3DViewer v2.0

Generated by Doxygen 1.9.1

Chapter 1

3DViewer v2.0

1.1 Table of Contents

1. Project Description
2. Features
   - Settings
   - Recording
3. Technical Requirements
4. Installation
5. Other functions
6. Authors

1.2 Project Description

3DViewer v2.0 is an application for visualizing 3D wireframe models, developed in C++ following object-oriented programming principles. This project is an improved version of the previous application (3DViewer v1.0) and includes new features, support for large datasets, and flexible interface settings.

1.3 Features

- Load 3D models in `.obj` format (vertices and face lists only).
- Perform the following affine transformations on the model:
  - Translate along the X, Y, Z axes.
  - Rotate around the X, Y, Z axes.
  - Scale the model.

- Graphical User Interface (GUI) that includes:
  - A button to select the model file and a field to display its name.
  - A visualization area for the wireframe model.
  - Control elements for translating, rotating, and scaling the model.
  - Information about the uploaded model (file name, number of vertices and faces).
- Support for models with vertex counts ranging from to 1,000,000 and more without noticeable delays (interface freeze does not exceed 0.5 seconds).
- Implementation based on the MVC architectural pattern.
- Use of three different design patterns (Facade, Builder, Singleton).

1.4 Additional Features

1.4.1 Settings

- Choose the type of projection (parallel or central).
- Configure the type (solid, dashed), color, and thickness of edges, as well as the shape, color, and size of vertices.
- Change the background color.
- Save user settings between program runs.

1.4.2 Recording

- Save rendered model images in `.bmp` and `.jpeg` formats.
- Record model transformation animations as GIF files (10 FPS, 5 seconds).

1.5 Technical Requirements

- Language standard: C++17.
- Build with `Makefile` containing standard targets (`all`, `install`, `uninstall`, `clean`, `tests`, etc.).
- Code follows Google Style guidelines.
- Supported GUI librarie - Qt.

1.6 Installation

1. Clone the repository:
   ```
   git clone <repository-link>
   cd <repository-link>
   ```
2. Install:
   ```
   cd src
   make install
   ```
3. Run:
   ```
   make run
   ```
4. Uninstall:
   ```
   make uninstall
   ```

1.7 Other functions

Also there's some other functions which You can use. You should be in the src folder.

1. Testing:
   `make tests`
2. Valgrind testing:
   `make valgrind_test`
3. Make archive with program:
   `make dist`
4. Make documentation with html and pdf versions (making new folder - documentation):
   `make dvi`
5. Make gcov report with coverage report (making new folder - report):
   `make gcov_report`
6. Make UML-diagram:
   `make uml_diagram`

1.8 Authors

School 21 students:

- Buggkell
- Montoyay

2025

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

s21 . . . . . ??
Ui . . . . . ??

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

s21::Controller . . . . . ??
s21::Edge . . . . . ??
s21::Facade . . . . . ??
s21::FacadeOperationResult . . . . . ??
s21::Face . . . . . ??
s21::FileReader . . . . . ??
s21::FileReaderBuilder . . . . . ??
  s21::FigureBuilder . . . . . ??
s21::FileReaderDirector . . . . . ??
s21::NormalizationParameters . . . . . ??
s21::PairHash . . . . . ??
s21::Point . . . . . ??
s21::View::PrevPositions . . . . . ??
QDialog
  s21::SettingsWidget . . . . . ??
QMainWindow
  s21::MainWindow . . . . . ??
QOpenGLFunctions
  s21::QtSceneDrawer . . . . . ??
QOpenGLWidget
  s21::QtSceneDrawer . . . . . ??
QWidget
  s21::View . . . . . ??
s21::QtSceneDrawer::RendParams . . . . . ??
s21::SceneDrawerBase . . . . . ??
  s21::QtSceneDrawer . . . . . ??
s21::SceneObject . . . . . ??
  s21::Figure . . . . . ??
  s21::Vertex . . . . . ??
s21::TransformMatrix . . . . . ??
s21::TransformMatrixBuilder . . . . . ??

Chapter 4

Class Index

4.1 Class List

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

s21::Controller
: Provides an interface for interacting with Model in MVC pattern . . . ??

s21::Edge
: Represents an edge in a figure, connecting two vertices . . . ??

s21::Facade
: Represents interface for interacting with Controller class . . . ??

s21::FacadeOperationResult
: Represents the result of an operation performed by the Facade class . . . ??

s21::Face
: Represents a face composed of a vector of unsigned integers . . . ??

s21::Figure
: Represents a figure in a 3D scene, derived from SceneObject . . . ??

s21::FigureBuilder
: A concrete builder for creating Figure objects . . . ??

s21::FileReader
: A class for reading figure data from a file . . . ??

s21::FileReaderBuilder
: Abstract base class for building a FileReader. Use Builder pattern . . . ??

s21::FileReaderDirector
: Constructs a figure using the provided builder . . . ??

s21::MainWindow . . . ??

s21::NormalizationParameters
: Manages the normalization parameters for Figure objects . . . ??

s21::PairHash
: Hash function for pairs of values. This struct provides a hash function for pairs of values, used to store edges in a hash set . . . ??

s21::Point
: Represents a point in 3D space with homogeneous coordinates . . . ??

s21::View::PrevPositions
: Keeps track of the previous positions of the scale, move, and rotate 3D object . . . ??

s21::QtSceneDrawer
: The QtSceneDrawer is an OpenGL-based widget for rendering 3D figures . . . ??

s21::QtSceneDrawer::RendParams
: Struct contains rendering parameters for the 3D scene (projection type, background color, edge and vertex rendering styles and colors) . . . ??

s21::SceneDrawerBase
Abstract base class for scene drawers . . . . . ??
s21::SceneObject
Abstract base class representing a scene object . . . . . ??
s21::SettingsWidget
A dialog widget that allows user to customize type, color and thickness of the edges, display method, color and size of the vertices of a 3D model . . . . . ??
s21::TransformMatrix
A class representing a 4x4 transformation matrix . . . . . ??
s21::TransformMatrixBuilder
A utility class for creating transformation matrices . . . . . ??
s21::Vertex
Represents a vertex in 3D space, derived from SceneObject . . . . . ??
s21::View
Responsible for the GUI of the 3D viewer application and handling user interactions, such as opening and saving files, adjusting the 3D object's scale, position, and rotation, switching between parallel and central projection modes. It also includes a QtSceneDrawer object to render the 3D scene and a SettingsWidget object to manage the application's settings . . . . . ??

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

gui/main.cpp . . . ??
gui/mainwindow.cpp . . . ??
gui/mainwindow.h . . . ??
gui/controller/controller.cpp . . . ??
gui/controller/controller.h . . . ??
gui/view/qt_scene_drawer.cpp . . . ??
gui/view/qt_scene_drawer.h . . . ??
gui/view/settings_widget.cpp . . . ??
gui/view/settings_widget.h . . . ??
gui/view/view.cpp . . . ??
gui/view/view.h . . . ??
gui/view/widget_utils.cpp . . . ??
gui/view/widget_utils.h . . . ??
model/facade/facade.cpp . . . ??
model/facade/facade.h . . . ??
model/facade/normalization_parameters.h . . . ??
model/facade/scene_drawer_base.h . . . ??
model/figure/edge.cpp . . . ??
model/figure/edge.h . . . ??
model/figure/face.h . . . ??
model/figure/figure.cpp . . . ??
model/figure/figure.h . . . ??
model/figure/scene_object.h . . . ??
model/figure/vertex.cpp . . . ??
model/figure/vertex.h . . . ??
model/file_reader/file_reader.cpp . . . ??
model/file_reader/file_reader.h . . . ??
model/transform_matrix/transform_matrix.cpp . . . ??
model/transform_matrix/transform_matrix.h . . . ??

Chapter 6

Namespace Documentation

6.1 s21 Namespace Reference

Classes

- class FacadeOperationResult

  Represents the result of an operation performed by the Facade class.
- class Facade

  Represents interface for interacting with Controller class.
- class NormalizationParameters

  Manages the normalization parameters for Figure objects.
- class SceneDrawerBase

  Abstract base class for scene drawers.
- class Edge

  Represents an edge in a figure, connecting two vertices.
- class Face

  Represents a face composed of a vector of unsigned integers.
- struct PairHash

  Hash function for pairs of values. This struct provides a hash function for pairs of values, used to store edges in a hash set.
- class Figure

  Represents a figure in a 3D scene, derived from SceneObject.
- class SceneObject

  Abstract base class representing a scene object.
- struct Point

  Represents a point in 3D space with homogeneous coordinates.
- class Vertex

  Represents a vertex in 3D space, derived from SceneObject.
- class FileReader

  A class for reading figure data from a file.
- class FileReaderBuilder

  Abstract base class for building a FileReader. Use Builder pattern.
- class FigureBuilder

  A concrete builder for creating Figure objects.
- class FileReaderDirector

  Constructs a figure using the provided builder.

- class TransformMatrix

  A class representing a 4x4 transformation matrix.
- class TransformMatrixBuilder

  A utility class for creating transformation matrices.
- class Controller

  Provides an interface for interacting with Model in MVC pattern.
- class MainWindow
- class QtSceneDrawer

  The QtSceneDrawer is an OpenGL-based widget for rendering 3D figures.
- class SettingsWidget

  A dialog widget that allows user to customize type, color and thickness of the edges, display method, color and size of the vertices of a 3D model.
- class View

  The View class is responsible for the GUI of the 3D viewer application and handling user interactions, such as opening and saving files, adjusting the 3D object's scale, position, and rotation, switching between parallel and central projection modes. It also includes a QtSceneDrawer object to render the 3D scene and a SettingsWidget object to manage the application's settings.

Typedefs

- typedef struct s21::Point Point

Functions

- QPushButton * setup_button (QWidget *parent, const QString &text, const QString &icon_path, const QSize &icon_size, int x, int y, int width, int height, const QFont &font)

  Creates a new QPushButton with the specified text, icon, geometry, and font. If an icon path is provided, the button will display the icon with the given size.
- QLabel * setup_label (QWidget *parent, const QString &text, int x, int y, int width, int height, const QFont &font, Qt::Alignment align)

  Creates a new QLabel with the specified text, geometry, font, and alignment.
- QScrollBar * setup_scrollbar (QWidget *parent, Qt::Orientation orient, int x, int y, int width, int height, int min, int max)

  Creates a new QScrollBar with the specified orientation, geometry, and range.
- QSpinBox * setup_spinbox (QWidget *parent, int x, int y, int width, int height, int min, int max)

  Creates a new QSpinBox with the specified geometry and range.
- QComboBox * setup_combobox (QWidget *parent, const QStringList &options, int x, int y, int width, int height, const QFont &font)

  Creates a new QComboBox with the specified parent, options, geometry, and font.

6.1.1 Typedef Documentation

6.1.1.1 Point

```
typedef struct Point s21::Point
```

6.1.2 Function Documentation

6.1.2.1 setup_button()

```
QPushButton * s21::setup_button (
            QWidget * parent,
            const QString & text,
            const QString & icon_path,
            const QSize & icon_size,
            int x,
            int y,
            int width,
            int height,
            const QFont & font )
```

Creates a new QPushButton with the specified text, icon, geometry, and font. If an icon path is provided, the button will display the icon with the given size.

Parameters

parent	The parent widget for the button.
text	The text to be displayed on the button.
icon_path	The path to the icon image file, or an empty string if no icon is desired.
icon_size	The size of the icon to be displayed on the button.
x	The x-coordinate of the button's position.
y	The y-coordinate of the button's position.
width	The width of the button.
height	The height of the button.
font	The font to be used for the button's text.

Returns

A pointer to the newly created QPushButton.

6.1.2.2 setup_combobox()

```
QComboBox * s21::setup_combobox (
            QWidget * parent,
            const QStringList & options,
            int x,
            int y,
            int width,
            int height,
            const QFont & font )
```

Creates a new QComboBox with the specified parent, options, geometry, and font.

Parameters

parent	The parent widget for the combo box.
options	The list of options to populate the combo box.
x	The x-coordinate of the combo box's position.
y	The y-coordinate of the combo box's position.
width	The width of the combo box.
height	The height of the combo box.
font	The font to use for the combo box's text.

Returns

A pointer to the newly created QComboBox.

6.1.2.3 setup_label()

```
QLabel * s21::setup_label (
             QWidget * parent,
             const QString & text,
             int x,
             int y,
             int width,
             int height,
             const QFont & font,
             Qt::Alignment align )
```

Creates a new QLabel with the specified text, geometry, font, and alignment.

Parameters

parent	The parent widget for the label.
text	The text to be displayed on the label.
x	The x-coordinate of the label's position.
y	The y-coordinate of the label's position.
width	The width of the label.
height	The height of the label.
font	The font to be used for the label's text.
align	The alignment of the label's text.

Returns

A pointer to the newly created QLabel.

6.1.2.4 setup_scrollbar()

```
QScrollBar * s21::setup_scrollbar (
             QWidget * parent,
```

```
            Qt::Orientation orient,
            int x,
            int y,
            int width,
            int height,
            int min,
            int max )
```

Creates a new QScrollBar with the specified orientation, geometry, and range.

Parameters

parent	The parent widget for the scroll bar.
orient	The orientation of the scroll bar (Qt::Horizontal or Qt::Vertical).
x	The x-coordinate of the scroll bar's position.
y	The y-coordinate of the scroll bar's position.
width	The width of the scroll bar.
height	The height of the scroll bar.
min	The minimum value of the scroll bar's range.
max	The maximum value of the scroll bar's range.

Returns

A pointer to the newly created QScrollBar.

6.1.2.5 setup_spinbox()

```
QSpinBox * s21::setup_spinbox (
            QWidget * parent,
            int x,
            int y,
            int width,
            int height,
            int min,
            int max )
```

Creates a new QSpinBox with the specified geometry and range.

Parameters

parent	The parent widget for the spin box.
x	The x-coordinate of the spin box's position.
y	The y-coordinate of the spin box's position.
width	The width of the spin box.
height	The height of the spin box.
min	The minimum value of the spin box's range.
max	The maximum value of the spin box's range.

Returns

A pointer to the newly created QSpinBox.

6.2 Ui Namespace Reference

Chapter 7

Class Documentation

7.1 s21::Controller Class Reference

Provides an interface for interacting with Model in MVC pattern.

`#include <controller.h>`

Public Member Functions

- Controller (Facade *facade)

  Constructs a new Controller instance with the given Facade.
- ~Controller ()=default
- void scale_model (float x)

  Scales the model by the given factor in all three dimensions.
- void move_model (float x, float y, float z)

  Moves the model by the given offsets in the x, y, and z dimensions.
- void rotate_model (float x, float y, float z)

  Rotates the model by the given angles around the x, y, and z axes.
- void read_model (std::string file_path)

  Loads a wireframe model from given file path and puts it in the Facade.
- void reset_model_pos ()

  Resets the position of the 3D model to its start position.
- std::vector< int > get_model_data ()

  Returns a vector containing the number of vertices, total edges, and unique edges in the 3D model.
- Figure * get_figure ()

  Returns pointer to Figure object from Facade for drawing by QtSceneDrawer.

7.1.1 Detailed Description

Provides an interface for interacting with Model in MVC pattern.

The Controller class is responsible for managing the interaction between the GUI and the underlying 3D model. It provides methods for scaling, moving, rotating, and reading the model, as well as resetting its position and retrieving its data.

7.1.2 Constructor & Destructor Documentation

7.1.2.1 Controller()

```
s21::Controller::Controller (
            Facade * facade )
```

Constructs a new Controller instance with the given Facade.

Parameters

facade	Pointer to Facade object for using by Controller.

7.1.2.2 ~Controller()

```
s21::Controller::~Controller ( )  [default]
```

7.1.3 Member Function Documentation

7.1.3.1 get_figure()

```
Figure * s21::Controller::get_figure ( )
```

Returns pointer to Figure object from Facade for drawing by QtSceneDrawer.

Returns

A pointer to Figure object.

7.1.3.2 get_model_data()

```
std::vector< int > s21::Controller::get_model_data ( )
```

Returns a vector containing the number of vertices, total edges, and unique edges in the 3D model.

Returns

A vector of integers representing the vertex count, total edge count, and unique edge count of the 3D model.

7.1.3.3 move_model()

```
void s21::Controller::move_model (
            float x,
            float y,
            float z )
```

Moves the model by the given offsets in the x, y, and z dimensions.

Parameters

x	The offset to apply to the model's x-coordinate.
y	The offset to apply to the model's y-coordinate.
z	The offset to apply to the model's z-coordinate.

7.1.3.4 read_model()

```
void s21::Controller::read_model (
            std::string file_path )
```

Loads a wireframe model from given file path and puts it in the Facade.

Parameters

file_path	The file path for loading the 3d model.

7.1.3.5 reset_model_pos()

```
void s21::Controller::reset_model_pos ( )
```

Resets the position of the 3D model to its start position.

7.1.3.6 rotate_model()

```
void s21::Controller::rotate_model (
            float x,
            float y,
            float z )
```

Rotates the model by the given angles around the x, y, and z axes.

Parameters

x	The angle (in degrees) to rotate the model around the x-axis.
y	The angle (in degrees) to rotate the model around the y-axis.
z	The angle (in degrees) to rotate the model around the z-axis.

7.1.3.7 scale_model()

```
void s21::Controller::scale_model (
            float x )
```

Scales the model by the given factor in all three dimensions.

Parameters

x	The scaling factor to apply to the model.

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

- gui/controller/controller.h
- gui/controller/controller.cpp

7.2 s21::Edge Class Reference

Represents an edge in a figure, connecting two vertices.

`#include <edge.h>`

Public Member Functions

- Edge (Vertex &begin, Vertex &end)
  Constructs an Edge object connecting two vertices.
- ~Edge ()=default
  Default destructor.
- Vertex & get_begin () const
  Gets the starting vertex of the edge.
- Vertex & get_end () const
  Gets the ending vertex of the edge.
- bool operator== (const Edge &other) const
  Equality comparison operator for Edge objects.
- Edge & operator= (const Edge &other)
  Assignment operator for Edge objects.

7.2.1 Detailed Description

Represents an edge in a figure, connecting two vertices.

This class models a directed edge in a graph structure. Each edge has a starting vertex (`begin_`) and an ending vertex (`end_`).

Note

In test mode (`#ifdef TEST`), the member variables are public for testing purposes.

7.2.2 Constructor & Destructor Documentation

7.2.2.1 Edge()

```
s21::Edge::Edge (
            Vertex & begin,
            Vertex & end )  [inline]
```

Constructs an Edge object connecting two vertices.

Parameters

begin	The starting vertex of the edge.
end	The ending vertex of the edge.

7.2.2.2 ~Edge()

```
s21::Edge::~Edge ( )  [default]
```

Default destructor.

7.2.3 Member Function Documentation

7.2.3.1 get_begin()

```
Vertex& s21::Edge::get_begin ( ) const  [inline]
```

Gets the starting vertex of the edge.

Returns

A reference to the starting vertex.

7.2.3.2 get_end()

```
Vertex& s21::Edge::get_end ( ) const  [inline]
```

Gets the ending vertex of the edge.

Returns

A reference to the ending vertex.

7.2.3.3 operator=()

```
Edge & s21::Edge::operator= (
            const Edge & other )
```

Assignment operator for Edge objects.

This operator assigns the values of another Edge object to this object. It performs a self-assignment check to ensure that assigning an object to itself does not cause any issues.

Parameters

other	The Edge object to assign from.

Returns

A reference to this Edge object after assignment.

7.2.3.4 operator==()

```
bool s21::Edge::operator== (
            const Edge & other ) const
```

Equality comparison operator for Edge objects.

This operator checks if two Edge objects are equal. Two edges are considered equal if their begin and end points are the same, regardless of the order (i.e., an edge from A to B is considered equal to an edge from B to A).

Parameters

other	The Edge object to compare with.

Returns

true if the edges are equal, false otherwise.

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

- model/figure/edge.h
- model/figure/edge.cpp

7.3 s21::Facade Class Reference

Represents interface for interacting with Controller class.

```
#include <facade.h>
```

Public Member Functions

- Facade ()
- Facade (SceneDrawerBase ∗sceneDrawer)
- ∼Facade ()=default
- FacadeOperationResult load_figure (std::string path)

  Loads a wireframe model from the specified file path.
- FacadeOperationResult reset_figure ()

  Deletes old figure object and creates new one.
- FacadeOperationResult move_figure (float x, float y, float z)

  Moves the figure by the specified x, y, and z coordinates.
- FacadeOperationResult rotate_figure (float x, float y, float z)

  Rotates the figure by the specified x, y, and z angles.
- FacadeOperationResult scale_figure (float x, float y, float z)

  Scales the figure by the specified x, y, and z factors.
- int get_vertices_count ()
- int get_unique_edges_count ()
- int get_all_edges_count ()
- Figure ∗ get_figure ()
- void reset_figure_pos ()

7.3.1 Detailed Description

Represents interface for interacting with Controller class.

The Facade class is central point of access to model part of MVC pattern. Contains FileReader, Figure and SceneDrawerBase. It provides API for operations like loading, moving, rotating, and scaling a 3D figure.

7.3.2 Constructor & Destructor Documentation

7.3.2.1 Facade() [1/2]

```
s21::Facade::Facade ( )  [inline]
```

7.3.2.2 Facade() [2/2]

```
s21::Facade::Facade (
            SceneDrawerBase * sceneDrawer )  [inline]
```

7.3.2.3 ∼Facade()

```
s21::Facade::~Facade ( )  [default]
```

7.3.3 Member Function Documentation

7.3.3.1 get_all_edges_count()

```
int s21::Facade::get_all_edges_count ( )  [inline]
```

7.3.3.2 get_figure()

```
Figure* s21::Facade::get_figure ( )  [inline]
```

7.3.3.3 get_unique_edges_count()

```
int s21::Facade::get_unique_edges_count ( )  [inline]
```

7.3.3.4 get_vertices_count()

```
int s21::Facade::get_vertices_count ( )  [inline]
```

7.3.3.5 load_figure()

```
FacadeOperationResult s21::Facade::load_figure (
            std::string path )
```

Loads a wireframe model from the specified file path.

Parameters

path	The file path of figure to load.

Returns

A FacadeOperationResult object containing the result of the operation.

7.3.3.6 move_figure()

```
FacadeOperationResult s21::Facade::move_figure (
            float x,
            float y,
            float z )
```

Moves the figure by the specified x, y, and z coordinates.

Parameters

x	The x-coordinate to move the figure by.
y	The y-coordinate to move the figure by.
z	The z-coordinate to move the figure by.

Returns

A FacadeOperationResult object containing the result of the operation.

7.3.3.7 reset_figure()

```
FacadeOperationResult s21::Facade::reset_figure ( )
```

Deletes old figure object and creates new one.

Returns

A FacadeOperationResult object containing the result of the operation.

7.3.3.8 reset_figure_pos()

```
void s21::Facade::reset_figure_pos ( )  [inline]
```

7.3.3.9 rotate_figure()

```
FacadeOperationResult s21::Facade::rotate_figure (
            float x,
            float y,
            float z )
```

Rotates the figure by the specified x, y, and z angles.

Parameters

x	The angle to rotate the figure around the x-axis.
y	The angle to rotate the figure around the y-axis.
z	The angle to rotate the figure around the z-axis.

Returns

A FacadeOperationResult object containing the result of the operation.

7.3.3.10 scale_figure()

```
FacadeOperationResult s21::Facade::scale_figure (
            float x,
            float y,
            float z )
```

Scales the figure by the specified x, y, and z factors.

Parameters

x	The scaling factor for the x-axis.
y	The scaling factor for the y-axis.
z	The scaling factor for the z-axis.

Returns

A FacadeOperationResult object containing the result of the operation.

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

- model/facade/facade.h
- model/facade/facade.cpp

7.4 s21::FacadeOperationResult Class Reference

Represents the result of an operation performed by the Facade class.

`#include <facade.h>`

Public Member Functions

- FacadeOperationResult ()
- ~FacadeOperationResult ()=default
- std::string GetErrorMessage () const
- bool IsSuccess () const
- void SetErrorMessage (std::string message)
- void SetSuccess (bool status)

7.4.1 Detailed Description

Represents the result of an operation performed by the Facade class.

This class contains the success or failure of an operation and optional error message.

7.4.2 Constructor & Destructor Documentation

7.4.2.1 FacadeOperationResult()

```
s21::FacadeOperationResult::FacadeOperationResult ( )  [inline]
```

7.4.2.2 ~FacadeOperationResult()

```
s21::FacadeOperationResult::~FacadeOperationResult ( )  [default]
```

7.4.3 Member Function Documentation

7.4.3.1 GetErrorMessage()

```
std::string s21::FacadeOperationResult::GetErrorMessage ( ) const  [inline]
```

7.4.3.2 IsSuccess()

```
bool s21::FacadeOperationResult::IsSuccess ( ) const  [inline]
```

7.4.3.3 SetErrorMessage()

```
void s21::FacadeOperationResult::SetErrorMessage (
            std::string message )  [inline]
```

7.4.3.4 SetSuccess()

```
void s21::FacadeOperationResult::SetSuccess (
            bool status )  [inline]
```

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

- model/facade/facade.h

7.5 s21::Face Class Reference

Represents a face composed of a vector of unsigned integers.

`#include <face.h>`

Public Member Functions

- Face ()

  Default constructor for the Face class. Initializes an empty face.
- Face (std::vector< unsigned > face)

  Parameterized constructor for the Face class. Initializes the face with the provided vector of unsigned integers.
- ~Face ()=default

  Destructor for the Face class. Default destructor.
- std::vector< unsigned > get_face () const

  Getter for the face vector.
- bool operator== (const Face &other) const

  Equality comparison operator for Face objects. Compares two Face objects for equality based on their internal face vectors.

7.5.1 Detailed Description

Represents a face composed of a vector of unsigned integers.

The `Face` class encapsulates a face, which is represented as a vector of unsigned integers.

Note

In test mode (`#ifdef TEST`), the member variables are public for testing purposes.

7.5.2 Constructor & Destructor Documentation

7.5.2.1 Face() [1/2]

```
s21::Face::Face ( )  [inline]
```

Default constructor for the Face class. Initializes an empty face.

7.5.2.2 Face() [2/2]

```
s21::Face::Face (
            std::vector< unsigned > face )  [inline]
```

Parameterized constructor for the Face class. Initializes the face with the provided vector of unsigned integers.

Parameters

face	A vector of unsigned integers representing the face.

7.5.2.3 ~Face()

```
s21::Face::~Face ( )  [default]
```

Destructor for the Face class. Default destructor.

7.5.3 Member Function Documentation

7.5.3.1 get_face()

```
std::vector<unsigned> s21::Face::get_face ( ) const  [inline]
```

Getter for the face vector.

Returns

A copy of the vector representing the face.

7.5.3.2 operator==()

```
bool s21::Face::operator== (
            const Face & other ) const  [inline]
```

Equality comparison operator for Face objects. Compares two Face objects for equality based on their internal face vectors.

Parameters

other	The Face object to compare with.

Returns

true if the face vectors are equal, false otherwise.

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

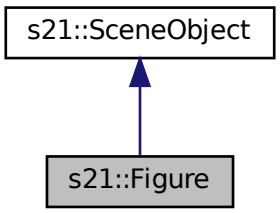
- model/figure/face.h

7.6 s21::Figure Class Reference

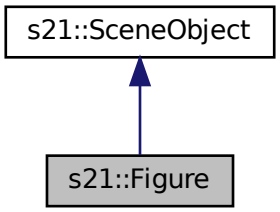
Represents a figure in a 3D scene, derived from SceneObject.

`#include <figure.h>`

Inheritance diagram for s21::Figure:



Collaboration diagram for s21::Figure:



Public Member Functions

- Figure ()=default

  Default constructor for the Figure class.
- ~Figure ()=default

  Default destructor for the Figure class.
- void set_name (std::string name)

  Sets the name of the figure.
- std::vector< Vertex > get_vertices () const

  Gets the vertices of the figure.
- std::string get_name () const

  Gets the name of the figure.
- std::vector< Edge > get_edges () const

Gets the edges of the figure.

- std::vector< Face > get_faces () const

  Gets the faces of the figure.

- unsigned long long get_all_edges_count () const

  Gets the total count of edges in the figure.

- void transform (const TransformMatrix &matrix) override

  Transforms the figure's vertices using the provided transformation matrix. This method applies the transformation matrix to each vertex in the figure.

- void make_edges ()

  Generates the edges of the figure based on its vertices and faces.

- void save_default_coords ()

  Saves the current vertices as the default (original) coordinates.

- void get_default_coords ()

  Restores the vertices to their default (original) coordinates.

Friends

- class FileReader
- class Facade

7.6.1 Detailed Description

Represents a figure in a 3D scene, derived from SceneObject.

The `Figure` class encapsulates a 3D figure, including its vertices, faces, edges, and name. It provides functionality for transforming the figure, printing its properties, and managing its geometry. The class is a friend of `FileReader` and `Facade` to allow controlled access to its private members.

Note

In test mode (`#ifdef TEST`), the member variables are public for testing purposes.

7.6.2 Constructor & Destructor Documentation

7.6.2.1 Figure()

```
s21::Figure::Figure ( )  [default]
```

Default constructor for the Figure class.

7.6.2.2 ~Figure()

```
s21::Figure::~Figure ( )  [default]
```

Default destructor for the Figure class.

7.6.3 Member Function Documentation

7.6.3.1 get_all_edges_count()

```
unsigned long long s21::Figure::get_all_edges_count ( ) const  [inline]
```

Gets the total count of edges in the figure.

Returns

The total count of edges.

7.6.3.2 get_default_coords()

```
void s21::Figure::get_default_coords ( )
```

Restores the vertices to their default (original) coordinates.

7.6.3.3 get_edges()

```
std::vector<Edge> s21::Figure::get_edges ( ) const  [inline]
```

Gets the edges of the figure.

Returns

A vector of edges defining the figure.

7.6.3.4 get_faces()

```
std::vector<Face> s21::Figure::get_faces ( ) const  [inline]
```

Gets the faces of the figure.

Returns

A vector of faces defining the figure.

7.6.3.5 get_name()

```
std::string s21::Figure::get_name ( ) const  [inline]
```

Gets the name of the figure.

Returns

The name of the figure.

7.6.3.6 get_vertices()

```
std::vector<Vertex> s21::Figure::get_vertices ( ) const  [inline]
```

Gets the vertices of the figure.

Returns

A vector of vertices defining the figure.

7.6.3.7 make_edges()

```
void s21::Figure::make_edges ( )
```

Generates the edges of the figure based on its vertices and faces.

This method clears the existing edges and generates new edges by iterating through the faces and connecting vertices. It ensures that each edge is unique and updates the total count of edges.

7.6.3.8 save_default_coords()

```
void s21::Figure::save_default_coords ( )
```

Saves the current vertices as the default (original) coordinates.

7.6.3.9 set_name()

```
void s21::Figure::set_name (
            std::string name )  [inline]
```

Sets the name of the figure.

Parameters

name	The name to assign to the figure.

7.6.3.10 transform()

```
void s21::Figure::transform (
            const TransformMatrix & matrix )  [override], [virtual]
```

Transforms the figure's vertices using the provided transformation matrix. This method applies the transformation matrix to each vertex in the figure.

Parameters

matrix	The transformation matrix to apply.

Implements s21::SceneObject.

7.6.4 Friends And Related Function Documentation

7.6.4.1 Facade

```
friend class Facade  [friend]
```

7.6.4.2 FileReader

```
friend class FileReader  [friend]
```

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

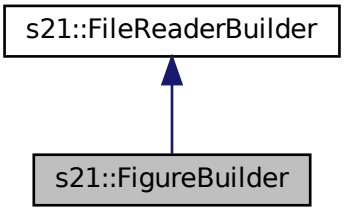
- model/figure/figure.h
- model/figure/figure.cpp

7.7 s21::FigureBuilder Class Reference

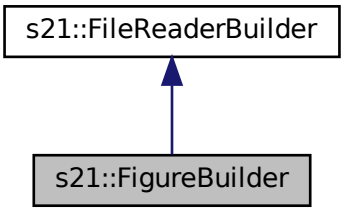
A concrete builder for creating Figure objects.

`#include <file_reader.h>`

Inheritance diagram for s21::FigureBuilder:



Collaboration diagram for s21::FigureBuilder:



Public Member Functions

- void reset () override

  Resets the FigureBuilder to its initial state.
- void set_normalization_parameters (NormalizationParameters *params) override

  Sets normalization parameters.
- Figure get_result (const std::string &filename)

  Constructs and returns a figure by reading data from a file.

7.7.1 Detailed Description

A concrete builder for creating Figure objects.

This class implements the FileReaderBuilder interface to construct Figure objects.

7.7.2 Member Function Documentation

7.7.2.1 get_result()

```
Figure s21::FigureBuilder::get_result (
            const std::string & filename )
```

Constructs and returns a figure by reading data from a file.

Parameters

filename	The name of the file to read.

Returns

The constructed figure.

7.7.2.2 reset()

```
void s21::FigureBuilder::reset ( )  [override], [virtual]
```

Resets the FigureBuilder to its initial state.

Implements s21::FileReaderBuilder.

7.7.2.3 set_normalization_parameters()

```
void s21::FigureBuilder::set_normalization_parameters (
            NormalizationParameters * params )  [inline], [override], [virtual]
```

Sets normalization parameters.

Parameters

params	Normalization parameters.

Implements s21::FileReaderBuilder.

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

- model/file_reader/file_reader.h
- model/file_reader/file_reader.cpp

7.8 s21::FileReader Class Reference

A class for reading figure data from a file.

`#include <file_reader.h>`

Public Member Functions

- FileReader ()=default
- ~FileReader ()=default
- Figure read_figure (std::string filename, NormalizationParameters *params)

  Reads a figure from a file and returns it.
- void set_file (std::string filename)

  Sets the file name.
- void get_data_figure (Figure &figure)

  Reads figure data from the file and updates the figure object.
- bool process_line (const std::string &current, Figure &figure)

  Processes a line from the file and updates the figure.
- bool open_file ()

  Opens the file for reading.
- void close_file ()

  Closes the file if it is open.
- bool read_successfull ()

  Checks if the file was read successfully.
- bool get_vertices (const std::string &current, Figure &figure)

  Extracts vertices from a file line and adds them to the figure.
- bool get_faces (const std::string &current, Figure &figure)

  Extracts faces from a file line and adds them to the figure.
- bool get_name (const std::string &current, Figure &figure)

  Extracts the figure name from a file line.
- void set_min_max_for_normalization (Vertex &vertex)

  Updates the min and max values for vertex normalization.
- void normalize_vertex (Vertex &vertex)

  Normalizes a vertex based on the normalization parameters.
- void normalize_figure (Figure &figure)

  Normalizes all vertices in the figure.

7.8.1 Detailed Description

A class for reading figure data from a file.

This class provides functionality for reading figure data from a file, processing the data, and normalizing vertex coordinates.

Note

In test mode (`#ifdef TEST`), the member variables are public for testing purposes.

7.8.2 Constructor & Destructor Documentation

7.8.2.1 FileReader()

```
s21::FileReader::FileReader ( )  [default]
```

7.8.2.2 ~FileReader()

```
s21::FileReader::~FileReader ( )  [default]
```

7.8.3 Member Function Documentation

7.8.3.1 close_file()

```
void s21::FileReader::close_file ( )
```

Closes the file if it is open.

7.8.3.2 get_data_figure()

```
void s21::FileReader::get_data_figure (
            Figure & figure )
```

Reads figure data from the file and updates the figure object.

Parameters

figure	Reference to the figure object.

7.8.3.3 get_faces()

```
bool s21::FileReader::get_faces (
            const std::string & current,
            Figure & figure )
```

Extracts faces from a file line and adds them to the figure.

Parameters

current	The current line from the file.
figure	Reference to the figure object.

Returns

True if faces were extracted successfully, otherwise false.

7.8.3.4 get_name()

```
bool s21::FileReader::get_name (
            const std::string & current,
            Figure & figure )
```

Extracts the figure name from a file line.

Parameters

current	The current line from the file.
figure	Reference to the figure object.

Returns

True if the name was extracted successfully, otherwise false.

7.8.3.5 get_vertices()

```
bool s21::FileReader::get_vertices (
            const std::string & current,
            Figure & figure )
```

Extracts vertices from a file line and adds them to the figure.

Parameters

current	The current line from the file.
figure	Reference to the figure object.

Returns

True if vertices were extracted successfully, otherwise false.

7.8.3.6 normalize_figure()

```
void s21::FileReader::normalize_figure (
            Figure & figure )
```

Normalizes all vertices in the figure.

Parameters

figure	Reference to the figure object.

7.8.3.7 normalize_vertex()

```
void s21::FileReader::normalize_vertex (
            Vertex & vertex )
```

Normalizes a vertex based on the normalization parameters.

Parameters

vertex	The vertex to normalize.

7.8.3.8 open_file()

```
bool s21::FileReader::open_file ( )
```

Opens the file for reading.

Returns

True if the file was opened successfully, otherwise false.

Exceptions

std::runtime_error	if the file cannot be opened.

7.8.3.9 process_line()

```
bool s21::FileReader::process_line (
            const std::string & current,
            Figure & figure )
```

Processes a line from the file and updates the figure.

Parameters

current	The current line from the file.
figure	Reference to the figure object.

Returns

True if the line was processed successfully, otherwise false.

7.8.3.10 read_figure()

```
Figure s21::FileReader::read_figure (
            std::string filename,
            NormalizationParameters * params )
```

Reads a figure from a file and returns it.

Parameters

filename	The name of the file to read.
params	Normalization parameters.

Returns

The constructed figure.

7.8.3.11 read_successfull()

```
bool s21::FileReader::read_successfull ( )
```

Checks if the file was read successfully.

Returns

True if the file was read successfully, otherwise false.

7.8.3.12 set_file()

```
void s21::FileReader::set_file (
            std::string filename )  [inline]
```

Sets the file name.

Parameters

filename	The name of the file.

7.8.3.13 set_min_max_for_normalization()

```
void s21::FileReader::set_min_max_for_normalization (
            Vertex & vertex )
```

Updates the min and max values for vertex normalization.

Parameters

vertex	The vertex to process.

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

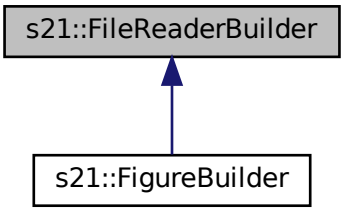
- model/file_reader/file_reader.h
- model/file_reader/file_reader.cpp

7.9 s21::FileReaderBuilder Class Reference

Abstract base class for building a FileReader. Use Builder pattern.

`#include <file_reader.h>`

Inheritance diagram for s21::FileReaderBuilder:



Public Member Functions

- virtual void reset ()=0

  Resets the builder.
- virtual void set_normalization_parameters (NormalizationParameters *params)=0

  Sets normalization parameters.
- virtual ~FileReaderBuilder ()=default

7.9.1 Detailed Description

Abstract base class for building a FileReader. Use Builder pattern.

This class defines the interface for building a FileReader object.

7.9.2 Constructor & Destructor Documentation

7.9.2.1 ~FileReaderBuilder()

```
virtual s21::FileReaderBuilder::~FileReaderBuilder ( )  [virtual], [default]
```

7.9.3 Member Function Documentation

7.9.3.1 reset()

```
virtual void s21::FileReaderBuilder::reset ( )  [pure virtual]
```

Resets the builder.

Implemented in s21::FigureBuilder.

7.9.3.2 set_normalization_parameters()

```
virtual void s21::FileReaderBuilder::set_normalization_parameters (
            NormalizationParameters * params )  [pure virtual]
```

Sets normalization parameters.

Parameters

params	Normalization parameters.

Implemented in s21::FigureBuilder.

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

- model/file_reader/file_reader.h

7.10 s21::FileReaderDirector Class Reference

Constructs a figure using the provided builder.

`#include <file_reader.h>`

Public Member Functions

- Figure construct_figure (FigureBuilder &builder, const std::string &filename, NormalizationParameters *params)

  Constructs a figure using the provided builder.

7.10.1 Detailed Description

Constructs a figure using the provided builder.

Parameters

builder	The builder to use.
filename	The name of the file to read.
params	Normalization parameters.

Returns

The constructed figure.

7.10.2 Member Function Documentation

7.10.2.1 construct_figure()

```
Figure s21::FileReaderDirector::construct_figure (
            FigureBuilder & builder,
            const std::string & filename,
            NormalizationParameters * params )
```

Constructs a figure using the provided builder.

Parameters

builder	The builder to use.
filename	The name of the file to read.
params	Normalization parameters.

Returns

The constructed figure.

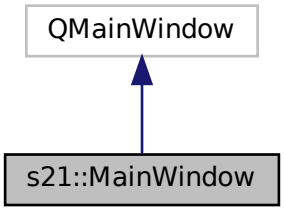
The documentation for this class was generated from the following files:

- model/file_reader/file_reader.h
- model/file_reader/file_reader.cpp

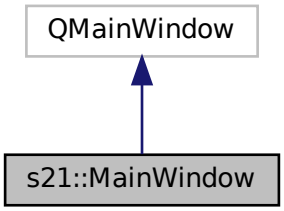
7.11 s21::MainWindow Class Reference

`#include <mainwindow.h>`

Inheritance diagram for s21::MainWindow:



Collaboration diagram for s21::MainWindow:



Public Member Functions

- MainWindow (QWidget *parent=nullptr)
- ~MainWindow ()

7.11.1 Constructor & Destructor Documentation

7.11.1.1 MainWindow()

```
s21::MainWindow::MainWindow (
            QWidget * parent = nullptr )
```

7.11.1.2 ~MainWindow()

```
s21::MainWindow::~MainWindow ( )
```

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

- gui/mainwindow.h
- gui/mainwindow.cpp

7.12 s21::NormalizationParameters Class Reference

Manages the normalization parameters for Figure objects.

```
#include <normalization_parameters.h>
```

Public Member Functions

- ~NormalizationParameters ()=default
- void reset_normalization_parameters ()

Static Public Member Functions

- static NormalizationParameters * instance ()

  Implementation of the singleton instance of the NormalizationParameters class.

Public Attributes

- float x_min
- float x_max
- float dx_step
- float y_min
- float y_max
- float dy_step
- float z_min
- float z_max
- float dz_step
- float max_range

Protected Member Functions

- NormalizationParameters ()
- NormalizationParameters (NormalizationParameters const &)=delete
- NormalizationParameters & operator= (NormalizationParameters const &)=delete

7.12.1 Detailed Description

Manages the normalization parameters for Figure objects.

This class provides a singleton instance that holds the minimum and maximum values for the x, y, and z axes, as well as the step sizes for each axis.

7.12.2 Constructor & Destructor Documentation

7.12.2.1 ~NormalizationParameters()

```
s21::NormalizationParameters::~NormalizationParameters ( )  [default]
```

7.12.2.2 NormalizationParameters() [1/2]

```
s21::NormalizationParameters::NormalizationParameters ( )  [inline], [protected]
```

7.12.2.3 NormalizationParameters() [2/2]

```
s21::NormalizationParameters::NormalizationParameters (
            NormalizationParameters const &  )  [protected], [delete]
```

7.12.3 Member Function Documentation

7.12.3.1 instance()

```
static NormalizationParameters* s21::NormalizationParameters::instance ( )  [inline], [static]
```

Implementation of the singleton instance of the NormalizationParameters class.

Returns

A pointer to the only one NormalizationParameters object.

7.12.3.2 operator=()

```
NormalizationParameters& s21::NormalizationParameters::operator= (
            NormalizationParameters const &  )  [protected], [delete]
```

7.12.3.3 reset_normalization_parameters()

```
void s21::NormalizationParameters::reset_normalization_parameters ( )  [inline]
```

7.12.4 Member Data Documentation

7.12.4.1 dx_step

```
float s21::NormalizationParameters::dx_step
```

7.12.4.2 dy_step

```
float s21::NormalizationParameters::dy_step
```

7.12.4.3 dz_step

```
float s21::NormalizationParameters::dz_step
```

7.12.4.4 max_range

```
float s21::NormalizationParameters::max_range
```

7.12.4.5 x_max

```
float s21::NormalizationParameters::x_max
```

7.12.4.6 x_min

```
float s21::NormalizationParameters::x_min
```

7.12.4.7 y_max

```
float s21::NormalizationParameters::y_max
```

7.12.4.8 y_min

```
float s21::NormalizationParameters::y_min
```

7.12.4.9 z_max

```
float s21::NormalizationParameters::z_max
```

7.12.4.10 z_min

```
float s21::NormalizationParameters::z_min
```

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

- model/facade/normalization_parameters.h

7.13 s21::PairHash Struct Reference

Hash function for pairs of values. This struct provides a hash function for pairs of values, used to store edges in a hash set.

Public Member Functions

- template<typename T1 , typename T2 >
  std::size_t operator() (const std::pair< T1, T2 > &p) const

7.13.1 Detailed Description

Hash function for pairs of values. This struct provides a hash function for pairs of values, used to store edges in a hash set.

7.13.2 Member Function Documentation

7.13.2.1 operator()()

```
template<typename T1 , typename T2 >
std::size_t s21::PairHash::operator() (
            const std::pair< T1, T2 > & p ) const  [inline]
```

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

- model/figure/figure.cpp

7.14 s21::Point Struct Reference

Represents a point in 3D space with homogeneous coordinates.

```
#include <vertex.h>
```

Public Member Functions

- Point ()

  Default constructor for the Point struct. Initializes the point to (0, 0, 0, 1).
- Point (float x, float y, float z, float w=1.0)

  Parameterized constructor for the Point struct. Initializes the point with the provided coordinates.
- Point operator∗ (const TransformMatrix &matrix) const

  Multiplies the point by a transformation matrix. This method applies the transformation matrix to the point, resulting in a new point with transformed coordinates.
- bool operator== (const Point &other) const

  Compares two points for equality. Two points are considered equal if all their coordinates (x, y, z, w) are the same.
- void print_point () const

  Prints the point's coordinates to the standard output.
- float get_x () const

  Gets the x-coordinate of the point.
- float get_y () const

  Gets the y-coordinate of the point.
- float get_z () const

  Gets the z-coordinate of the point.
- float get_w () const

  Gets the homogeneous coordinate (w) of the point.
- void set_x (float x)

  Sets the x-coordinate of the point.
- void set_y (float y)

  Sets the y-coordinate of the point.
- void set_z (float z)

  Sets the z-coordinate of the point.

Public Attributes

- float x_

  The x-coordinate of the point.
- float y_

  The y-coordinate of the point.
- float z_

  The z-coordinate of the point.
- float w_

  The homogeneous coordinate (default is 1.0).

7.14.1 Detailed Description

Represents a point in 3D space with homogeneous coordinates.

The `Point` struct defines a point with coordinates (x, y, z, w), where w is the homogeneous coordinate (default is 1.0). It provides methods for transformation, comparison, and printing, as well as getters and setters for the coordinates.

7.14.2 Constructor & Destructor Documentation

7.14.2.1 Point() [1/2]

```
s21::Point::Point ( )  [inline]
```

Default constructor for the Point struct. Initializes the point to (0, 0, 0, 1).

7.14.2.2 Point() [2/2]

```
s21::Point::Point (
            float x,
            float y,
            float z,
            float w = 1.0 )  [inline]
```

Parameterized constructor for the Point struct. Initializes the point with the provided coordinates.

Parameters

x	The x-coordinate.
y	The y-coordinate.
z	The z-coordinate.
w	The homogeneous coordinate (default is 1.0).

7.14.3 Member Function Documentation

7.14.3.1 get_w()

```
float s21::Point::get_w ( ) const  [inline]
```

Gets the homogeneous coordinate (w) of the point.

Returns

The homogeneous coordinate.

7.14.3.2 get_x()

```
float s21::Point::get_x ( ) const  [inline]
```

Gets the x-coordinate of the point.

Returns

The x-coordinate.

7.14.3.3 get_y()

```
float s21::Point::get_y ( ) const  [inline]
```

Gets the y-coordinate of the point.

Returns

The y-coordinate.

7.14.3.4 get_z()

```
float s21::Point::get_z ( ) const  [inline]
```

Gets the z-coordinate of the point.

Returns

The z-coordinate.

7.14.3.5 operator*()

```
Point s21::Point::operator* (
            const TransformMatrix & matrix ) const
```

Multiplies the point by a transformation matrix. This method applies the transformation matrix to the point, resulting in a new point with transformed coordinates.

Parameters

matrix	The transformation matrix to apply.

Returns

A new Point representing the transformed point.

7.14.3.6 operator==()

```
bool s21::Point::operator== (
            const Point & other ) const
```

Compares two points for equality. Two points are considered equal if all their coordinates (x, y, z, w) are the same.

Parameters

other	The point to compare with.

Returns

true if the points are equal, false otherwise.

7.14.3.7 print_point()

```
void s21::Point::print_point ( ) const
```

Prints the point's coordinates to the standard output.

7.14.3.8 set_x()

```
void s21::Point::set_x (
            float x )  [inline]
```

Sets the x-coordinate of the point.

Parameters

x	The new x-coordinate.

7.14.3.9 set_y()

```
void s21::Point::set_y (
            float y )  [inline]
```

Sets the y-coordinate of the point.

Parameters

y	The new y-coordinate.

7.14.3.10 set_z()

```
void s21::Point::set_z (
            float z )  [inline]
```

Sets the z-coordinate of the point.

Parameters

z	The new z-coordinate.

7.14.4 Member Data Documentation

7.14.4.1 w_

```
float s21::Point::w_
```

The homogeneous coordinate (default is 1.0).

7.14.4.2 x_

```
float s21::Point::x_
```

The x-coordinate of the point.

7.14.4.3 y_

```
float s21::Point::y_
```

The y-coordinate of the point.

7.14.4.4 z_

```
float s21::Point::z_
```

The z-coordinate of the point.

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

- model/figure/vertex.h
- model/figure/vertex.cpp

7.15 s21::View::PrevPositions Struct Reference

Keeps track of the previous positions of the scale, move, and rotate 3D object.

```
#include <view.h>
```

Public Attributes

- float prev_scale = START_SCALE / 100.0
- float prev_x_pos = START_POS
- float prev_y_pos = START_POS
- float prev_z_pos = START_POS
- float prev_x_angle = START_ANGLE
- float prev_y_angle = START_ANGLE
- float prev_z_angle = START_ANGLE

7.15.1 Detailed Description

Keeps track of the previous positions of the scale, move, and rotate 3D object.

7.15.2 Member Data Documentation

7.15.2.1 prev_scale

```
float s21::View::PrevPositions::prev_scale = START_SCALE / 100.0
```

7.15.2.2 prev_x_angle

```
float s21::View::PrevPositions::prev_x_angle = START_ANGLE
```

7.15.2.3 prev_x_pos

```
float s21::View::PrevPositions::prev_x_pos = START_POS
```

7.15.2.4 prev_y_angle

```
float s21::View::PrevPositions::prev_y_angle = START_ANGLE
```

7.15.2.5 prev_y_pos

```
float s21::View::PrevPositions::prev_y_pos = START_POS
```

7.15.2.6 prev_z_angle

```
float s21::View::PrevPositions::prev_z_angle = START_ANGLE
```

7.15.2.7 prev_z_pos

```
float s21::View::PrevPositions::prev_z_pos = START_POS
```

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

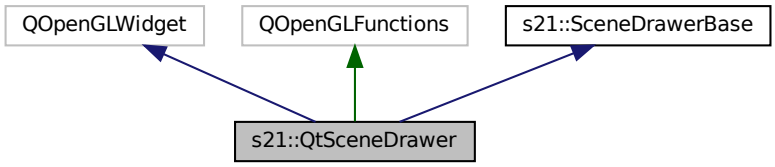
- gui/view/view.h

7.16 s21::QtSceneDrawer Class Reference

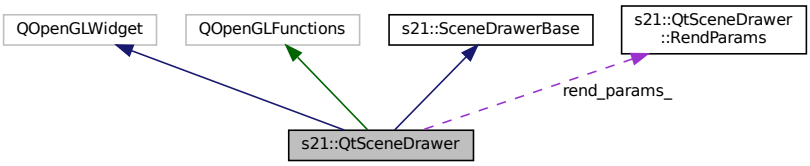
The QtSceneDrawer is an OpenGL-based widget for rendering 3D figures.

`#include <qt_scene_drawer.h>`

Inheritance diagram for s21::QtSceneDrawer:



Collaboration diagram for s21::QtSceneDrawer:



Classes

- struct RendParams

  Struct contains rendering parameters for the 3D scene (projection type, background color, edge and vertex rendering styles and colors).

Public Member Functions

- QtSceneDrawer (QWidget *parent=Q_NULLPTR)

  *Constructor for QtSceneDrawer object. *.*
- ~QtSceneDrawer ()

  Destructor for the QtSceneDrawer object.
- void initializeGL () override

  Initializes the OpenGL functions. This function called automatically when the OpenGL context is created.
- void resizeGL (int w, int h) override

  Resizes the OpenGL area to the specified width and height. Called automatically when widget is resized.
- void paintGL () override

  Renders the 3D model using the current rendering parameters. This function is called automatically when widget is updated.

- void DrawFigure (Figure *figure) override

  Fills array of points for drawing 3D model from Figure object.
- RendParams * get_rend_params ()

  Returns a pointer to the RendParams object used by the QtSceneDrawer.
- void load_view_settings ()

  Loads the current model view settings. If there are no saved settings, the default settings are used.
- void save_view_settings ()

  Saves the current model view settings.

Protected Attributes

- float * edges_array_
- size_t array_size_
- RendParams * rend_params_
- QSettings view_settings_

7.16.1 Detailed Description

The QtSceneDrawer is an OpenGL-based widget for rendering 3D figures.

7.16.2 Constructor & Destructor Documentation

7.16.2.1 QtSceneDrawer()

```
s21::QtSceneDrawer::QtSceneDrawer (
            QWidget * parent = Q_NULLPTR )  [explicit]
```

Constructor for QtSceneDrawer object. *.

Parameters

parent	The parent QWidget for this QtSceneDrawer.

7.16.2.2 ~QtSceneDrawer()

```
s21::QtSceneDrawer::~QtSceneDrawer ( )
```

Destructor for the QtSceneDrawer object.

7.16.3 Member Function Documentation

7.16.3.1 DrawFigure()

```
void s21::QtSceneDrawer::DrawFigure (
            Figure * figure )  [override], [virtual]
```

Fills array of points for drawing 3D model from Figure object.

If the figure has edges, the function generates an array of 6 floats per edge. If the figure has no edges, the function generates an array of 3 floats per vertex.

Parameters

figure	The Figure object to be drawn.

Implements s21::SceneDrawerBase.

7.16.3.2 get_rend_params()

```
QtSceneDrawer::RendParams * s21::QtSceneDrawer::get_rend_params ( )
```

Returns a pointer to the RendParams object used by the QtSceneDrawer.

Simple getter function for delivering render parameters to View and SettingsWidget objects.

Returns

QtSceneDrawer::RendParams* A pointer to the RendParams object.

7.16.3.3 initializeGL()

```
void s21::QtSceneDrawer::initializeGL ( )  [override]
```

Initializes the OpenGL functions. This function called automatically when the OpenGL context is created.

7.16.3.4 load_view_settings()

```
void s21::QtSceneDrawer::load_view_settings ( )
```

Loads the current model view settings. If there are no saved settings, the default settings are used.

7.16.3.5 paintGL()

```
void s21::QtSceneDrawer::paintGL ( )  [override]
```

Renders the 3D model using the current rendering parameters. This function is called automatically when widget is updated.

7.16.3.6 resizeGL()

```
void s21::QtSceneDrawer::resizeGL (
            int w,
            int h )  [override]
```

Resizes the OpenGL area to the specified width and height. Called automatically when widget is resized.

Parameters

w	The new width of widget.
h	The new height of widget.

7.16.3.7 save_view_settings()

```
void s21::QtSceneDrawer::save_view_settings ( )
```

Saves the current model view settings.

By default the settings are saved in /home/user/.config/company_name/project_name.conf in Ubuntu

7.16.4 Member Data Documentation

7.16.4.1 array_size_

```
size_t s21::QtSceneDrawer::array_size_  [protected]
```

7.16.4.2 edges_array_

```
float* s21::QtSceneDrawer::edges_array_  [protected]
```

7.16.4.3 rend_params_

```
RendParams* s21::QtSceneDrawer::rend_params_  [protected]
```

7.16.4.4 view_settings_

```
QSettings s21::QtSceneDrawer::view_settings_  [protected]
```

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

- gui/view/qt_scene_drawer.h
- gui/view/qt_scene_drawer.cpp

7.17 s21::QtSceneDrawer::RendParams Struct Reference

Struct contains rendering parameters for the 3D scene (projection type, background color, edge and vertex rendering styles and colors).

`#include <qt_scene_drawer.h>`

Public Types

- enum Projection { PARALLEL = 0 , CENTRAL }
  Projection type.
- enum EdgesType { NO_EDGES = 0 , SOLID , DASHED }
  Edge rendering style.
- enum VerticesType { NO_VERTICES = 0 , CIRCLE , SQUARE }
  Vertex rendering style.

Public Attributes

- int projection = PARALLEL
- QColor bg_color = {184, 213, 238}
- EdgesType edge_type = SOLID
- int edge_thickness = 1
- QColor edge_color = {0, 0, 255}
- VerticesType vertex_type = NO_VERTICES
- int vertex_size = 3
- QColor vertex_color = {0, 0, 0}

7.17.1 Detailed Description

Struct contains rendering parameters for the 3D scene (projection type, background color, edge and vertex rendering styles and colors).

7.17.2 Member Enumeration Documentation

7.17.2.1 EdgesType

```
enum s21::QtSceneDrawer::RendParams::EdgesType
```

Edge rendering style.

Enumerator

NO_EDGES	
SOLID	
DASHED	

7.17.2.2 Projection

`enum s21::QtSceneDrawer::RendParams::Projection`

Projection type.

Enumerator

PARALLEL	
CENTRAL	

7.17.2.3 VerticesType

`enum s21::QtSceneDrawer::RendParams::VerticesType`

Vertex rendering style.

Enumerator

NO_VERTICES	
CIRCLE	
SQUARE	

7.17.3 Member Data Documentation

7.17.3.1 bg_color

```
QColor s21::QtSceneDrawer::RendParams::bg_color = {184, 213, 238}
```

7.17.3.2 edge_color

```
QColor s21::QtSceneDrawer::RendParams::edge_color = {0, 0, 255}
```

7.17.3.3 edge_thickness

```
int s21::QtSceneDrawer::RendParams::edge_thickness = 1
```

7.17.3.4 edge_type

```
EdgesType s21::QtSceneDrawer::RendParams::edge_type = SOLID
```

7.17.3.5 projection

```
int s21::QtSceneDrawer::RendParams::projection = PARALLEL
```

7.17.3.6 vertex_color

```
QColor s21::QtSceneDrawer::RendParams::vertex_color = {0, 0, 0}
```

7.17.3.7 vertex_size

```
int s21::QtSceneDrawer::RendParams::vertex_size = 3
```

7.17.3.8 vertex_type

```
VerticesType s21::QtSceneDrawer::RendParams::vertex_type = NO_VERTICES
```

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

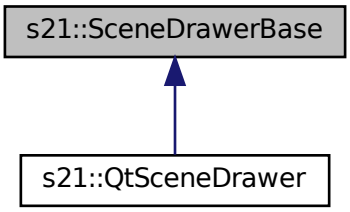
- gui/view/qt_scene_drawer.h

7.18 s21::SceneDrawerBase Class Reference

Abstract base class for scene drawers.

`#include <scene_drawer_base.h>`

Inheritance diagram for s21::SceneDrawerBase:



Public Member Functions

- SceneDrawerBase ()=default
- virtual ~SceneDrawerBase ()
- virtual void DrawFigure (Figure *figure)=0

7.18.1 Detailed Description

Abstract base class for scene drawers.

Concrete subclass of this class must implement the DrawFigure method to handle the actual rendering of a figure.

7.18.2 Constructor & Destructor Documentation

7.18.2.1 SceneDrawerBase()

```
s21::SceneDrawerBase::SceneDrawerBase ( )  [default]
```

7.18.2.2 ~SceneDrawerBase()

```
virtual s21::SceneDrawerBase::~SceneDrawerBase ( )  [inline], [virtual]
```

7.18.3 Member Function Documentation

7.18.3.1 DrawFigure()

```
virtual void s21::SceneDrawerBase::DrawFigure (
            Figure * figure )  [pure virtual]
```

Implemented in s21::QtSceneDrawer.

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

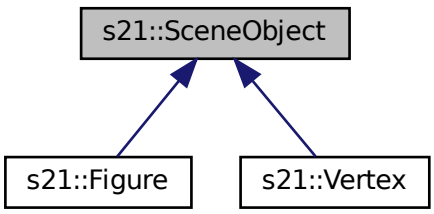
- model/facade/scene_drawer_base.h

7.19 s21::SceneObject Class Reference

Abstract base class representing a scene object.

```
#include <scene_object.h>
```

Inheritance diagram for s21::SceneObject:



Public Member Functions

- virtual ~SceneObject ()=default

  Virtual destructor for the SceneObject class. Ensures proper cleanup of derived class objects when deleted through a base class pointer.
- virtual void transform (const TransformMatrix &matrix)=0

  Pure virtual method to transform the scene object. This method must be implemented by derived classes to apply a transformation matrix to the object.

7.19.1 Detailed Description

Abstract base class representing a scene object.

The `SceneObject` class serves as a base class for all objects in a 3D scene. It defines an interface for transforming objects using a transformation matrix. Classes derived from `SceneObject` must implement the `transform` method.

7.19.2 Constructor & Destructor Documentation

7.19.2.1 ~SceneObject()

```
virtual s21::SceneObject::~SceneObject ( )  [virtual], [default]
```

Virtual destructor for the SceneObject class. Ensures proper cleanup of derived class objects when deleted through a base class pointer.

7.19.3 Member Function Documentation

7.19.3.1 transform()

```
virtual void s21::SceneObject::transform (
            const TransformMatrix & matrix )  [pure virtual]
```

Pure virtual method to transform the scene object. This method must be implemented by derived classes to apply a transformation matrix to the object.

Parameters

matrix	The transformation matrix to apply.

Implemented in s21::Vertex, and s21::Figure.

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

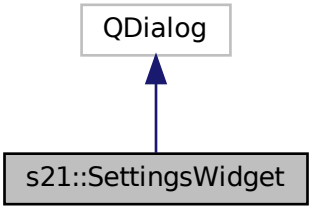
- model/figure/scene_object.h

7.20 s21::SettingsWidget Class Reference

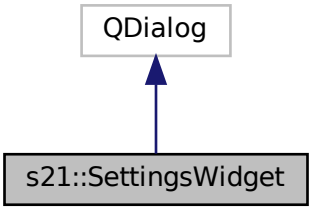
A dialog widget that allows user to customize type, color and thickness of the edges, display method, color and size of the vertices of a 3D model.

`#include <settings_widget.h>`

Inheritance diagram for s21::SettingsWidget:



Collaboration diagram for s21::SettingsWidget:



Public Member Functions

- SettingsWidget (QWidget *parent=nullptr, QtSceneDrawer::RendParams *params=nullptr)

  Constructor for SettingsWidget object.
- ~SettingsWidget ()

  Destructor for SettingsWidget object. Deletes temporary rendering parameters struct.

7.20.1 Detailed Description

A dialog widget that allows user to customize type, color and thickness of the edges, display method, color and size of the vertices of a 3D model.

7.20.2 Constructor & Destructor Documentation

7.20.2.1 SettingsWidget()

```
s21::SettingsWidget::SettingsWidget (
            QWidget * parent = nullptr,
            QtSceneDrawer::RendParams * params = nullptr )  [explicit]
```

Constructor for SettingsWidget object.

Parameters

parent	The parent widget for the SettingsWidget.
params	A pointer to the rendering parameters struct in QtSceneDrawer

7.20.2.2 ~SettingsWidget()

```
s21::SettingsWidget::~SettingsWidget ( )
```

Destructor for SettingsWidget object. Deletes temporary rendering parameters struct.

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

- gui/view/settings_widget.h
- gui/view/settings_widget.cpp

7.21 s21::TransformMatrix Class Reference

A class representing a 4x4 transformation matrix.

`#include <transform_matrix.h>`

Public Member Functions

- TransformMatrix ()

  Default constructor for TransformMatrix.
- ~TransformMatrix ()=default

  Default destructor.
- void reset ()

  Resets the matrix to a zero matrix. All elements of the matrix are set to 0.
- TransformMatrix operator* (const TransformMatrix &other_matrix) const

  Overloads the multiplication operator for matrix multiplication.
- Point operator* (const Point &point) const

  Overloads the multiplication operator for transforming a point.
- Point transform_point (Point &point)

  Transforms a point using the transformation matrix.
- float get_elem (int row, int col) const

  Gets the value of a matrix element at the specified row and column.
- void set_elem (int row, int col, float v)

  Sets the value of a matrix element at the specified row and column.

7.21.1 Detailed Description

A class representing a 4x4 transformation matrix.

This class provides functionality to manage and manipulate a 4x4 transformation matrix.

Note

In test mode (`#ifdef TEST`), the member variables are public for testing purposes.

7.21.2 Constructor & Destructor Documentation

7.21.2.1 TransformMatrix()

```
s21::TransformMatrix::TransformMatrix ( )
```

Default constructor for TransformMatrix.

7.21.2.2 ~TransformMatrix()

```
s21::TransformMatrix::~TransformMatrix ( )  [default]
```

Default destructor.

7.21.3 Member Function Documentation

7.21.3.1 get_elem()

```
float s21::TransformMatrix::get_elem (
            int row,
            int col ) const  [inline]
```

Gets the value of a matrix element at the specified row and column.

Parameters

row	The row index (starts from 0).
col	The column index (starts from 0).

Returns

The value of the matrix element at the specified position.

7.21.3.2 operator*() [1/2]

```
Point s21::TransformMatrix::operator* (
            const Point & point ) const
```

Overloads the multiplication operator for transforming a point.

Applies the transformation matrix to a point and returns the transformed point.

Parameters

point	The point to transform.

Returns

A new Point representing the transformed point.

7.21.3.3 operator*() [2/2]

```
TransformMatrix s21::TransformMatrix::operator* (
            const TransformMatrix & other_matrix ) const
```

Overloads the multiplication operator for matrix multiplication.

Parameters

other_matrix	The matrix to multiply with.

Returns

A new TransformMatrix representing the result of the multiplication.

Multiplies this matrix with another matrix and returns the result.

Parameters

other_matrix	The matrix to multiply with.

Returns

A new TransformMatrix representing the result of the multiplication.

7.21.3.4 reset()

```
void s21::TransformMatrix::reset ( )
```

Resets the matrix to a zero matrix. All elements of the matrix are set to 0.

7.21.3.5 set_elem()

```
void s21::TransformMatrix::set_elem (
            int row,
            int col,
            float v )
```

Sets the value of a matrix element at the specified row and column.

Parameters

row	The row index (starts from 0).
col	The column index (starts from 0).
v	The value to set.

Exceptions

std::out_of_range	If the row or column index is out of bounds.

7.21.3.6 transform_point()

```
Point s21::TransformMatrix::transform_point (
            Point & point )
```

Transforms a point using the transformation matrix.

Parameters

point	The point to transform.

Returns

A new Point representing the transformed point.

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

- model/transform_matrix/transform_matrix.h
- model/transform_matrix/transform_matrix.cpp

7.22 s21::TransformMatrixBuilder Class Reference

A utility class for creating transformation matrices.

`#include <transform_matrix.h>`

Static Public Member Functions

- static TransformMatrix create_only_x_rotation_matrix (float angle)

  Creates a rotation matrix for rotation around the X-axis.
- static TransformMatrix create_only_y_rotation_matrix (float angle)

  Creates a rotation matrix for rotation around the Y-axis.
- static TransformMatrix create_only_z_rotation_matrix (float angle)

  Creates a rotation matrix for rotation around the Z-axis.
- static TransformMatrix create_rotation_matrix (float x_degree, float y_degree, float z_degree)

  Creates a combined rotation matrix for rotations around all three axes.
- static TransformMatrix create_move_matrix (float x, float y, float z)

  Creates a moving matrix for moving points in 3D space.
- static TransformMatrix create_scale_matrix (float x, float y, float z)

  Creates a scaling matrix for scaling points in 3D space.

7.22.1 Detailed Description

A utility class for creating transformation matrices.

This class provides static methods to generate specific types of transformation matrices. This class works like "builder" pattern.

7.22.2 Member Function Documentation

7.22.2.1 create_move_matrix()

```
TransformMatrix s21::TransformMatrixBuilder::create_move_matrix (
            float x,
            float y,
            float z )  [static]
```

Creates a moving matrix for moving points in 3D space.

Parameters

x	The distance along the X-axis.
y	The distance along the Y-axis.
z	The distance along the Z-axis.

Returns

A TransformMatrix representing the moving.

7.22.2.2 create_only_x_rotation_matrix()

```
TransformMatrix s21::TransformMatrixBuilder::create_only_x_rotation_matrix (
            float angle )  [static]
```

Creates a rotation matrix for rotation around the X-axis.

Parameters

angle	The rotation angle in radians.

Returns

A TransformMatrix representing the X-axis rotation.

7.22.2.3 create_only_y_rotation_matrix()

```
TransformMatrix s21::TransformMatrixBuilder::create_only_y_rotation_matrix (
            float angle )  [static]
```

Creates a rotation matrix for rotation around the Y-axis.

Parameters

angle	The rotation angle in radians.

Returns

A TransformMatrix representing the Y-axis rotation.

7.22.2.4 create_only_z_rotation_matrix()

```
TransformMatrix s21::TransformMatrixBuilder::create_only_z_rotation_matrix (
            float angle )  [static]
```

Creates a rotation matrix for rotation around the Z-axis.

Parameters

angle	The rotation angle in radians.

Returns

A TransformMatrix representing the Z-axis rotation.

7.22.2.5 create_rotation_matrix()

```
TransformMatrix s21::TransformMatrixBuilder::create_rotation_matrix (
            float x_degree,
            float y_degree,
            float z_degree )  [static]
```

Creates a combined rotation matrix for rotations around all three axes.

Look at the order - it must be: x -> y -> z!

Parameters

x_degree	The rotation angle around the X-axis in degrees.
y_degree	The rotation angle around the Y-axis in degrees.
z_degree	The rotation angle around the Z-axis in degrees.

Returns

A TransformMatrix representing the combined rotation.

7.22.2.6 create_scale_matrix()

```
TransformMatrix s21::TransformMatrixBuilder::create_scale_matrix (
            float x,
            float y,
            float z )  [static]
```

Creates a scaling matrix for scaling points in 3D space.

Parameters

x	The scaling factor along the X-axis.
y	The scaling factor along the Y-axis.
z	The scaling factor along the Z-axis.

Returns

A TransformMatrix representing the scaling.

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

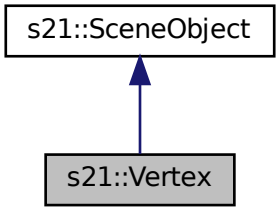
- model/transform_matrix/transform_matrix.h
- model/transform_matrix/transform_matrix.cpp

7.23 s21::Vertex Class Reference

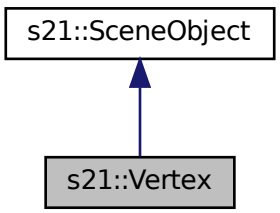
Represents a vertex in 3D space, derived from SceneObject.

`#include <vertex.h>`

Inheritance diagram for s21::Vertex:



Collaboration diagram for s21::Vertex:



Public Member Functions

- Vertex ()

  Default constructor for the Vertex class. Initializes the vertex with a default point (0, 0, 0, 1).
- Vertex (Point p)

  Parameterized constructor for the Vertex class. Initializes the vertex with the provided point.
- ~Vertex ()=default

  Destructor for the Vertex class.
- Point get_position () const

  Gets the position of the vertex.
- void set_position (Point point)

  Sets the position of the vertex.
- void transform (const TransformMatrix &matrix) override

  Transforms the vertex by multiplying its position with the transformation matrix. This method applies the transformation matrix to the vertex's position, updating its coordinates.
- bool operator== (const Vertex &other) const

  Compares two vertices for equality. Two vertices are considered equal if their positions are the same.

7.23.1 Detailed Description

Represents a vertex in 3D space, derived from SceneObject.

The `Vertex` class encapsulates a point in 3D space and provides functionality for transforming the vertex and comparing it with other vertices.

Note

In test mode (`#ifdef TEST`), the member variables are public for testing purposes.

7.23.2 Constructor & Destructor Documentation

7.23.2.1 Vertex() [1/2]

```
s21::Vertex::Vertex ( )  [inline]
```

Default constructor for the Vertex class. Initializes the vertex with a default point (0, 0, 0, 1).

7.23.2.2 Vertex() [2/2]

```
s21::Vertex::Vertex (
            Point p )  [inline], [explicit]
```

Parameterized constructor for the Vertex class. Initializes the vertex with the provided point.

Parameters

p	The point representing the vertex's position.

7.23.2.3 ~Vertex()

```
s21::Vertex::~Vertex ( )  [default]
```

Destructor for the Vertex class.

7.23.3 Member Function Documentation

7.23.3.1 get_position()

```
Point s21::Vertex::get_position ( ) const  [inline]
```

Gets the position of the vertex.

Returns

The point representing the vertex's position.

7.23.3.2 operator==()

```
bool s21::Vertex::operator== (
            const Vertex & other ) const
```

Compares two vertices for equality. Two vertices are considered equal if their positions are the same.

Parameters

other	The vertex to compare with.

Returns

true if the vertices are equal, false otherwise.

7.23.3.3 set_position()

```
void s21::Vertex::set_position (
            Point point )  [inline]
```

Sets the position of the vertex.

Parameters

point	The new position of the vertex.

7.23.3.4 transform()

```
void s21::Vertex::transform (
            const TransformMatrix & matrix )  [override], [virtual]
```

Transforms the vertex by multiplying its position with the transformation matrix. This method applies the transformation matrix to the vertex's position, updating its coordinates.

Parameters

matrix	The transformation matrix to apply.

Implements s21::SceneObject.

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

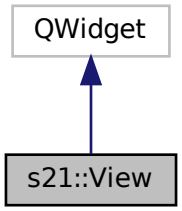
- model/figure/vertex.h
- model/figure/vertex.cpp

7.24 s21::View Class Reference

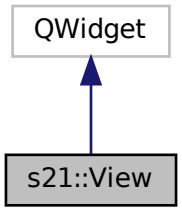
The View class is responsible for the GUI of the 3D viewer application and handling user interactions, such as opening and saving files, adjusting the 3D object's scale, position, and rotation, switching between parallel and central projection modes. It also includes a QtSceneDrawer object to render the 3D scene and a SettingsWidget object to manage the application's settings.

```
#include <view.h>
```

Inheritance diagram for s21::View:



Collaboration diagram for s21::View:



Classes

- struct PrevPositions

  Keeps track of the previous positions of the scale, move, and rotate 3D object.

Public Types

- enum ControlRanges {
  SCALE_MIN = 1 , SCALE_MAX = 500 , MOVE_MIN = -200 , MOVE_MAX = 200 ,
  ROTATE_MIN = -180 , ROTATE_MAX = 180 }

  Defines the minimum and maximum values for the scale, move, and rotate controls used in the View class.

Public Member Functions

- View (QWidget *parent=nullptr, Controller *controller=nullptr)

  Constructor for View object. The View class contains all user interface elements and visualisation area for 3D scene.
- ~View ()

  Destructor for the View class.

7.24.1 Detailed Description

The View class is responsible for the GUI of the 3D viewer application and handling user interactions, such as opening and saving files, adjusting the 3D object's scale, position, and rotation, switching between parallel and central projection modes. It also includes a QtSceneDrawer object to render the 3D scene and a SettingsWidget object to manage the application's settings.

7.24.2 Member Enumeration Documentation

7.24.2.1 ControlRanges

```
enum s21::View::ControlRanges
```

Defines the minimum and maximum values for the scale, move, and rotate controls used in the View class.

Enumerator

SCALE_MIN	
SCALE_MAX	
MOVE_MIN	
MOVE_MAX	
ROTATE_MIN	
ROTATE_MAX	

7.24.3 Constructor & Destructor Documentation

7.24.3.1 View()

```
s21::View::View (
            QWidget * parent = nullptr,
            Controller * controller = nullptr )  [explicit]
```

Constructor for View object. The View class contains all user interface elements and visualisation area for 3D scene.

7.24.3.2 ~View()

```
s21::View::~View ( )
```

Destructor for the View class.

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

- gui/view/view.h
- gui/view/view.cpp

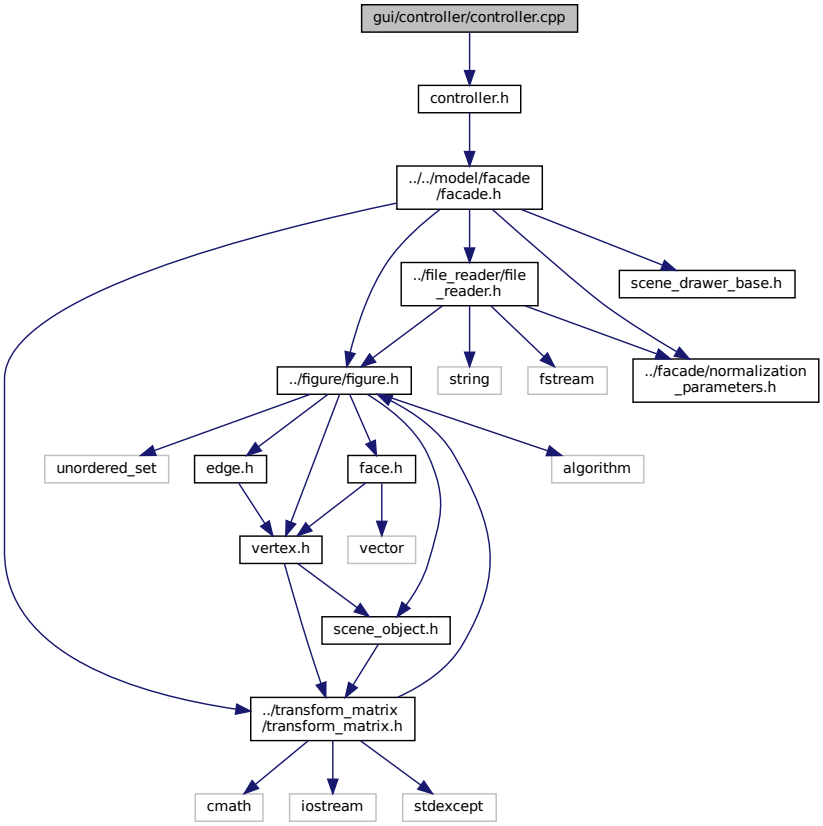
Chapter 8

File Documentation

8.1 gui/controller/controller.cpp File Reference

```
#include "controller.h"
```

Include dependency graph for controller.cpp:



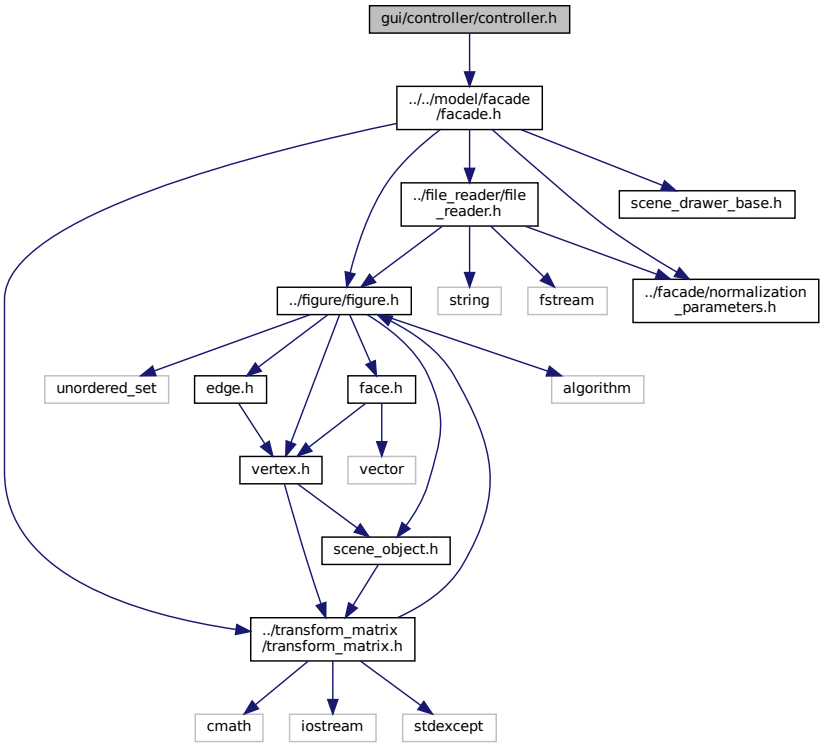
Namespaces

- s21

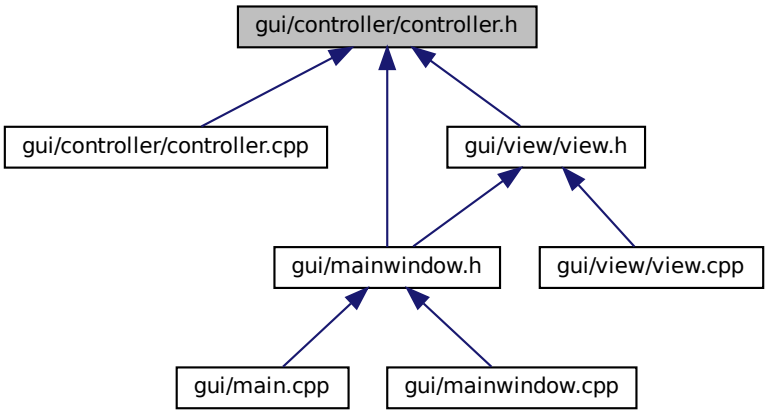
8.2 gui/controller/controller.h File Reference

```
#include "../../model/facade/facade.h"
```

Include dependency graph for controller.h:



This graph shows which files directly or indirectly include this file:



Classes

- class s21::Controller

  Provides an interface for interacting with Model in MVC pattern.

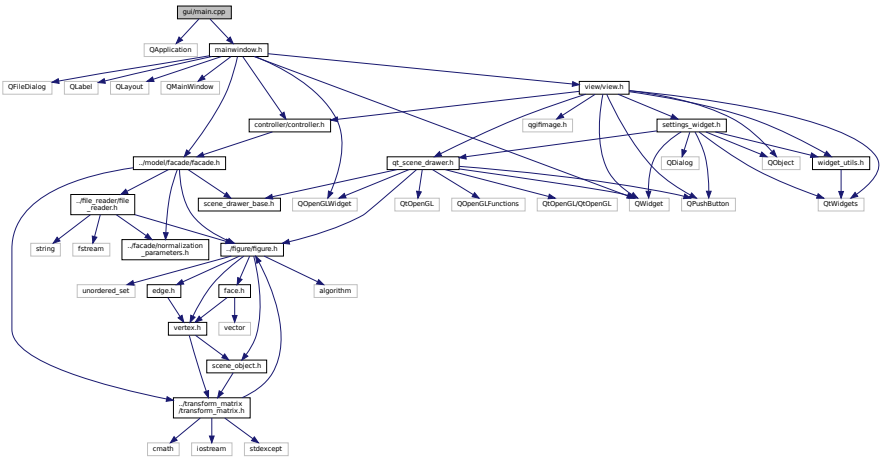
Namespaces

- s21

8.3 gui/main.cpp File Reference

```
#include <QApplication>
#include "mainwindow.h"
```

Include dependency graph for main.cpp:



Functions

- int main (int argc, char ∗argv[ ])

8.3.1 Function Documentation

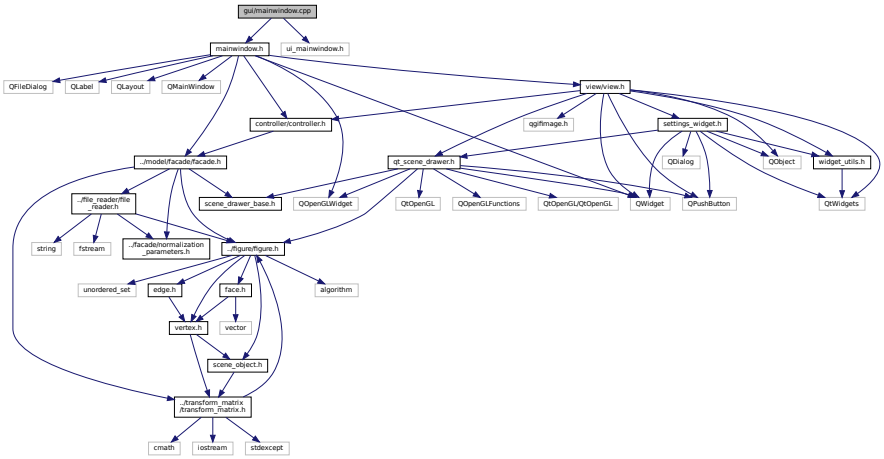
8.3.1.1 main()

```
int main (
            int argc,
            char * argv[] )
```

8.4 gui/mainwindow.cpp File Reference

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
```

Include dependency graph for mainwindow.cpp:



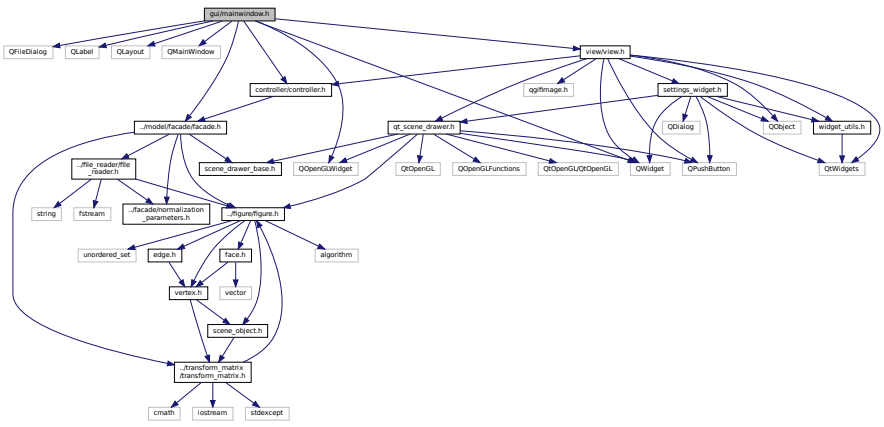
Namespaces

- s21

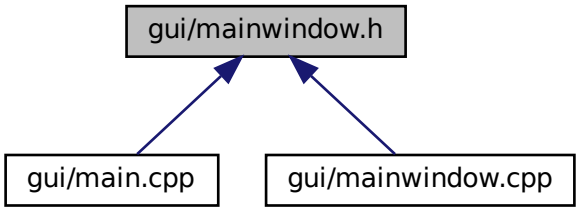
8.5 gui/mainwindow.h File Reference

```
#include <QFileDialog>
#include <QLabel>
#include <QLayout>
#include <QMainWindow>
#include <QOpenGLWidget>
#include <QWidget>
#include "../model/facade/facade.h"
#include "controller/controller.h"
#include "view/view.h"
```

Include dependency graph for mainwindow.h:



This graph shows which files directly or indirectly include this file:



Classes

- class s21::MainWindow

Namespaces

- Ui
- s21

Macros

- #define MAIN_WINDOW_WIDTH 1400
- #define MAIN_WINDOW_HEIGHT 830

8.5.1 Macro Definition Documentation

8.5.1.1 MAIN_WINDOW_HEIGHT

```
#define MAIN_WINDOW_HEIGHT 830
```

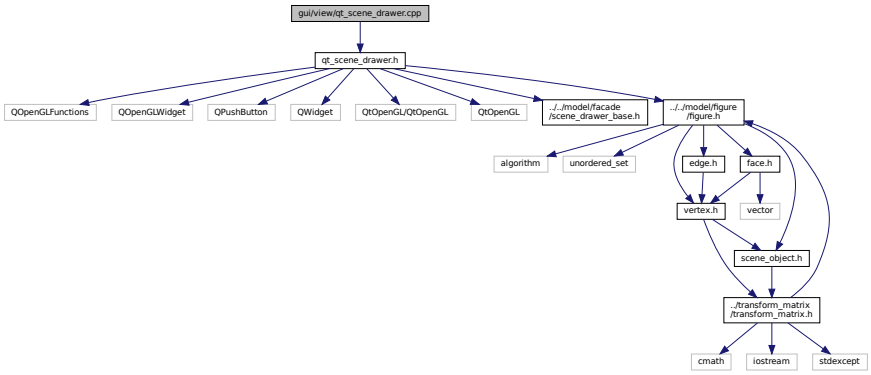
8.5.1.2 MAIN_WINDOW_WIDTH

```
#define MAIN_WINDOW_WIDTH 1400
```

8.6 gui/view/qt_scene_drawer.cpp File Reference

```
#include "qt_scene_drawer.h"
```

Include dependency graph for qt_scene_drawer.cpp:



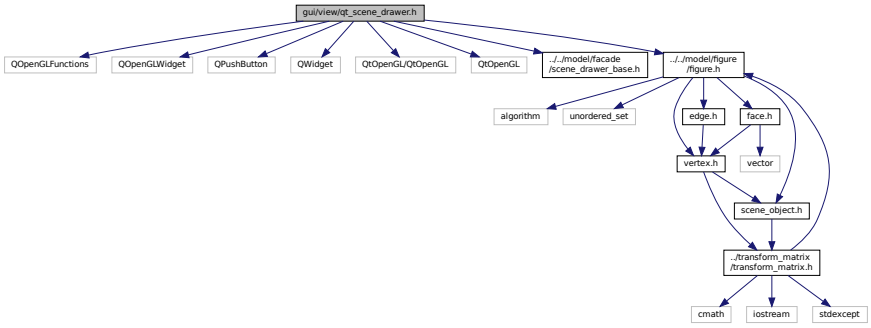
Namespaces

- s21

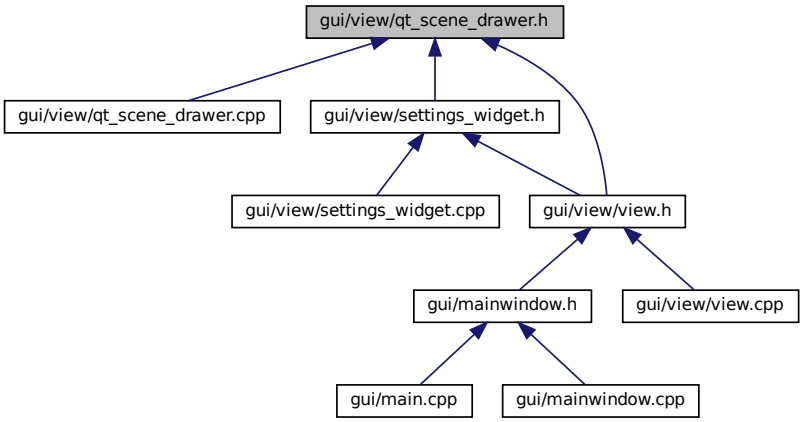
8.7 gui/view/qt_scene_drawer.h File Reference

```
#include <QOpenGLFunctions>
#include <QOpenGLWidget>
#include <QPushButton>
#include <QWidget>
#include <QtOpenGL/QtOpenGL>
#include <QtOpenGL>
#include "../../model/facade/scene_drawer_base.h"
#include "../../model/figure/figure.h"
```

Include dependency graph for qt_scene_drawer.h:



This graph shows which files directly or indirectly include this file:



Classes

- class s21::QtSceneDrawer

  The QtSceneDrawer is an OpenGL-based widget for rendering 3D figures.
- struct s21::QtSceneDrawer::RendParams

  Struct contains rendering parameters for the 3D scene (projection type, background color, edge and vertex rendering styles and colors).

Namespaces

- s21

Macros

- #define DRAWER_WIDTH 1000
- #define DRAWER_HEIGHT 800
- #define DRAWER_INDENT 5

8.7.1 Macro Definition Documentation

8.7.1.1 DRAWER_HEIGHT

```
#define DRAWER_HEIGHT 800
```

8.7.1.2 DRAWER_INDENT

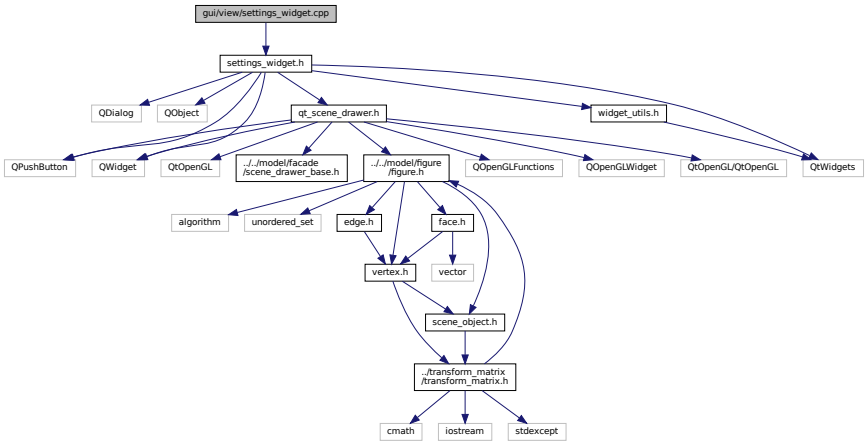
```
#define DRAWER_INDENT 5
```

8.7.1.3 DRAWER_WIDTH

```
#define DRAWER_WIDTH 1000
```

8.8 gui/view/settings_widget.cpp File Reference

```
#include "settings_widget.h"
```
Include dependency graph for settings_widget.cpp:



Namespaces

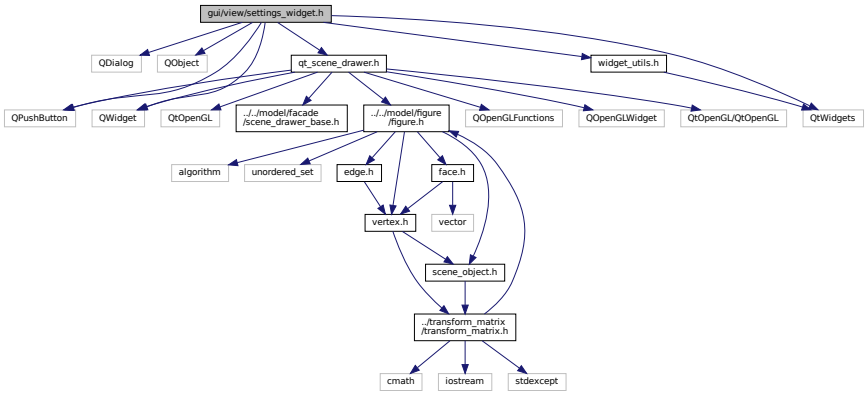
- s21

8.9 gui/view/settings_widget.h File Reference

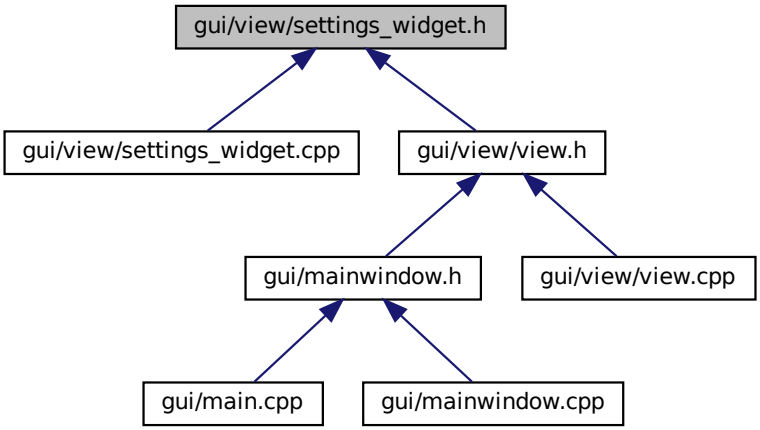
```
#include <QDialog>
#include <QObject>
#include <QPushButton>
#include <QWidget>
#include <QtWidgets>
#include "qt_scene_drawer.h"
```

```
#include "widget_utils.h"
```

Include dependency graph for settings_widget.h:



This graph shows which files directly or indirectly include this file:



Classes

- class s21::SettingsWidget

  A dialog widget that allows user to customize type, color and thickness of the edges, display method, color and size of the vertices of a 3D model.

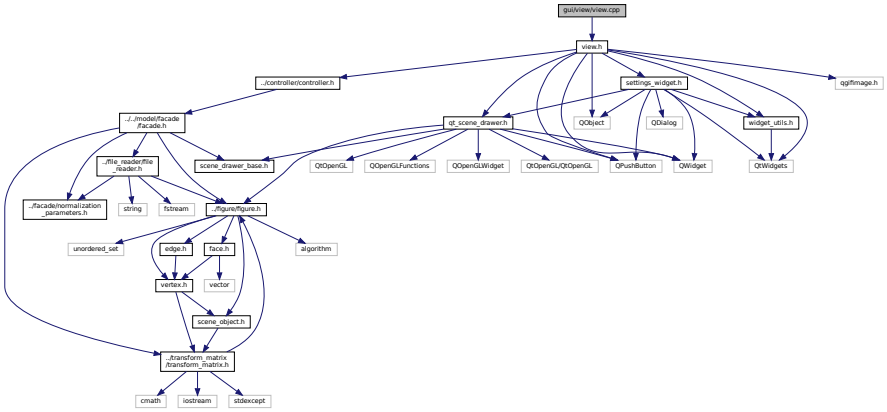
Namespaces

- s21

8.10 gui/view/view.cpp File Reference

```
#include "view.h"
```

Include dependency graph for view.cpp:



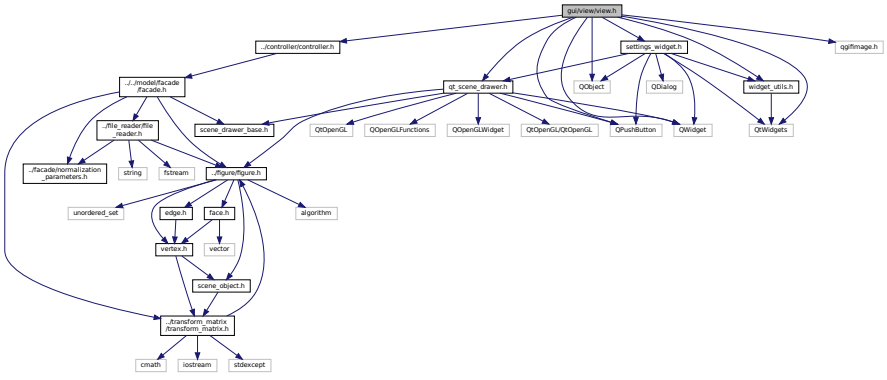
Namespaces

- s21

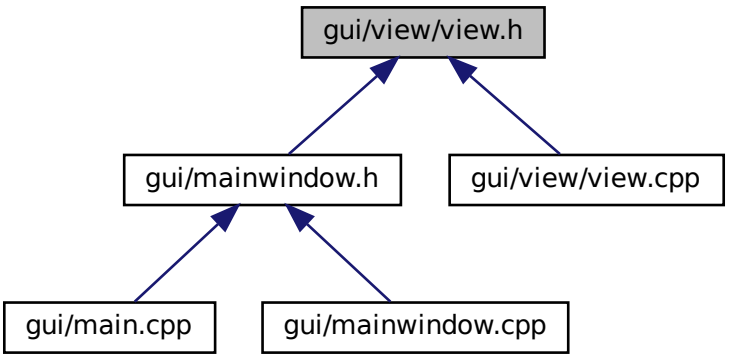
8.11 gui/view/view.h File Reference

```
#include <QObject>
#include <QPushButton>
#include <QWidget>
#include <QtWidgets>
#include "../controller/controller.h"
#include "qgifimage.h"
#include "qt_scene_drawer.h"
#include "settings_widget.h"
#include "widget_utils.h"
```

Include dependency graph for view.h:



This graph shows which files directly or indirectly include this file:



Classes

- class s21::View

  The View class is responsible for the GUI of the 3D viewer application and handling user interactions, such as opening and saving files, adjusting the 3D object's scale, position, and rotation, switching between parallel and central projection modes. It also includes a QtSceneDrawer object to render the 3D scene and a SettingsWidget object to manage the application's settings.
- struct s21::View::PrevPositions

  Keeps track of the previous positions of the scale, move, and rotate 3D object.

Namespaces

- s21

Macros

- #define START_SCALE 100.0f
- #define START_POS 0.0f
- #define START_ANGLE 0.0f

8.11.1 Macro Definition Documentation

8.11.1.1 START_ANGLE

```
#define START_ANGLE 0.0f
```

8.11.1.2 START_POS

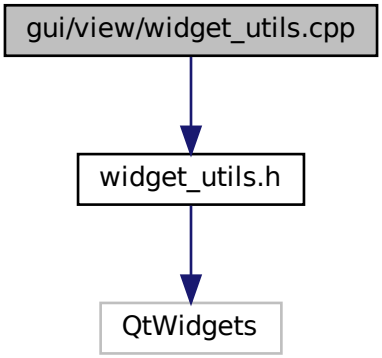
```
#define START_POS 0.0f
```

8.11.1.3 START_SCALE

```
#define START_SCALE 100.0f
```

8.12 gui/view/widget_utils.cpp File Reference

```
#include "widget_utils.h"
```
Include dependency graph for widget_utils.cpp:



Namespaces

- s21

Functions

- QPushButton * s21::setup_button (QWidget *parent, const QString &text, const QString &icon_path, const QSize &icon_size, int x, int y, int width, int height, const QFont &font)

  Creates a new QPushButton with the specified text, icon, geometry, and font. If an icon path is provided, the button will display the icon with the given size.
- QLabel * s21::setup_label (QWidget *parent, const QString &text, int x, int y, int width, int height, const QFont &font, Qt::Alignment align)

  Creates a new QLabel with the specified text, geometry, font, and alignment.
- QScrollBar * s21::setup_scrollbar (QWidget *parent, Qt::Orientation orient, int x, int y, int width, int height, int min, int max)

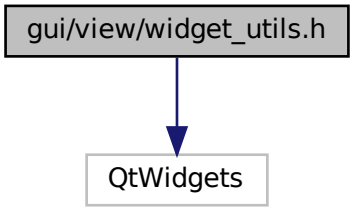
  Creates a new QScrollBar with the specified orientation, geometry, and range.
- QSpinBox * s21::setup_spinbox (QWidget *parent, int x, int y, int width, int height, int min, int max)

  Creates a new QSpinBox with the specified geometry and range.
- QComboBox * s21::setup_combobox (QWidget *parent, const QStringList &options, int x, int y, int width, int height, const QFont &font)

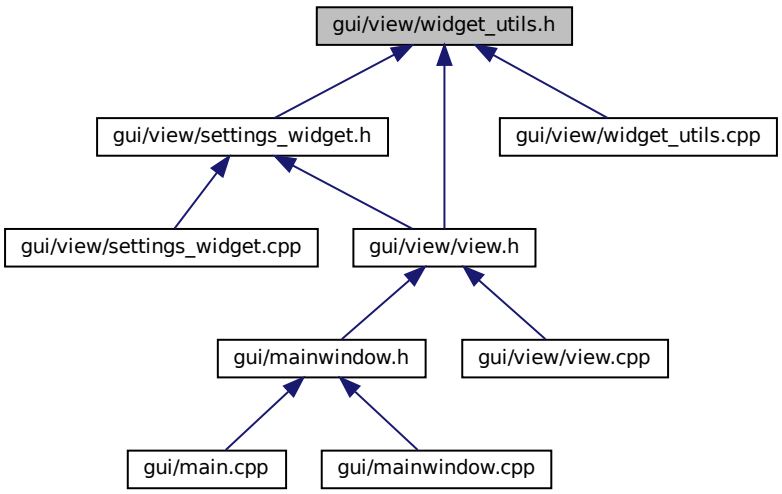
  Creates a new QComboBox with the specified parent, options, geometry, and font.

8.13 gui/view/widget_utils.h File Reference

`#include <QtWidgets>`
Include dependency graph for widget_utils.h:



This graph shows which files directly or indirectly include this file:



Namespaces

- s21

Functions

- QPushButton * s21::setup_button (QWidget *parent, const QString &text, const QString &icon_path, const QSize &icon_size, int x, int y, int width, int height, const QFont &font)

  Creates a new QPushButton with the specified text, icon, geometry, and font. If an icon path is provided, the button will display the icon with the given size.

- QLabel * s21::setup_label (QWidget *parent, const QString &text, int x, int y, int width, int height, const QFont &font, Qt::Alignment align)

  Creates a new QLabel with the specified text, geometry, font, and alignment.
- QScrollBar * s21::setup_scrollbar (QWidget *parent, Qt::Orientation orient, int x, int y, int width, int height, int min, int max)

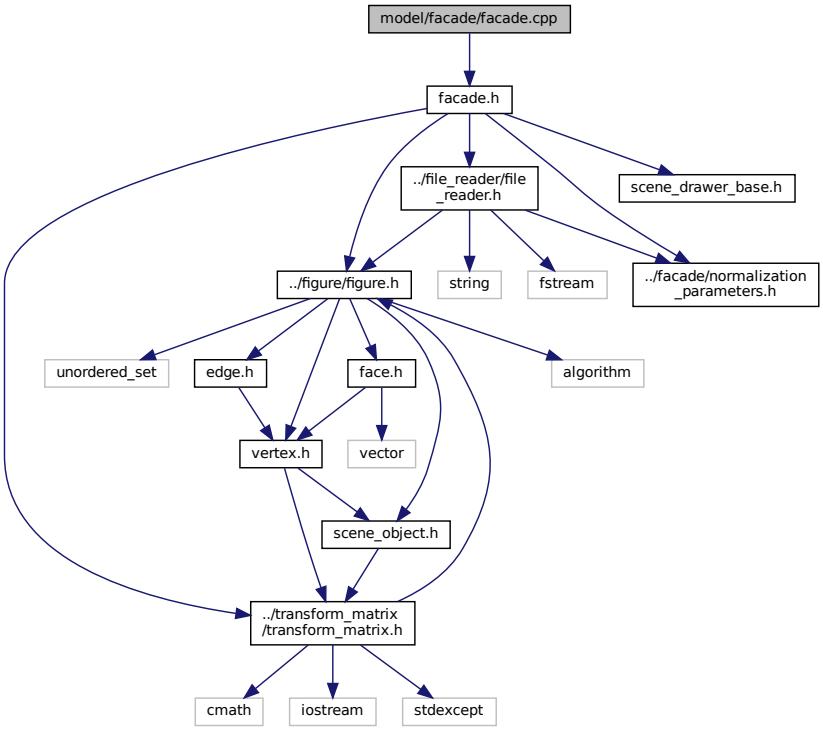
  Creates a new QScrollBar with the specified orientation, geometry, and range.
- QSpinBox * s21::setup_spinbox (QWidget *parent, int x, int y, int width, int height, int min, int max)

  Creates a new QSpinBox with the specified geometry and range.
- QComboBox * s21::setup_combobox (QWidget *parent, const QStringList &options, int x, int y, int width, int height, const QFont &font)

  Creates a new QComboBox with the specified parent, options, geometry, and font.

8.14 model/facade/facade.cpp File Reference

`#include "facade.h"`
Include dependency graph for facade.cpp:



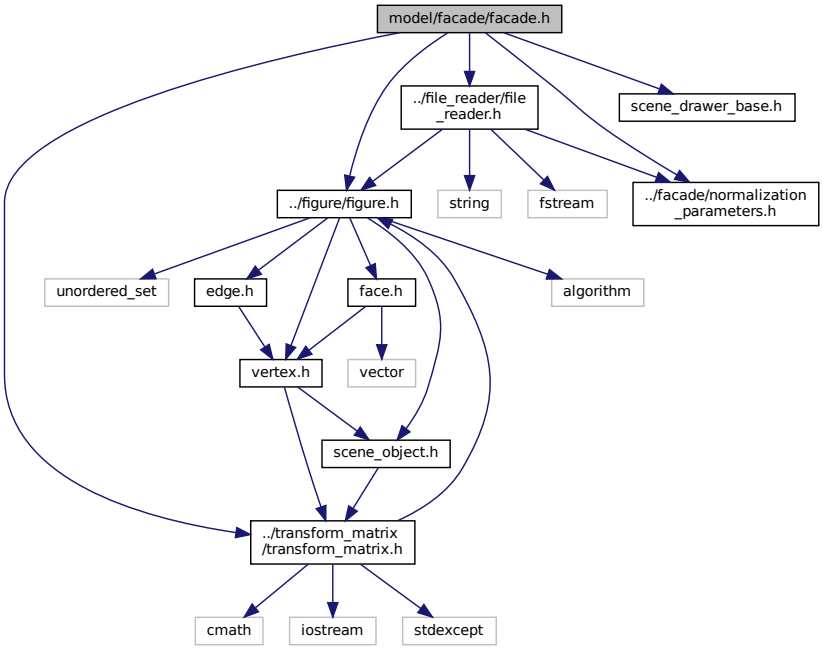
Namespaces

- s21

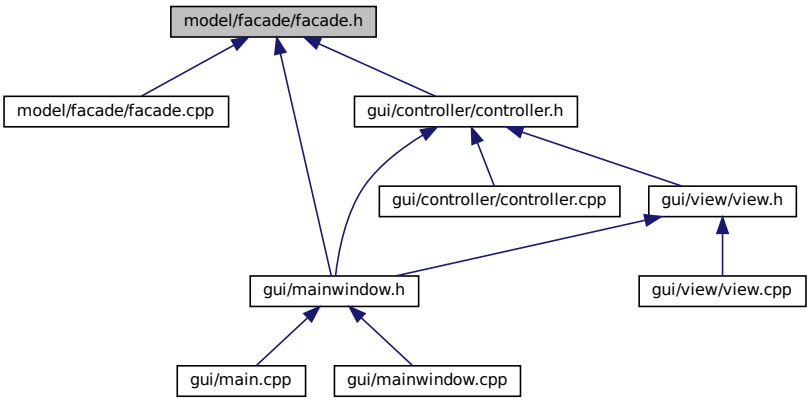
8.15 model/facade/facade.h File Reference

```
#include "../figure/figure.h"
#include "../file_reader/file_reader.h"
#include "../transform_matrix/transform_matrix.h"
#include "normalization_parameters.h"
#include "scene_drawer_base.h"
```

Include dependency graph for facade.h:



This graph shows which files directly or indirectly include this file:



Classes

- class s21::FacadeOperationResult

  Represents the result of an operation performed by the Facade class.
- class s21::Facade

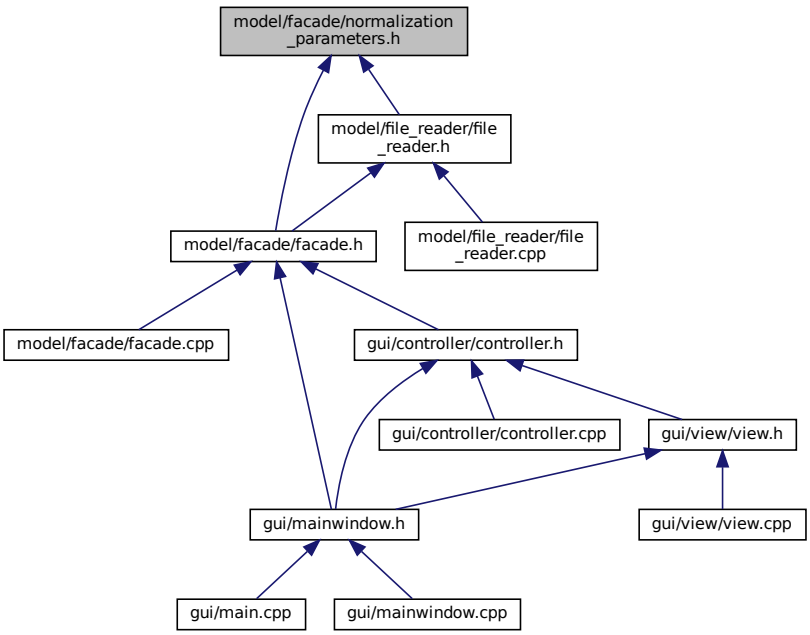
  Represents interface for interacting with Controller class.

Namespaces

- s21

8.16 model/facade/normalization_parameters.h File Reference

This graph shows which files directly or indirectly include this file:



Classes

- class s21::NormalizationParameters

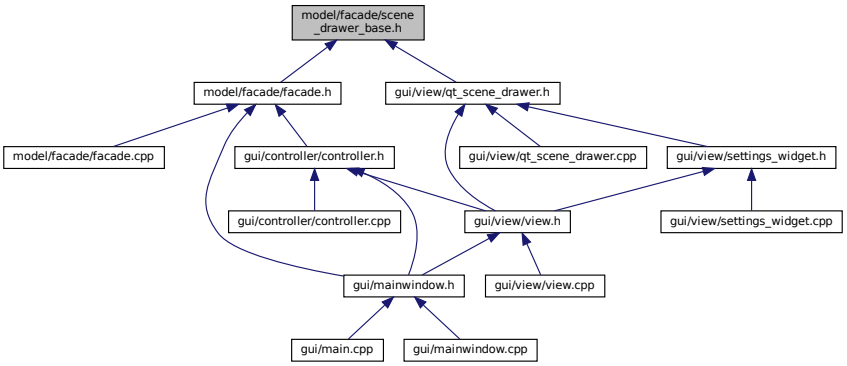
  Manages the normalization parameters for Figure objects.

Namespaces

- s21

8.17 model/facade/scene_drawer_base.h File Reference

This graph shows which files directly or indirectly include this file:



Classes

- class s21::SceneDrawerBase

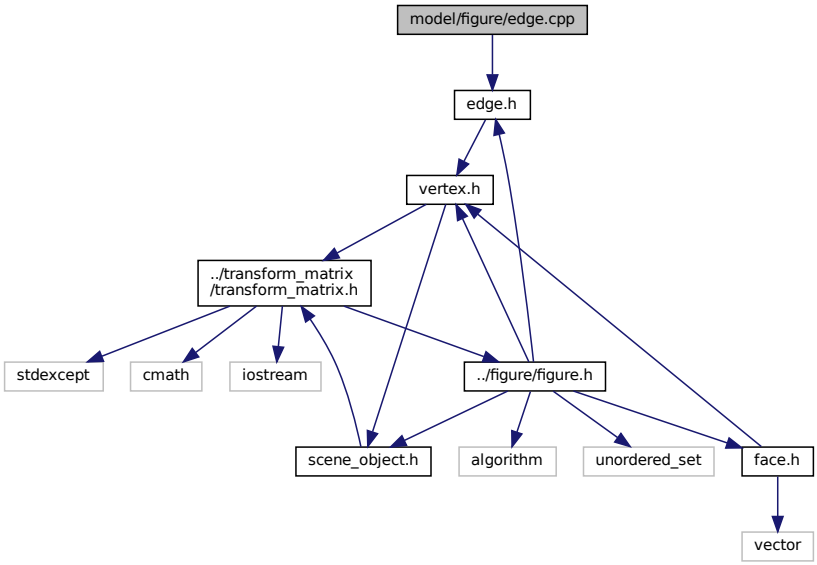
  Abstract base class for scene drawers.

Namespaces

- s21

8.18 model/figure/edge.cpp File Reference

`#include "edge.h"`
Include dependency graph for edge.cpp:

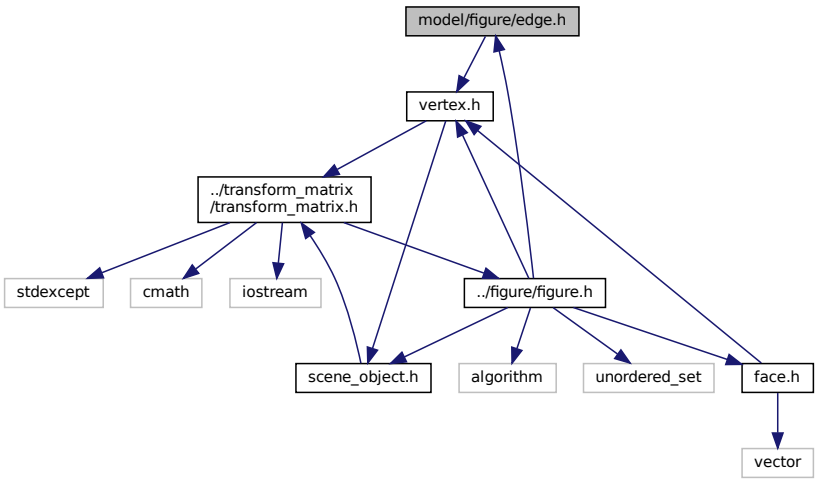


Namespaces

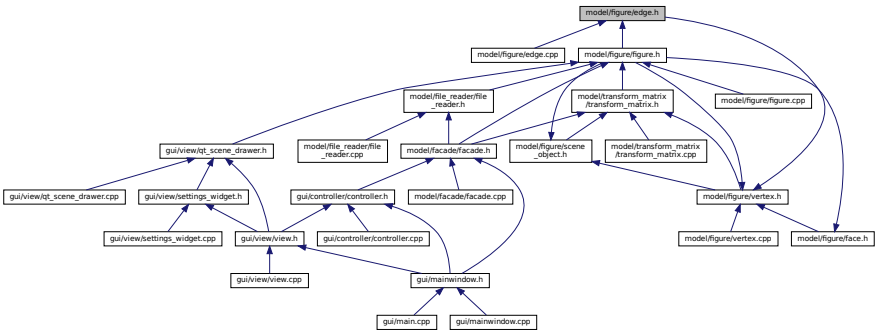
- s21

8.19 model/figure/edge.h File Reference

`#include "vertex.h"`
Include dependency graph for edge.h:



This graph shows which files directly or indirectly include this file:



Classes

- class s21::Edge

  Represents an edge in a figure, connecting two vertices.

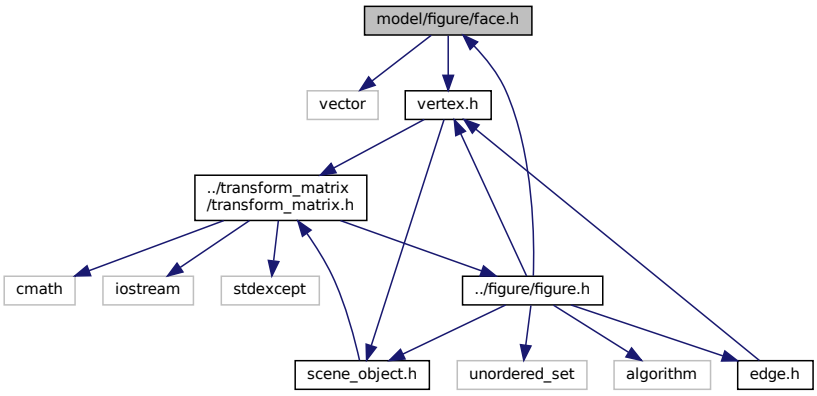
Namespaces

- s21

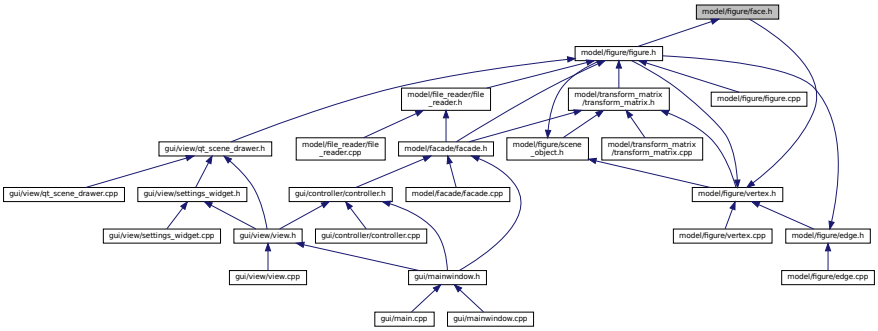
8.20 model/figure/face.h File Reference

```
#include <vector>
#include "vertex.h"
```

Include dependency graph for face.h:



This graph shows which files directly or indirectly include this file:



Classes

- class s21::Face

  Represents a face composed of a vector of unsigned integers.

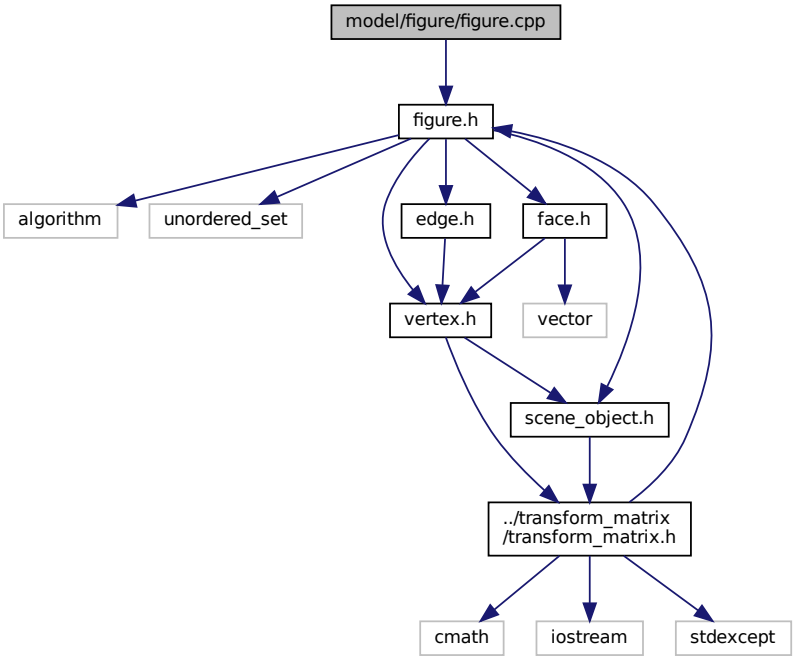
Namespaces

- s21

8.21 model/figure/figure.cpp File Reference

```
#include "figure.h"
```

Include dependency graph for figure.cpp:



Classes

- struct s21::PairHash

  Hash function for pairs of values. This struct provides a hash function for pairs of values, used to store edges in a hash set.

Namespaces

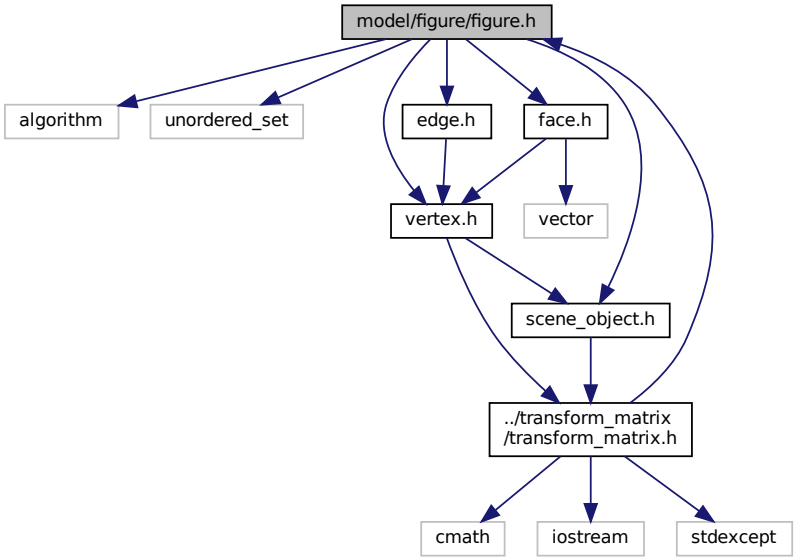
- s21

8.22 model/figure/figure.h File Reference

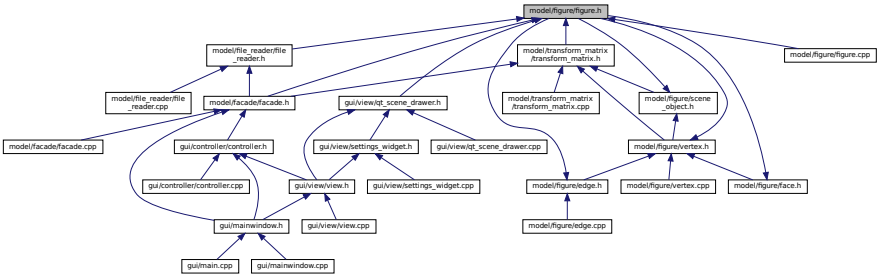
```
#include <algorithm>
#include <unordered_set>
#include "edge.h"
#include "face.h"
#include "scene_object.h"
```

```
#include "vertex.h"
```

Include dependency graph for figure.h:



This graph shows which files directly or indirectly include this file:



Classes

- class s21::Figure

  Represents a figure in a 3D scene, derived from SceneObject.

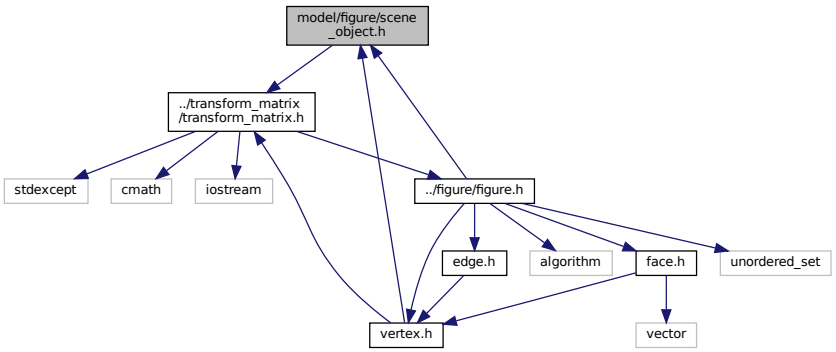
Namespaces

- s21

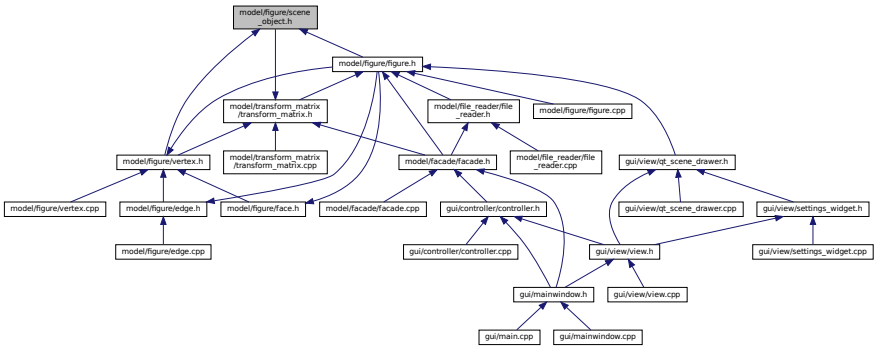
8.23 model/figure/scene_object.h File Reference

```
#include "../transform_matrix/transform_matrix.h"
```

Include dependency graph for scene_object.h:



This graph shows which files directly or indirectly include this file:



Classes

- class s21::SceneObject

  Abstract base class representing a scene object.

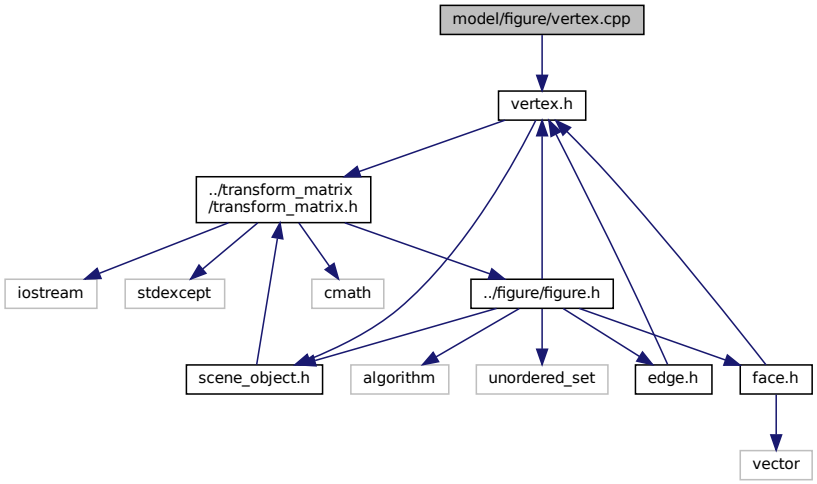
Namespaces

- s21

8.24 model/figure/vertex.cpp File Reference

```
#include "vertex.h"
```

Include dependency graph for vertex.cpp:



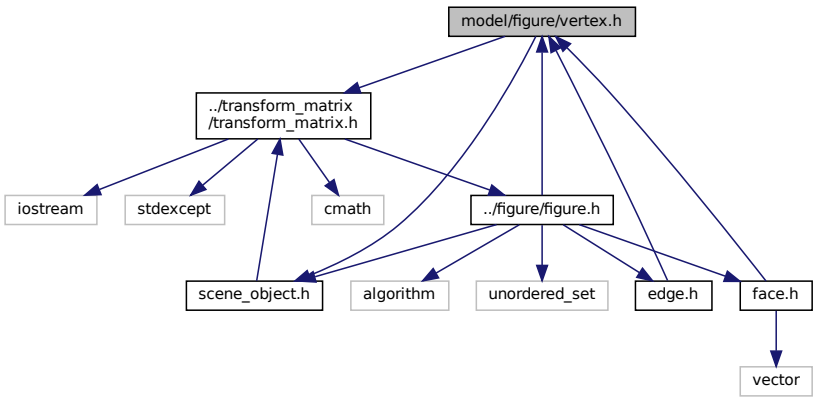
Namespaces

- s21

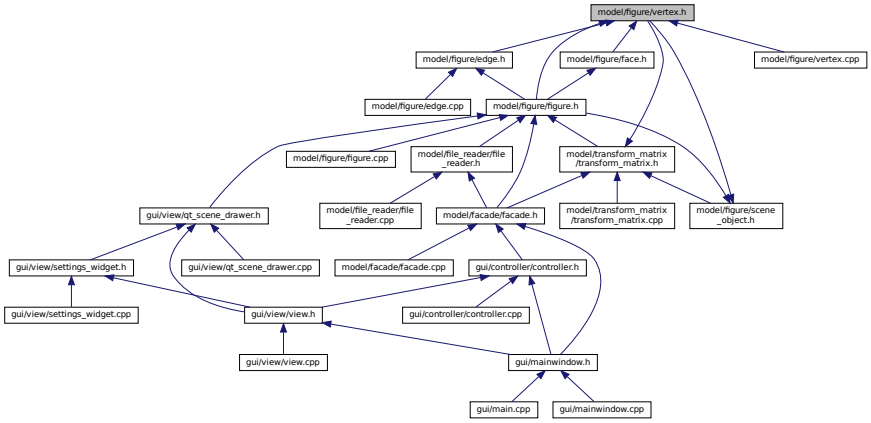
8.25 model/figure/vertex.h File Reference

```
#include "../transform_matrix/transform_matrix.h"
#include "scene_object.h"
```

Include dependency graph for vertex.h:



This graph shows which files directly or indirectly include this file:



Classes

- struct s21::Point

  Represents a point in 3D space with homogeneous coordinates.
- class s21::Vertex

  Represents a vertex in 3D space, derived from SceneObject.

Namespaces

- s21

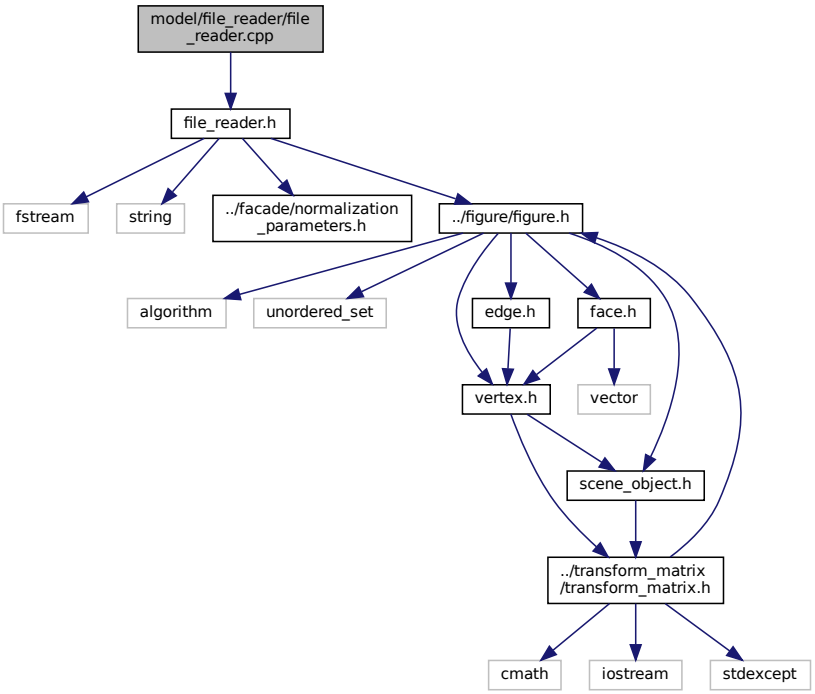
Typedefs

- typedef struct s21::Point s21::Point

8.26 model/file_reader/file_reader.cpp File Reference

```
#include "file_reader.h"
```

Include dependency graph for file_reader.cpp:



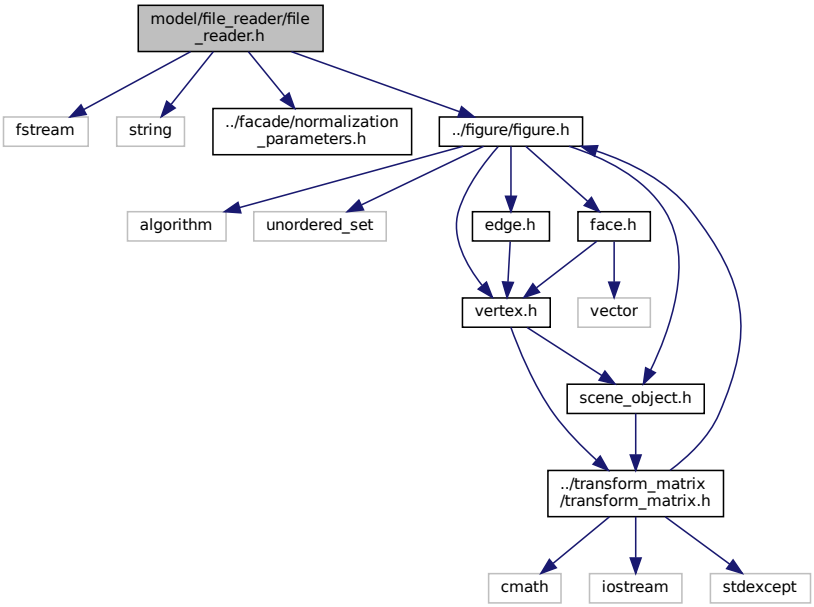
Namespaces

- s21

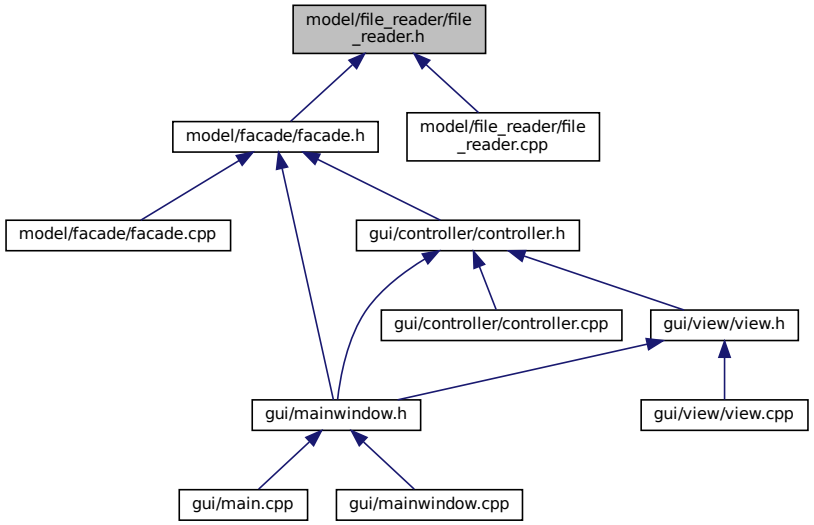
8.27 model/file_reader/file_reader.h File Reference

```
#include <fstream>
#include <string>
#include "../facade/normalization_parameters.h"
#include "../figure/figure.h"
```

Include dependency graph for file_reader.h:



This graph shows which files directly or indirectly include this file:



Classes

- class s21::FileReader

  A class for reading figure data from a file.
- class s21::FileReaderBuilder

Abstract base class for building a FileReader. Use Builder pattern.

- class s21::FigureBuilder

  A concrete builder for creating Figure objects.

- class s21::FileReaderDirector

  Constructs a figure using the provided builder.

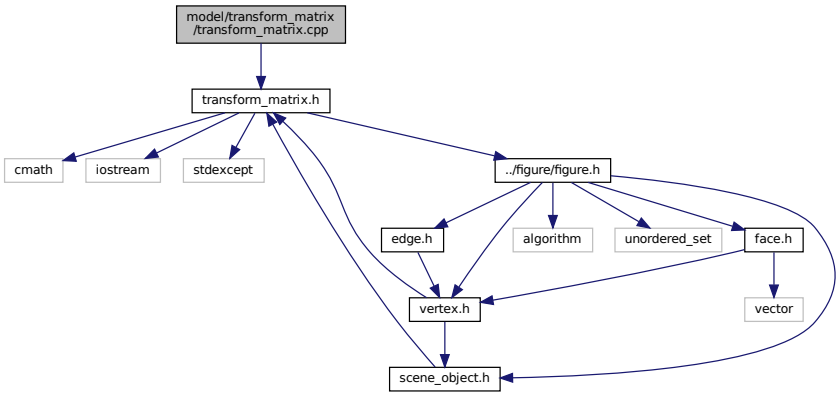
Namespaces

- s21

8.28 model/transform_matrix/transform_matrix.cpp File Reference

```
#include "transform_matrix.h"
```

Include dependency graph for transform_matrix.cpp:



Namespaces

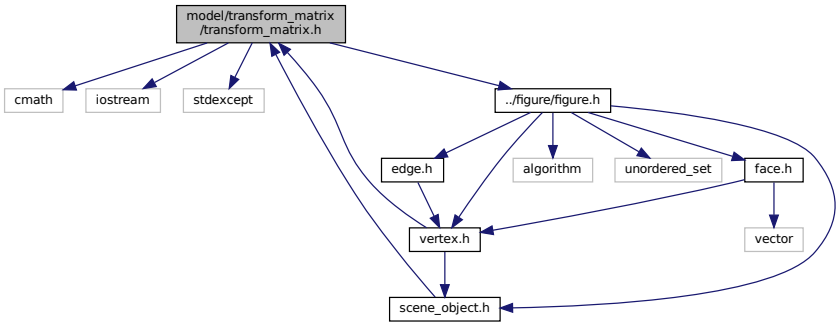
- s21

8.29 model/transform_matrix/transform_matrix.h File Reference

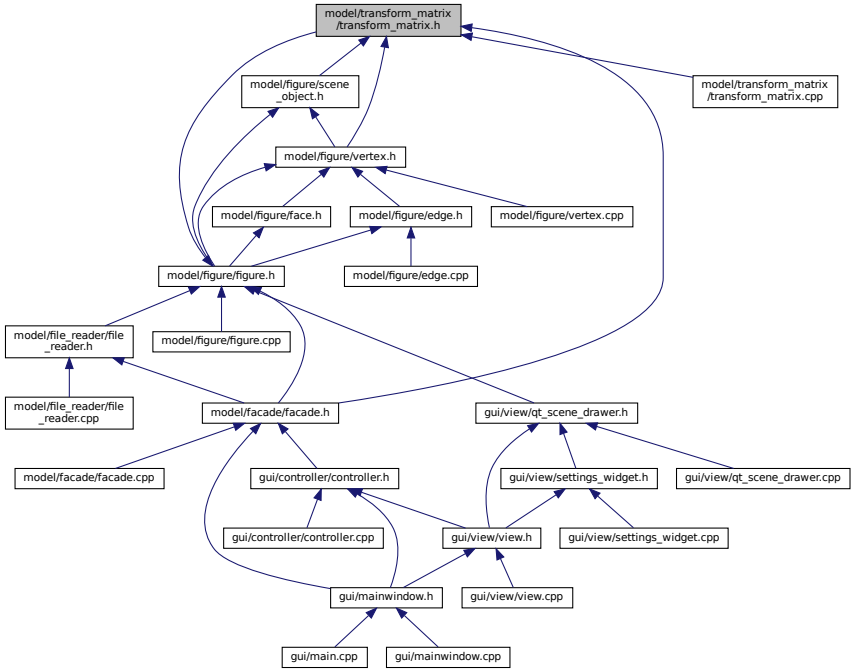
```
#include <cmath>
#include <iostream>
#include <stdexcept>
```

```
#include "../figure/figure.h"
```

Include dependency graph for transform_matrix.h:



This graph shows which files directly or indirectly include this file:



Classes

- class s21::TransformMatrix

  A class representing a 4x4 transformation matrix.
- class s21::TransformMatrixBuilder

  A utility class for creating transformation matrices.

Namespaces

- s21

Functions

- float deg_to_rad (float angle)

  Service function for getting radians from degrees. Use in transformation functions for matrix.

8.29.1 Function Documentation

8.29.1.1 deg_to_rad()

```
float deg_to_rad (
            float angle )  [inline]
```

Service function for getting radians from degrees. Use in transformation functions for matrix.

Parameters

angle	Angle (degrees) which need to be converted to radians.

Returns

Radians - float type.

8.30 readme_src/dvi_readme.md File Reference