Voxel World

0.02

Generated by Doxygen 1.8.14

Contents

Chapter 1

Class Index

1.1 Class List

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

Chapter 2

File Index

2.1 File List

Here is a list of all files with brief descriptions:

Chapter 3

Class Documentation

3.1 Camera Class Reference

`#include <Camera.hpp>`

Public Member Functions

- Camera (mat3 frame, vec3 pos, float w, float h)
- Camera ()
- ~Camera ()
- mat4 getViewMatrix ()
- mat4 getPerspectiveMatrix ()
- void setLookDirection (vec3 v)
- void move (vec3 v)
- void setPosition (vec3 p)
- void turnH (float angle)
- void turnV (float angle)
- void incline (float angle)
- void resetView ()
- void resetCamera ()
- vec3 getPosition ()
- vec3 getForward ()
- vec3 getUp ()
- vec3 getSide ()
- float getFov ()

Private Attributes

- vec3 forward
- vec3 up
- vec3 side
- vec3 position
- vec3 orig_forward
- vec3 orig_up
- vec3 orig_side
- vec3 orig_position
- float fov
- float width
- float height
- float zNear
- float zFar

3.1.1 Constructor & Destructor Documentation

3.1.1.1 Camera() [1/2]

```
Camera::Camera (
            mat3 frame,
            vec3 pos,
            float w,
            float h )
```

3.1.1.2 Camera() [2/2]

```
Camera::Camera ( )
```

3.1.1.3 ~Camera()

```
Camera::~Camera ( )
```

3.1.2 Member Function Documentation

3.1.2.1 getForward()

```
vec3 Camera::getForward ( )
```

3.1.2.2 getFov()

```
float Camera::getFov ( )
```

3.1.2.3 getPerspectiveMatrix()

```
mat4 Camera::getPerspectiveMatrix ( )
```

3.1.2.4 getPosition()

```
vec3 Camera::getPosition ( )
```

3.1.2.5 getSide()

```
vec3 Camera::getSide ( )
```

3.1.2.6 getUp()

```
vec3 Camera::getUp ( )
```

3.1.2.7 getViewMatrix()

```
mat4 Camera::getViewMatrix ( )
```

3.1.2.8 incline()

```
void Camera::incline (
            float angle )
```

3.1.2.9 move()

```
void Camera::move (
            vec3 v )
```

3.1.2.10 resetCamera()

```
void Camera::resetCamera ( )
```

3.1.2.11 resetView()

```
void Camera::resetView ( )
```

3.1.2.12 setLookDirection()

```
void Camera::setLookDirection (
            vec3 v )
```

3.1.2.13 setPosition()

```
void Camera::setPosition (
            vec3 p )
```

3.1.2.14 turnH()

```
void Camera::turnH (
            float angle )
```

3.1.2.15 turnV()

```
void Camera::turnV (
            float angle )
```

3.1.3 Member Data Documentation

3.1.3.1 forward

```
vec3 Camera::forward  [private]
```

3.1.3.2 fov

```
float Camera::fov  [private]
```

3.1.3.3 height

```
float Camera::height  [private]
```

3.1.3.4 orig_forward

```
vec3 Camera::orig_forward  [private]
```

3.1.3.5 orig_position

```
vec3 Camera::orig_position  [private]
```

3.1.3.6 orig_side

```
vec3 Camera::orig_side  [private]
```

3.1.3.7 orig_up

```
vec3 Camera::orig_up  [private]
```

3.1.3.8 position

```
vec3 Camera::position  [private]
```

3.1.3.9 side

```
vec3 Camera::side  [private]
```

3.1.3.10 up

```
vec3 Camera::up  [private]
```

3.1.3.11 width

```
float Camera::width  [private]
```

3.1.3.12 zFar

```
float Camera::zFar  [private]
```

3.1.3.13 zNear

```
float Camera::zNear  [private]
```

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

- Rendering/Camera/Camera.hpp
- Rendering/Camera/Camera.cpp

3.2 Chunk Class Reference

```
#include <World.hpp>
```

Public Member Functions

- Cube * operator() (int, int, int)
- Chunk ()
- Chunk (vec3)
- Chunk (vec3, World *)
- ~Chunk ()
- void create_cubes (vec3)
- void update ()
- void send_render_data (Renderer *)

Public Attributes

- vec3 position

Private Member Functions

- void update_visible_faces ()
- bool check_neighbour (Cube *c, Cube *n)

Private Attributes

- World * world
- Cube * chunk_cubes [CHUNK_DIMS *CHUNK_DIMS *CHUNK_DIMS] = {}
- Object_3D * render_data
- vector< vec4 > cubes_info

3.2.1 Constructor & Destructor Documentation

3.2.1.1 Chunk() [1/3]

```
Chunk::Chunk ( )
```

3.2.1.2 Chunk() [2/3]

```
Chunk::Chunk (
            vec3 offset )
```

3.2.1.3 Chunk() [3/3]

```
Chunk::Chunk (
            vec3 offset,
            World * w )
```

3.2.1.4 ~Chunk()

```
Chunk::~Chunk ( )
```

3.2.2 Member Function Documentation

3.2.2.1 check_neighbour()

```
bool Chunk::check_neighbour (
            Cube * c,
            Cube * n )  [inline], [private]
```

3.2.2.2 create_cubes()

```
void Chunk::create_cubes (
            vec3 offset )
```

3.2.2.3 operator()()

```
Cube * Chunk::operator() (
            int x,
            int y,
            int z )
```

3.2.2.4 send_render_data()

```
void Chunk::send_render_data (
            Renderer * handler )  [inline]
```

3.2.2.5 update()

```
void Chunk::update ( )
```

3.2.2.6 update_visible_faces()

```
void Chunk::update_visible_faces ( )  [private]
```

3.2.3 Member Data Documentation

3.2.3.1 chunk_cubes

```
Cube* Chunk::chunk_cubes[CHUNK_DIMS *CHUNK_DIMS *CHUNK_DIMS] = {}  [private]
```

3.2.3.2 cubes_info

```
vector<vec4> Chunk::cubes_info  [private]
```

3.2.3.3 position

```
vec3 Chunk::position
```

3.2.3.4 render_data

```
Object_3D* Chunk::render_data  [private]
```

3.2.3.5 world

```
World* Chunk::world  [private]
```

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

- World.hpp
- World.cpp

3.3 Chunk_Holder Class Reference

```
#include <World.hpp>
```

Public Member Functions

- Chunk_Holder ()
- Chunk_Holder (int, int, int, World *)
- ~Chunk_Holder ()
- Chunk * operator() (int, int, int)
- void shift (ivec3)

Private Attributes

- cirArray< cirArray< cirArray< Chunk * > > > chunkBox
- World * world

3.3.1 Constructor & Destructor Documentation

3.3.1.1 Chunk_Holder() [1/2]

```
Chunk_Holder::Chunk_Holder ( )
```

3.3.1.2 Chunk_Holder() [2/2]

```
Chunk_Holder::Chunk_Holder (
            int x_dim,
            int y_dim,
            int z_dim,
            World * w )
```

3.3.1.3 ~Chunk_Holder()

```
Chunk_Holder::~Chunk_Holder ( )
```

3.3.2 Member Function Documentation

3.3.2.1 operator()()

```
Chunk * Chunk_Holder::operator() (
            int x,
            int y,
            int z )
```

3.3.2.2 shift()

```
void Chunk_Holder::shift (
            ivec3 offset )
```

3.3.3 Member Data Documentation

3.3.3.1 chunkBox

```
cirArray<cirArray<cirArray<Chunk*> > > Chunk_Holder::chunkBox  [private]
```

3.3.3.2 world

```
World* Chunk_Holder::world  [private]
```

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

- World.hpp
- World.cpp

3.4 cirArray< T > Class Template Reference

```
#include <tools.hpp>
```

Public Member Functions

- cirArray ()
- cirArray (uint size)
- void shift (int)
- T & operator[ ] (int)
- void operator= (T)
- uint size ()

Private Attributes

- vector< T > array
- int start

3.4.1 Constructor & Destructor Documentation

3.4.1.1 cirArray() [1/2]

```
template<typename T >
cirArray< T >::cirArray ( )
```

3.4.1.2 cirArray() [2/2]

```
template<typename T >
cirArray< T >::cirArray (
            uint size )
```

3.4.2 Member Function Documentation

3.4.2.1 operator=()

```
template<typename T>
void cirArray< T >::operator= (
            T  )
```

3.4.2.2 operator[]()

```
template<typename T >
T & cirArray< T >::operator[] (
            int i )
```

3.4.2.3 shift()

```
template<typename T >
void cirArray< T >::shift (
            int i )
```

3.4.2.4 size()

```
template<typename T >
uint cirArray< T >::size ( )
```

3.4.3 Member Data Documentation

3.4.3.1 array

```
template<typename T>
vector<T> cirArray< T >::array  [private]
```

3.4.3.2 start

```
template<typename T>
int cirArray< T >::start  [private]
```

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

- Helpers/tools.hpp

3.5 Cube Class Reference

```
#include <Cube.hpp>
```

Public Member Functions

- void update (vec3 offset)
- Cube (vec3 p, CubeID type)
- Cube (vec3 p)
- Cube ()
- ~Cube ()
- Mesh getMesh ()

Static Public Member Functions

- static void initialize ()
- static void cleanup ()

Public Attributes

- vec3 position
- CubeID cube_type = DEFAULT
- bool transparent = false

Static Public Attributes

- static vector< Mesh * > meshes
- static vector< Texture * > textures

3.5.1 Constructor & Destructor Documentation

3.5.1.1 Cube() [1/3]

```
Cube::Cube (
            vec3 p,
            CubeID type )
```

3.5.1.2 Cube() [2/3]

```
Cube::Cube (
            vec3 p )
```

3.5.1.3 Cube() [3/3]

```
Cube::Cube ( )
```

3.5.1.4 ~Cube()

```
Cube::~Cube ( )
```

3.5.2 Member Function Documentation

3.5.2.1 cleanup()

```
void Cube::cleanup ( )  [static]
```

3.5.2.2 getMesh()

```
Mesh Cube::getMesh ( )
```

3.5.2.3 initialize()

```
void Cube::initialize ( )  [static]
```

3.5.2.4 update()

```
void Cube::update (
            vec3 offset )
```

3.5.3 Member Data Documentation

3.5.3.1 cube_type

```
CubeID Cube::cube_type = DEFAULT
```

3.5.3.2 meshes

```
vector< Mesh * > Cube::meshes  [static]
```

3.5.3.3 position

```
vec3 Cube::position
```

3.5.3.4 textures

```
vector< Texture * > Cube::textures  [static]
```

3.5.3.5 transparent

```
bool Cube::transparent = false
```

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

- Cube.hpp
- Cube.cpp

3.6 Light Struct Reference

`#include <World.hpp>`

Public Attributes

- vec3 position
- vec4 color
- double intensity

3.6.1 Member Data Documentation

3.6.1.1 color

`vec4 Light::color`

3.6.1.2 intensity

`double Light::intensity`

3.6.1.3 position

`vec3 Light::position`

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

- World.hpp

3.7 Mesh Struct Reference

`#include <OpenGL-Wrappers.hpp>`

Public Member Functions

- ~Mesh ()

Public Attributes

- vector< vec3 > vertices
- vector< vec3 > normals
- vector< uint > indices
- vector< vec2 > uvs

3.7.1 Constructor & Destructor Documentation

3.7.1.1 ~Mesh()

```
Mesh::~Mesh ( )
```

3.7.2 Member Data Documentation

3.7.2.1 indices

```
vector<uint> Mesh::indices
```

3.7.2.2 normals

```
vector<vec3> Mesh::normals
```

3.7.2.3 uvs

```
vector<vec2> Mesh::uvs
```

3.7.2.4 vertices

```
vector<vec3> Mesh::vertices
```

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

- Rendering/OpenGL-Wrappers.hpp
- Rendering/OpenGL-Wrappers.cpp

3.8 Object_3D Class Reference

`#include <OpenGL-Wrappers.hpp>`

Public Member Functions

- Object_3D (Mesh *)
- template<class T >
  void set_instance_data (Renderer *, vector< T >)

Public Attributes

- GLuint VAO
- vector< GLuint > VBOs
- vector< GLuint > types
- uint layouts
- uint render_instances
- uint mesh_indices

3.8.1 Constructor & Destructor Documentation

3.8.1.1 Object_3D()

```
Object_3D::Object_3D (
            Mesh * mesh )
```

3.8.2 Member Function Documentation

3.8.2.1 set_instance_data()

```
template<class T >
void Object_3D::set_instance_data (
            Renderer * handler,
            vector< T > info )
```

3.8.3 Member Data Documentation

3.8.3.1 layouts

```
uint Object_3D::layouts
```

3.8.3.2 mesh_indices

```
uint Object_3D::mesh_indices
```

3.8.3.3 render_instances

```
uint Object_3D::render_instances
```

3.8.3.4 types

```
vector<GLuint> Object_3D::types
```

3.8.3.5 VAO

```
GLuint Object_3D::VAO
```

3.8.3.6 VBOs

```
vector<GLuint> Object_3D::VBOs
```

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

- Rendering/OpenGL-Wrappers.hpp
- Rendering/OpenGL-Wrappers.cpp

3.9 Renderer Class Reference

```
#include <OpenGL-Wrappers.hpp>
```

Public Member Functions

- Renderer ()
- Renderer (int width, int height)
- ~Renderer ()
- Shader * find_shader (string shader_name)
- void update (GLFWwindow *window)
- void add_Shader (string shader, GLuint type)
- void make_program (vector< uint > *shaders)
- void set_camera (Camera *new_cam)
- void multi_render (GLuint VAO, vector< GLuint > *VBOs, vector< GLuint > *buffer_types, GLuint layout_num, GLuint index_num, GLuint instances)
- void change_active_program (GLuint newProgram)
- void add_data (Object_3D *)
- void render ()
- void clear ()

Public Attributes

- mutex busy_queue
- Camera * cam
- GLuint current_program

Private Attributes

- vector< GLuint > shading_programs
- vector< Shader > vertex_shaders
- vector< Shader > fragment_shaders
- vector< Shader > tessellation_shaders
- vector< Object_3D * > render_queue

3.9.1 Constructor & Destructor Documentation

3.9.1.1 Renderer() [1/2]

```
Renderer::Renderer ( )
```

3.9.1.2 Renderer() [2/2]

```
Renderer::Renderer (
            int width,
            int height )
```

3.9.1.3 ~Renderer()

```
Renderer::~Renderer ( )
```

3.9.2 Member Function Documentation

3.9.2.1 add_data()

```
void Renderer::add_data (
            Object_3D * data )
```

3.9.2.2 add_Shader()

```
void Renderer::add_Shader (
            string shader,
            GLuint type )
```

3.9.2.3 change_active_program()

```
void Renderer::change_active_program (
            GLuint newProgram )
```

3.9.2.4 clear()

```
void Renderer::clear ( )
```

3.9.2.5 find_shader()

```
Shader * Renderer::find_shader (
            string shader_name )
```

3.9.2.6 make_program()

```
void Renderer::make_program (
            vector< uint > * shaders )
```

3.9.2.7 multi_render()

```
void Renderer::multi_render (
            GLuint VAO,
            vector< GLuint > * VBOs,
            vector< GLuint > * buffer_types,
            GLuint layout_num,
            GLuint index_num,
            GLuint instances )
```

3.9.2.8 render()

```
void Renderer::render ( )
```

3.9.2.9 set_camera()

```
void Renderer::set_camera (
            Camera * new_cam )
```

3.9.2.10 update()

```
void Renderer::update (
            GLFWwindow * window )
```

3.9.3 Member Data Documentation

3.9.3.1 busy_queue

```
mutex Renderer::busy_queue
```

3.9.3.2 cam

```
Camera* Renderer::cam
```

3.9.3.3 current_program

```
GLuint Renderer::current_program
```

3.9.3.4 fragment_shaders

```
vector<Shader> Renderer::fragment_shaders  [private]
```

3.9.3.5 render_queue

```
vector<Object_3D*> Renderer::render_queue  [private]
```

3.9.3.6 shading_programs

```
vector<GLuint> Renderer::shading_programs  [private]
```

3.9.3.7 tessellation_shaders

```
vector<Shader> Renderer::tessellation_shaders  [private]
```

3.9.3.8 vertex_shaders

```
vector<Shader> Renderer::vertex_shaders  [private]
```

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

- Rendering/OpenGL-Wrappers.hpp
- Rendering/OpenGL-Wrappers.cpp

3.10 Shader Class Reference

```
#include <OpenGL-Wrappers.hpp>
```

Public Member Functions

- Shader ()
- Shader (string file, GLenum type)
- ~Shader ()
- string load_from_file (string &)
- void clear ()

Public Attributes

- string fileName
- GLuint shaderID
- GLuint type

3.10.1 Constructor & Destructor Documentation

3.10.1.1 Shader() [1/2]

```
Shader::Shader ( )
```

3.10.1.2 Shader() [2/2]

```
Shader::Shader (
            string file,
            GLenum type )
```

3.10.1.3 ~Shader()

```
Shader::~Shader ( )
```

3.10.2 Member Function Documentation

3.10.2.1 clear()

```
void Shader::clear ( )
```

3.10.2.2 load_from_file()

```
string Shader::load_from_file (
            string & filepath )
```

3.10.3 Member Data Documentation

3.10.3.1 fileName

```
string Shader::fileName
```

3.10.3.2 shaderID

```
GLuint Shader::shaderID
```

3.10.3.3 type

```
GLuint Shader::type
```

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

- Rendering/OpenGL-Wrappers.hpp
- Rendering/OpenGL-Wrappers.cpp

3.11 Texture Class Reference

```
#include <OpenGL-Wrappers.hpp>
```

Public Member Functions

- Texture (const char *filename, GLuint target=GL_TEXTURE_2D)
- ~Texture ()
- void load_to_GPU (GLuint)
- void clear ()

Public Attributes

- GLuint textureID
- GLuint target
- string texture
- int width
- int height

3.11.1 Constructor & Destructor Documentation

3.11.1.1 Texture()

```
Texture::Texture (
            const char * filename,
            GLuint target = GL_TEXTURE_2D )
```

3.11.1.2 ~Texture()

```
Texture::~Texture ( )
```

3.11.2 Member Function Documentation

3.11.2.1 clear()

```
void Texture::clear ( )
```

3.11.2.2 load_to_GPU()

```
void Texture::load_to_GPU (
            GLuint program )
```

3.11.3 Member Data Documentation

3.11.3.1 height

```
int Texture::height
```

3.11.3.2 target

```
GLuint Texture::target
```

3.11.3.3 texture

```
string Texture::texture
```

3.11.3.4 textureID

```
GLuint Texture::textureID
```

3.11.3.5 width

```
int Texture::width
```

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

- Rendering/OpenGL-Wrappers.hpp
- Rendering/OpenGL-Wrappers.cpp

3.12 World Class Reference

```
#include <World.hpp>
```

Public Member Functions

- World ()
- ~World ()
- Cube * operator() (int x, int y, int z)
- void center_frame (ivec3 offset)
- void send_render_data (Renderer *)

Public Attributes

- int h_radius = 7
- int v_radius = 4
- ivec3 origin = ivec3(0)

Private Attributes

- Chunk_Holder * loaded_chunks
- vector< Light > loaded_lights

3.12.1 Constructor & Destructor Documentation

3.12.1.1 World()

```
World::World ( )
```

3.12.1.2 ~World()

```
World::~World ( )
```

3.12.2 Member Function Documentation

3.12.2.1 center_frame()

```
void World::center_frame (
            ivec3 offset )
```

3.12.2.2 operator()()

```
Cube * World::operator() (
            int x,
            int y,
            int z )
```

3.12.2.3 send_render_data()

```
void World::send_render_data (
            Renderer * handler )
```

3.12.3 Member Data Documentation

3.12.3.1 h_radius

```
int World::h_radius = 7
```

3.12.3.2 loaded_chunks

```
Chunk_Holder* World::loaded_chunks  [private]
```

3.12.3.3 loaded_lights

```
vector<Light> World::loaded_lights  [private]
```

3.12.3.4 origin

```
ivec3 World::origin = ivec3(0)
```

3.12.3.5 v_radius

```
int World::v_radius = 4
```

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

- World.hpp
- World.cpp

Chapter 4

File Documentation

4.1 Cube.cpp File Reference

```
#include "system-libraries.hpp"
#include "Cube.hpp"
#include "cout-definitions.hpp"
```

Variables

- vector< string > texture_source_files = {"Assets/Textures/white_cube.png"}
- vector< string > obj_source_files = {"Assets/Objs/cube.obj"}

4.1.1 Variable Documentation

4.1.1.1 obj_source_files

```
vector<string> obj_source_files = {"Assets/Objs/cube.obj"}
```

4.1.1.2 texture_source_files

```
vector<string> texture_source_files = {"Assets/Textures/white_cube.png"}
```

4.2 Cube.hpp File Reference

```
#include <string>
#include "OpenGL-Wrappers.hpp"
#include "wavefront-loader.hpp"
```

Classes

- class Cube

Enumerations

- enum CubeID { DEFAULT =0 }

Variables

- const uint cube_types = 1

4.2.1 Enumeration Type Documentation

4.2.1.1 CubeID

```
enum CubeID
```

Enumerator

DEFAULT	

4.2.2 Variable Documentation

4.2.2.1 cube_types

```
const uint cube_types = 1
```

4.3 Helpers/cout-definitions.cpp File Reference

```
#include "cout-definitions.hpp"
```

Functions

- ostream & operator<< (ostream &os, vec2 &v)
- ostream & operator<< (ostream &os, vec3 &v)
- ostream & operator<< (ostream &os, vec4 &v)
- ostream & operator<< (ostream &os, vector< float > &v)

4.3.1 Function Documentation

4.3.1.1 operator<<() [1/4]

```
ostream& operator<< (
            ostream & os,
            vec2 & v )
```

Print a vec2

4.3.1.2 operator<<() [2/4]

```
ostream& operator<< (
            ostream & os,
            vec3 & v )
```

Print a vec3

4.3.1.3 operator<<() [3/4]

```
ostream& operator<< (
            ostream & os,
            vec4 & v )
```

Print a vec4

4.3.1.4 operator<<() [4/4]

```
ostream& operator<< (
            ostream & os,
            vector< float > & v )
```

Print a vector of floats

4.4 Helpers/cout-definitions.hpp File Reference

test

```
#include <iostream>
#include <fstream>
#include <cstdlib>
#include <glm/glm.hpp>
#include <glm/gtc/matrix_transform.hpp>
#include <glm/gtx/transform.hpp>
#include <glm/gtc/type_ptr.hpp>
#include <string>
#include <vector>
#include <unistd.h>
```

Functions

- ostream & operator<< (ostream &os, vec2 &v)
- ostream & operator<< (ostream &os, vec3 &v)
- ostream & operator<< (ostream &os, vec4 &v)
- ostream & operator<< (ostream &os, vector< float > &v)

4.4.1 Detailed Description

test

Author

: Camilo Talero

Version: 0.0.2

Implementation of the output functions for I/O debugging

4.4.2 Function Documentation

4.4.2.1 operator<<() [1/4]

```
ostream& operator<< (
            ostream & os,
            vec2 & v )
```

Print a vec2

4.4.2.2 operator<<() [2/4]

```
ostream& operator<< (
            ostream & os,
            vec3 & v )
```

Print a vec3

4.4.2.3 operator<<() [3/4]

```
ostream& operator<< (
            ostream & os,
            vec4 & v )
```

Print a vec4

4.4.2.4 operator<<() [4/4]

```
ostream& operator<< (
            ostream & os,
            vector< float > & v )
```

Print a vector of floats

4.5 Helpers/system-libraries.hpp File Reference

```
#include <GL/glew.h>
#include <GLFW/glfw3.h>
#include <string>
#include <iostream>
#include <vector>
#include <fstream>
#include <cstdlib>
#include <unistd.h>
#include <time.h>
#include <thread>
#include <mutex>
#include <math.h>
#include <chrono>
#include <ctime>
#include <glm/glm.hpp>
#include <glm/gtc/matrix_transform.hpp>
#include <glm/gtx/transform.hpp>
#include <glm/gtc/type_ptr.hpp>
#include <ft2build.h>
```

Macros

- #define GLEW_DYNAMIC

4.5.1 Macro Definition Documentation

4.5.1.1 GLEW_DYNAMIC

```
#define GLEW_DYNAMIC
```

4.6 Helpers/tools.cpp File Reference

```
#include "tools.hpp"
```

Functions

- void vec_field_init ()
- double fade (double d)
- double length (double x, double y)
- double surflet (double x, double y, double grad_x, double grad_y)
- double perlin_noise (double x, double y)
- double noise_2D (double x, double y)

Variables

- int const size = 256
- int const mask = size-1
- int perm [size]
- float vec_field_x [size]
- float vec_field_y [size]

4.6.1 Function Documentation

4.6.1.1 fade()

```
double fade (
            double d )  [inline]
```

Function to smooth out the transition from each gridd cell to another f(x)=1-6∗|x|^5-15|x|^4+10|x|^3

4.6.1.2 length()

```
double length (
            double x,
            double y )  [inline]
```

Return the length of the vector (x,y) for radial fading.

4.6.1.3 noise_2D()

```
double noise_2D (
            double x,
            double y )
```

Composite 2D noise function. Combines multiple iterations of Perlin noise at different sampling rates and amplitudes and merges them using octaves to create more complex noise functions

4.6.1.4 perlin_noise()

```
double perlin_noise (
            double x,
            double y )
```

2D Perlin Noise funtion

4.6.1.5 surflet()

```
double surflet (
            double x,
            double y,
            double grad_x,
            double grad_y )  [inline]
```

2D convolution surflet function, returns a scalar based on the gradient at (x,y)

4.6.1.6 vec_field_init()

```
void vec_field_init ( )
```

Initialize the perlin noise grid. We basically rotate a 2D vector 2PI units in the counter clockwise direction and assign a random location to it in a lookup table

4.6.2 Variable Documentation

4.6.2.1 mask

```
int const mask = size-1
```

4.6.2.2 perm

```
int perm[size]
```

4.6.2.3 size

```
int const size = 256
```

4.6.2.4 vec_field_x

```
float vec_field_x[size]
```

4.6.2.5 vec_field_y

```
float vec_field_y[size]
```

4.7 Helpers/tools.hpp File Reference

```
#include "system-libraries.hpp"
```

Classes

- class cirArray< T >

Functions

- double noise_2D (double x, double y)
- void vec_field_init ()

4.7.1 Function Documentation

4.7.1.1 noise_2D()

```
double noise_2D (
            double x,
            double y )
```

Composite 2D noise function. Combines multiple iterations of Perlin noise at different sampling rates and amplitudes and merges them using octaves to create more complex noise functions

4.7.1.2 vec_field_init()

```
void vec_field_init ( )
```

Initialize the perlin noise grid. We basically rotate a 2D vector 2PI units in the counter clockwise direction and assign a random location to it in a lookup table

4.8 Helpers/wavefront-loader.cpp File Reference

```
#include "wavefront-loader.hpp"
#include <algorithm>
```

Functions

- void load_obj (string filename, vector< float > *vertices, vector< float > *normals, vector< float > *texture_coords)

4.8.1 Function Documentation

4.8.1.1 load_obj()

```
void load_obj (
            string filename,
            vector< float > * vertices,
            vector< float > * normals,
            vector< float > * texture_coords )
```

Function to load the mesh information from a .obj file, it assumes triangular meshes only. All return arrays must be cleared before using the function, else information will be returned at the end of the arrays.

Params: filename: the path to the file to be loaded. vertices: a pointer to a vector of floats where the vertex information will be loaded normals: a pointer to a vector of floats where the normal information will be loaded texture_coords: a pointer to a vector of floats where the texture mapping information will be loaded

4.9 Helpers/wavefront-loader.hpp File Reference

```
#include <fstream>
#include <iostream>
#include <sstream>
#include <vector>
#include <stdlib.h>
#include <string>
```

Functions

- void load_obj (std::string filename, std::vector< float > *vertices, std::vector< float > *normals, std::vector< float > *texture_coords)

4.9.1 Function Documentation

4.9.1.1 load_obj()

```
void load_obj (
            std::string filename,
            std::vector< float > * vertices,
            std::vector< float > * normals,
            std::vector< float > * texture_coords )
```

4.10 main.cpp File Reference

```
#include "system-libraries.hpp"
#include "Window-Management.hpp"
#include "Cube.hpp"
#include "World.hpp"
```

Typedefs

- typedef std::chrono::duration< int, std::ratio< 1, 60 > > frame_duration
- typedef std::chrono::duration< int, std::ratio< 1, 600 > > world_duration

Functions

- void render_loop (GLFWwindow *window)
- void update_loop (GLFWwindow *, GLFWwindow *)
- int main (int argc, char **argv)

4.10.1 Typedef Documentation

4.10.1.1 frame_duration

```
typedef std::chrono::duration<int, std::ratio<1, 60> > frame_duration
```

4.10.1.2 world_duration

```
typedef std::chrono::duration<int, std::ratio<1, 600> > world_duration
```

4.10.2 Function Documentation

4.10.2.1 main()

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

4.10.2.2 render_loop()

```
void render_loop (
             GLFWwindow * window )
```

4.10.2.3 update_loop()

```
void update_loop (
             GLFWwindow * window,
             GLFWwindow * o_window )
```

4.11 Rendering/Camera/Camera.cpp File Reference

```
#include "Camera.hpp"
```

4.12 Rendering/Camera/Camera.hpp File Reference

```
#include <glm/glm.hpp>
#include <glm/gtc/matrix_transform.hpp>
#include <glm/gtx/transform.hpp>
```

Classes

- class Camera

4.13 Rendering/OpenGL-Wrappers.cpp File Reference

```
#include <stb/stb_image.h>
#include <stb/stb_image_write.h>
#include "system-libraries.hpp"
#include "OpenGL-Wrappers.hpp"
```

Macros

- #define STB_IMAGE_IMPLEMENTATION
- #define STB_IMAGE_WRITE_IMPLEMENTATION

Variables

- Renderer * Rendering_Handler

4.13.1 Macro Definition Documentation

4.13.1.1 STB_IMAGE_IMPLEMENTATION

```
#define STB_IMAGE_IMPLEMENTATION
```

4.13.1.2 STB_IMAGE_WRITE_IMPLEMENTATION

```
#define STB_IMAGE_WRITE_IMPLEMENTATION
```

4.13.2 Variable Documentation

4.13.2.1 Rendering_Handler

```
Renderer* Rendering_Handler
```

4.14 Rendering/OpenGL-Wrappers.hpp File Reference

```
#include "system-libraries.hpp"
#include "Camera.hpp"
#include "cout-definitions.hpp"
```

Classes

- class Shader
- class Texture
- struct Mesh
- class Renderer
- class Object_3D

Enumerations

- enum PROGRAM

Functions

- int openGLerror ()

Variables

- Renderer * Rendering_Handler

4.14.1 Enumeration Type Documentation

4.14.1.1 PROGRAM

```
enum PROGRAM
```

4.14.2 Function Documentation

4.14.2.1 openGLerror()

```
int openGLerror ( )
```

4.14.3 Variable Documentation

4.14.3.1 Rendering_Handler

```
Renderer* Rendering_Handler
```

4.15 Rendering/Window-Management.cpp File Reference

```
#include "Window-Management.hpp"
```

Macros

- #define CAM_SPEED 0.3f

Functions

- GLFWwindow * create_context (GLFWwindow *other_window, bool visible)
- int openGLerror ()
- void callBackInit (GLFWwindow *window)
- GLFWwindow * createWindow (GLFWwindow *other_window, bool visible)
- int cursorSelectNode (GLFWwindow *window)
- void error_callback (int error, const char *description)
- void cursor_pos_callback (GLFWwindow *window, double xpos, double ypos)
- void mouse_button_callback (GLFWwindow *window, int button, int action, int mods)
- void key_callback (GLFWwindow *window, int key, int scancode, int action, int mods)

4.15.1 Macro Definition Documentation

4.15.1.1 CAM_SPEED

```
#define CAM_SPEED 0.3f
```

4.15.2 Function Documentation

4.15.2.1 callBackInit()

```
void callBackInit (
            GLFWwindow * window )
```

4.15.2.2 create_context()

```
GLFWwindow* create_context (
            GLFWwindow * other_window,
            bool visible )
```

4.15.2.3 createWindow()

```
GLFWwindow* createWindow (
            GLFWwindow * other_window,
            bool visible )
```

4.15.2.4 cursor_pos_callback()

```
void cursor_pos_callback (
            GLFWwindow * window,
            double xpos,
            double ypos )
```

4.15.2.5 cursorSelectNode()

```
int cursorSelectNode (
            GLFWwindow * window )
```

4.15.2.6 error_callback()

```
void error_callback (
            int error,
            const char * description )
```

4.15.2.7 key_callback()

```
void key_callback (
            GLFWwindow * window,
            int key,
            int scancode,
            int action,
            int mods )
```

4.15.2.8 mouse_button_callback()

```
void mouse_button_callback (
            GLFWwindow * window,
            int button,
            int action,
            int mods )
```

4.15.2.9 openGLerror()

```
int openGLerror ( )
```

4.16 Rendering/Window-Management.hpp File Reference

```
#include "system-libraries.hpp"
#include "OpenGL-Wrappers.hpp"
```

Functions

- void error_callback (int error, const char ∗description)
- void key_callback (GLFWwindow ∗window, int key, int scancode, int action, int mods)
- void mouse_button_callback (GLFWwindow ∗window, int button, int action, int mods)
- void cursor_pos_callback (GLFWwindow ∗window, double xpos, double ypos)
- void callBackInit (GLFWwindow ∗window)
- double calculateFPS (double prevTime, double currentTime)
- GLFWwindow ∗ createWindow (GLFWwindow ∗other_window, bool)
- GLFWwindow ∗ create_context (GLFWwindow ∗other_window, bool)

4.16.1 Function Documentation

4.16.1.1 calculateFPS()

```
double calculateFPS (
            double prevTime,
            double currentTime )
```

4.16.1.2 callBackInit()

```
void callBackInit (
            GLFWwindow * window )
```

4.16.1.3 create_context()

```
GLFWwindow* create_context (
            GLFWwindow * other_window,
            bool  )
```

4.16.1.4 createWindow()

```
GLFWwindow* createWindow (
            GLFWwindow * other_window,
            bool  )
```

4.16.1.5 cursor_pos_callback()

```
void cursor_pos_callback (
            GLFWwindow * window,
            double xpos,
            double ypos )
```

4.16.1.6 error_callback()

```
void error_callback (
            int error,
            const char * description )
```

4.16.1.7 key_callback()

```
void key_callback (
            GLFWwindow * window,
            int key,
            int scancode,
            int action,
            int mods )
```

4.16.1.8 mouse_button_callback()

```
void mouse_button_callback (
            GLFWwindow * window,
            int button,
            int action,
            int mods )
```

4.17 World.cpp File Reference

```
#include "World.hpp"
#include "cout-definitions.hpp"
```

Macros

- #define MESH Cube::meshes[0]

Variables

- World * the_world

4.17.1 Macro Definition Documentation

4.17.1.1 MESH

```
#define MESH Cube::meshes[0]
```

4.17.2 Variable Documentation

4.17.2.1 the_world

```
World* the_world
```

4.18 World.hpp File Reference

```
#include "Cube.hpp"
#include "tools.hpp"
```

Classes

- struct Light
- class Chunk
- class Chunk_Holder
- class World

Macros

- #define CHUNK_DIMS 16

Variables

- World * the_world

4.18.1 Macro Definition Documentation

4.18.1.1 CHUNK_DIMS

```
#define CHUNK_DIMS 16
```

4.18.2 Variable Documentation

4.18.2.1 the_world

```
World* the_world
```

Index