container** = std::unordered_map< std::string, std::string >

  rules_container contains 2 elements: name of variable over which it will be performed and formula itself E. g. rule (FF_F[+FF][-FF]F[-F][+F]F) equals <"FF", "F[+FF][-FF]F[-F][+F]F">

Public Member Functions

- LSystem (const std::string &axiom, rules_container map_rules, int number_generations)

  LSystem is uniquely determined by an axiom and rules. If number of generations is 0, we have LSystem equals axiom; this is default option.
- std::string GetString ()

  Annul previous generations and generate new (what depends on new number of generations)

Private Member Functions

- void **generate** ()

  Calculate res_ variable; called in method GetString()

Private Attributes

- std::string **axiom_**

  Axiom of LSystem: it can include any letters and special symbols E. g: axiom = VZFFF.
- rules_container rules {}

  Rules container.
- int **num_gen** = 0

  In fact, it's a number of iterations of the rules.
- std::string **res**

  What LSystem outputs; depends on axiom (initially equal to it), rules and number of generations.

3.4.1 Detailed Description

A class for LSystem, contains rules, axiom and number of generations.

3.4.2 Constructor & Destructor Documentation

3.4.2.1 LSystem()

```
LSystem::LSystem (
                const std::string & axiom,
                rules_container map_rules,
                int number_generations )
```

LSystem is uniquely determined by an axiom and rules. If number of generations is 0, we have LSystem equals axiom; this is default option.

Parameters

in	*axiom*	
in	*map_rules*	
in	*number_generations*	

3.4.3 Member Function Documentation

3.4.3.1 GetString()

```
std::string LSystem::GetString ( )
```

Annul previous generations and generate new (what depends on new number of generations)

Returns

res_

Here is the call graph for this function:

3.4.4 Member Data Documentation

3.4.4.1 rules

```
rules_container LSystem::rules {} [private]
```

Rules container.

- E. g: V->[+++W][—W]YV

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

- include/lsystem/lsystem.hpp
- src/lsystem.cpp

3.5 Drawer::Node Struct Reference

Node is position of Drawer on the plane, it is characterized by double coordinates and the angle of rotation (relative to the vertical)

```
#include <drawer.hpp>
```

Public Attributes

- glm::dvec2 **point**
- double **angle** = 45.

3.5.1 Detailed Description

Node is position of Drawer on the plane, it is characterized by double coordinates and the angle of rotation (relative to the vertical)

The documentation for this struct was generated from the following file:

- include/lsystem/drawer.hpp

3.6 Reader Struct Reference

A struct for reading data for LSystems in two ways: from arguments of command line or file, and then passing them to LSystem constructor.

```
#include <reader.hpp>
```

Public Types

- using **rules_container** = std::unordered_map< std::string, std::string >

  rules_container contains 2 elements: name of variable over which it will be performed and formula itself E. g. rule "FF->F[+FF][-FF]F[-F][+F]F" equals <"FF", "F[+FF][-FF]F[-F][+F]F">

Public Member Functions

- void **CheckNumGen** () const

  Checking if NumGen is non-negative (>= 0)
- bool ParseCommandLine (int ac, char ∗av[ ])

  Reading and editing data based on command line arguments; syntax of rules: Variable->Rule.
- void **ParseFile** (const std::string &file_path)

  Reading and editing data from file with rules; syntax of rules: Variable->Rule; N. B: new rule is written from new line, you have to give absolute path to file.

Public Attributes

- rules_container **rules**

  A container with rules.
- std::string **axiom**

  An axiom of LSystem.
- int **numGen** = 0

  A number of generations; default it is 0, axiom.
- int **width** = 500

  Parameters of window.
- int **height** = 500
- double **lineLength** = 50.

  Length of the drawing system (command 'F')
- double **rotationAngle** = 90.

  Angle of rotation (see commands '+' and '-')

3.6.1 Detailed Description

A struct for reading data for LSystems in two ways: from arguments of command line or file, and then passing them to LSystem constructor.

3.6.2 Member Function Documentation

3.6.2.1 ParseCommandLine()

```
bool Reader::ParseCommandLine (
            int ac,
            char * av[] )
```

Reading and editing data based on command line arguments; syntax of rules: Variable->Rule.

Returns

Program termination signal

Here is the call graph for this function:

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

- include/lsystem/reader.hpp
- src/reader.cpp

3.7 Renderer Class Reference

A class for drawing LSystems, works with OpenGL.

```
#include <renderer.hpp>
```

Public Member Functions

- Renderer (int width=800, int height=800)

  Class constructor.
- ∼**Renderer** ()

  Class destructor: deleting program, buffers, terminate GLFW process.
- void Runtime (double upd=60., double fps=60.)

  The main program loop.
- void AddLine (glm::vec2 begin, glm::vec2 end)

  Add new section (2 vectors, begin and end)
- void **UpdateData** ()

  Create new data store for buffer of vertices and count number of vertices.

Private Member Functions

- void **render** ()
  Rendering.
- void **input** ()
  Event handling (keystrokes)
- void update (double duration)
  Updating zoom and camera offset.

Private Attributes

- GLFWwindow * **window** = nullptr
  Pointer to GLFW window.
- unsigned int **program** = 0
  ID of program.
- unsigned int **vertex_array** = 0
  ID of array with vertices.
- unsigned int **vertex_buffer** = 0
  ID of buffer with vertices.
- glm::mat4 **projection** {1.f}
  Projection matrix; default is identity.
- glm::mat4 **view** {1.f}
  View matrix; default is identity.
- int **matrix_location** = 0
  ID of matrix.
- std::vector< glm::vec2 > **vertices** {}
  Container of vertices.
- int **number_vertices** = 0
- glm::ivec2 **camera_shift** = {0, 0}
  Camera offset: needed to move across the canvas.
- float **camera_speed** = 0.05
- float **scale** = 1
  Zoom factor: default unit, of course.
- float **scale_speed** = 0.000005

3.7.1 Detailed Description

A class for drawing LSystems, works with OpenGL.

3.7.2 Constructor & Destructor Documentation

3.7.2.1 Renderer()

```
Renderer::Renderer (
             int width = 800,
             int height = 800 )  [explicit]
```

Class constructor.

Parameters

width	Width of window
height	Height of window

3.7.3 Member Function Documentation

3.7.3.1 AddLine()

```
void Renderer::AddLine (
            glm::vec2 begin,
            glm::vec2 end )
```

Add new section (2 vectors, begin and end)

Parameters

begin	Coordinates of begin
end	Coordinates of end

Here is the caller graph for this function:

3.7.3.2 Runtime()

```
void Renderer::Runtime (
            double upd = 60.,
            double fps = 60.  )
```

The main program loop.

Parameters

upd	Updates Per Second: how many processes update
fps	Frames Per Second: what is drawn to screen

Here is the call graph for this function:

3.7.3.3 update()

```
void Renderer::update (
            double duration )  [private]
```

Updating zoom and camera offset.

Parameters

duration	Button press time

Here is the caller graph for this function:

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

- include/lsystem/renderer.hpp
- src/renderer.cpp

Chapter 4

File Documentation

4.1 drawer.hpp

```cpp
//
// Created by one_eyed_john on 18/05/23.
//

#ifndef L_SYSTEMS_DRAWER_HPP
#define L_SYSTEMS_DRAWER_HPP

#include <lsystem/renderer.hpp>
#include <lsystem/lsystem.hpp>

#include <glad/glad.h>
#include <GLFW/glfw3.h>
#include <glm/vec2.hpp>

#include <stack>

class Drawer {
public:
    struct Node {
        glm::dvec2 point;
        double angle = 45.;
    };

public:
    std::string l_system_;

    std::stack<Node> nodes_stack = {};

    double line_length_;

    double rotation_angle_;

    void turn_node(bool direction);

    void move_node(Renderer &render, bool should_draw = true);

public:
    explicit Drawer(std::string l_system_string, double line_length = 10., double rotation_angle =
90.);

    ~Drawer() = default;

    void Draw(Renderer &render);
};

#endif //L_SYSTEMS_DRAWER_HPP
```

4.2 lsystem.hpp

```cpp
//
// Created by one_eyed_john on 26/04/23.
//

#ifndef L_SYSTEMS_LSYSTEM_HPP
```

```
#define L_SYSTEMS_LSYSTEM_HPP

#include <string>
#include <vector>
#include <unordered_map>
#include <boost/algorithm/string/find_format.hpp>

class LSystem {
public:
    using rules_container = std::unordered_map<std::string, std::string>;

private:
    std::string axiom_;

    rules_container rules{};

    int num_gen = 0;

    std::string res;

    void generate();

public:
    LSystem(const std::string &axiom, rules_container map_rules, int number_generations);

    ~LSystem() = default;

    std::string GetString();
};

class finder_type {
public:
    explicit finder_type(LSystem::rules_container &rules) : rules_(rules) {}

    ~finder_type() = default;

    template<class iterator_type>
    boost::iterator_range<iterator_type> operator()(iterator_type begin, iterator_type end) {
        for (auto &it = begin; it != end; ++it) {
            for (auto &rule: rules_) {
                auto length = static_cast<int>(rule.first.size());
                if (abs(std::distance(it, end)) >= length) {
                    if (std::string_view(it, it + length) == rule.first) {
                        return {it, it + length};
                    }
                } else {
                    continue;
                }
            }
        }
        return {end, end};
    }

private:
    LSystem::rules_container &rules_;
};

class formatter_type {
public:
    explicit formatter_type(LSystem::rules_container &rules) : rules_(rules) {}

    ~formatter_type() = default;

    template<class iterator_type>
    boost::iterator_range<iterator_type> operator()(const boost::iterator_range<iterator_type> &range)
const {
        auto it = rules_.find(std::string(std::begin(range), std::end(range)));
        return {it->second};
    }

private:
    LSystem::rules_container &rules_;
};

#endif //L_SYSTEMS_LSYSTEM_HPP
```

4.3 reader.hpp

```
//
// Created by one_eyed_john on 24/05/23.
//

#ifndef L_SYSTEMS_READER_HPP
```

```
#define L_SYSTEMS_READER_HPP

#include <filesystem>
#include <unordered_map>

struct Reader {
    using rules_container = std::unordered_map<std::string, std::string>;

    rules_container rules;

    std::string axiom;

    int numGen = 0;

    int width = 500;
    int height = 500;

    double lineLength = 50.;

    double rotationAngle = 90.;

    void CheckNumGen() const {
        if (numGen < 0)
            // Check if number of generations is non-negative, then cast to unsigned int
            throw std::invalid_argument("Number of generation must be non-negative!");
    }

    bool ParseCommandLine(int ac, char *av[]);

    void ParseFile(const std::string &file_path);
};

#endif //L_SYSTEMS_READER_HPP
```

4.4 renderer.hpp

```
//
// Created by one_eyed_john on 07/06/23.
//

#ifndef L_SYSTEMS_RENDERER_HPP
#define L_SYSTEMS_RENDERER_HPP

#include <vector>
#include <glm/vec2.hpp>
#include <glm/mat4x4.hpp>

struct GLFWwindow;

class Renderer {
public:
    explicit Renderer(int width = 800, int height = 800);

    ~Renderer();

    void Runtime(double upd = 60., double fps = 60.);

    void AddLine(glm::vec2 begin, glm::vec2 end);

    void UpdateData();

private:
    GLFWwindow *window = nullptr;

    unsigned int program = 0;

    unsigned int vertex_array = 0;

    unsigned int vertex_buffer = 0;

    glm::mat4 projection{1.f};

    glm::mat4 view{1.f};

    int matrix_location = 0;

    std::vector<glm::vec2> vertices{};
    int number_vertices = 0;

    glm::ivec2 camera_shift = {0, 0};
    float camera_speed = 0.05;

    float scale = 1;
    float scale_speed = 0.000005;

    void render();

    void input();

    void update(double duration);
};


#endif //L_SYSTEMS_RENDERER_HPP
```