Path tracing

Generated by Doxygen 1.9.1

Chapter 1

Path Tracing

This is the template for the projects. Please copy the project description here. You can use Markdown language to render it as formatted **HTML** file.

1.1 Group

- Aleksi Lassila
- Johannes Karhapää
- Markus Lång
- Weronika Klatka

1.2 Repository organization

Your project implementation should follow the skelaton organization in this repository. See readme.md files in each folder.

1.3 Project Implementation

You must use git repository for the work on the project, making frequent enough commits so that the project group (and course staff) can follow the progress.

The completed project work will be demonstrated to the group's advisor at a demo session. The final demonstrations are arranged on week 50. After the final demonstrations project group evaluates another project, and self-evaluates own project. In addition, project members will give a confidential individual assessment of each group member

The course staff should be able to easily compile the project work using makefile and related instructions provided in the git repository. The final output should be in the **master branch** of the git repository.

1.4 Working practices

Each project group is assigned an advisor from the project teaching personnel. There will be a dedicated Teams channel for each project topic to facilitate discussion between the groups in the same topic and the advisor.

The group should meet weekly. The weekly meeting does not need to be long if there are no special issues to discuss, and can be taken remotely as voice/video chat on the group Teams channel (or Zoom or other similar tool), preferably at a regular weekly time. In the meeting the group updates:

- What each member has done during the week
- Are there challenges or problems? Discuss the possible solutions
- Plan for the next week for everyone
- Deviations and changes to the project plan, if any
- After the meetings, the meeting notes will be committed to the project repository in the `Meeting-notes.↩ md` file.
  - The commits within the week should have some commit messages referring to the meeting notes so that the project advisor can follow the progress.
  - **The meeting notes should be in English.**

  Everyone may not be able to participate to all meetings, but at least a couple of members should be present in each meeting. Regular absence from meetings will affect in individual evaluation.

1.5 Source code documentation

It is strongly recommended to use Doxygen to document your source code. Please go over the *Project Guidelines* for details.

1.6 TODOs (Date)

You can create a list of TODOs in this file. The recommended format is:

- Complete class implementation **foo**. Assigned to <Member 1>
- Test ...

Chapter 2

Source content

This folder should contain only hpp/cpp files of your implementation. You can also place hpp files in a separate directory `include`.

You can create a summary of files here. It might be useful to describe file relations, and brief summary of their content.

Chapter 3

Namespace Index

3.1 Namespace List

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

Chapter 4

Hierarchical Index

4.1 Class Hierarchy

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

Chapter 5

Class Index

5.1 Class List

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

Chapter 6

Namespace Documentation

6.1 FileManager Namespace Reference

Functions

- bool WriteOutput (const std::string &outputPath, const std::vector< std::vector< Colour >> &outputBuffer, unsigned int xDim, unsigned int yDim)
- bool SaveRenderImage (const std::string &outputPath, sf::Image &image)
- auto ColorFromJSON (const json &j, const std::string &name)
- auto VectorFromJSON (const json &j, const std::string &name)
- Camera CameraFromJSON (const json &j)
- auto MaterialFromJSON (const json &j)
- auto SphereFromJSON (const json &j, const Material &mat)
- auto TriangleFromJSON (const json &j, const Material &mat)
- auto ParallelogramFromJSON (const json &j, const Material &mat)
- Scene CreateScene (const std::string &inputPath)

6.1.1 Detailed Description

Namespace for managing files

- Writes output to ppm image file
- Saves rendered SFML image
- Reads scene from json input file

6.1.2 Function Documentation

6.1.2.1 CameraFromJSON()

```
Camera FileManager::CameraFromJSON (
            const json & j )
```

Reads Camera from json format

Parameters

json	reference

Returns

Camera

6.1.2.2 ColorFromJSON()

```
auto FileManager::ColorFromJSON (
             const json & j,
             const std::string & name )
```

Reads sf::Color from json format

Parameters

json	reference
name	of the attribute

Returns

sf::Color

6.1.2.3 CreateScene()

```
Scene FileManager::CreateScene (
             const std::string & inputPath )
```

Creates a scene based on a json file

Parameters

inputPath	

Returns

Scene

6.1.2.4 MaterialFromJSON()

```
auto FileManager::MaterialFromJSON (
            const json & j )
```

Reads Material from json format

Parameters

json	reference

Returns

Material

6.1.2.5 ParallelogramFromJSON()

```
auto FileManager::ParallelogramFromJSON (
            const json & j,
            const Material & mat )
```

Reads Parallelogram from json format

Parameters

json	reference
mat	Material reference

Returns

Parallelogram

6.1.2.6 SaveRenderImage()

```
bool FileManager::SaveRenderImage (
            const std::string & outputPath,
            sf::Image & image )
```

Saves an image using SFML method

Parameters

outputPath	
image	

Returns

bool

6.1.2.7 SphereFromJSON()

```
auto FileManager::SphereFromJSON (
             const json & j,
             const Material & mat )
```

Reads Sphere from json format

Parameters

json	reference
mat	Material reference

Returns

Sphere

6.1.2.8 TriangleFromJSON()

```
auto FileManager::TriangleFromJSON (
             const json & j,
             const Material & mat )
```

Reads Triangle from json format

Parameters

json	reference
mat	Material reference

Returns

Triangle

6.1.2.9 VectorFromJSON()

```
auto FileManager::VectorFromJSON (
             const json & j,
             const std::string & name )
```

Reads Vector from json format

Parameters

json	reference
name	of the attribute

Returns

Vector

6.1.2.10 WriteOutput()

```
bool FileManager::WriteOutput (
            const std::string & outputPath,
            const std::vector< std::vector< Colour >> & outputBuffer,
            unsigned int xDim,
            unsigned int yDim )
```

Writes output buffer to ppm image file

Parameters

outputPath	
outputBuffer	
xDim	
yDim	

Returns

bool

6.2 Random Namespace Reference

Functions

- double GetRandomDoubleUniform (double min, double max, unsigned int seed)

  Random double with uniform distribution.
- double GetRandomDoubleNormal (double stddev, double average, unsigned int seed)

  Random double with normal distribution.

6.2.1 Detailed Description

Random number generation. Contains a set of functions used for ray bounce direction

6.2.2 Function Documentation

6.2.2.1 GetRandomDoubleNormal()

```
double Random::GetRandomDoubleNormal (
            double stddev,
            double average,
            unsigned int seed )
```

Random double with normal distribution.

Returns

std::default_random_engine

6.2.2.2 GetRandomDoubleUniform()

```
double Random::GetRandomDoubleUniform (
            double min,
            double max,
            unsigned int seed )
```

Random double with uniform distribution.

Returns

std::default_random_engine

Chapter 7

Class Documentation

7.1 Camera Class Reference

```
#include <camera.h>
```

Public Member Functions

- **Camera** (Vector position=Vector(0, 0, 0), double viewPlaneDistance=1, double yaw=0, double pitch=0)
- Vector **GetPosition** ()
- double **GetViewPlaneDistance** () const
- double **GetYaw** () const
- double **GetPitch** () const
- double **GetFovDeg** () const
- void **SetViewPlaneDistance** (double viewPlaneDistance)
- void **SetYaw** (double yaw)
- void **SetPitch** (double pitch)
- void **SetFovDeg** (double fov=70)
- void MoveTo (const Vector &position)
- void LookAt (Vector target)
- void **IncrementMoveSpeed** (double amount=DEFAULT_MOVEMENT_INCREMENT)
- void **DecrementMoveSpeed** (double amount=DEFAULT_MOVEMENT_INCREMENT)
- void **IncrementLookSensitivity** (double amount=DEFAULT_LOOK_INCREMENT)
- void **DecrementLookSensitivity** (double amount=DEFAULT_LOOK_INCREMENT)
- Ray CastRay (double xs, double ys)
- void Rotate (double yawAdd, double pitchAdd)
- void **Move** (double x, double y, double z)
- void **MoveForward** ()
- void **MoveBackward** ()
- void **MoveRight** ()
- void **MoveLeft** ()
- void **MoveUp** ()
- void **MoveDown** ()
- void **MoveUpAlongYaxis** ()
- void **MoveDownAlongYaxis** ()
- void **LookRight** ()
- void **LookLeft** ()
- void **LookUp** ()
- void **LookDown** ()

7.1.1 Detailed Description

Class that implements a camera It is essentially a ray with extra properties, functionality and operations related to rendering

7.1.2 Member Function Documentation

7.1.2.1 CastRay()

```
Ray Camera::CastRay (
             double xs,
             double ys )  [inline]
```

Parameters

xs	x coordinate scaled between -1 and 1
ys	y coordinate scaled between -1 and 1

Returns

Ray

7.1.2.2 LookAt()

```
void Camera::LookAt (
             Vector target )  [inline]
```

Sets camera to look in a given direction

Parameters

Vector	target

7.1.2.3 MoveTo()

```
void Camera::MoveTo (
             const Vector & position )  [inline]
```

Sets position to given parameter

Parameters

Vector	position

7.1.2.4 Rotate()

```
void Camera::Rotate (
            double yawAdd,
            double pitchAdd )  [inline]
```

Rotates the camera by given angles

Parameters

yawAdd	
pitchAdd	

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

- src/world/camera.h

7.2 Colour Class Reference

```
#include <colour.hpp>
```

Public Member Functions

- **Colour** (unsigned int red, unsigned int green, unsigned int blue)
- Colour (sf::Color colour)
- unsigned char **GetRed** () const
- unsigned char **GetGreen** () const
- unsigned char **GetBlue** () const

7.2.1 Detailed Description

Simple colour class to hold RGB colour information

7.2.2 Constructor & Destructor Documentation

7.2.2.1 Colour()

```
Colour::Colour (
            sf::Color colour )  [inline], [explicit]
```

Color constructor using sf::Color

Parameters

colour	

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

- src/utils/colour.hpp

7.3 HitInfo Struct Reference

```
#include <pathtracer.hpp>
```

Collaboration diagram for HitInfo:

Public Attributes

- bool **hit**
- Vector **point**
- Vector **sNormal**
- Material **sMaterial**

7.3.1 Detailed Description

Struct that keeps information about a hit point

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

- src/pathtracer.hpp

7.4 Material Class Reference

```
#include <material.hpp>
```

Public Member Functions

- **Material** (sf::Color color=sf::Color(255, 0, 255), double roughness=0.5, double specularIntensity=0.5, sf↩ ::Color specularColour=sf::Color(255, 255, 255), Vector emission=Vector(), double n=1, const std::string &name="default")
- sf::Color **GetColor** () const
- double **GetRoughness** () const
- double **GetSpecularIntensity** () const
- sf::Color **GetSpecularColor** () const
- Vector **GetEmission** () const
- double **GetN** () const
- const std::string & **GetName** () const
- Vector FindSpecularBounceDirection (Ray &ray, Vector &normal, unsigned int randSeed) const

  Find the bounce direction after a ray hits an object based on a normal vector of a surface Ray must reflect within 90 degrees of the normal vector.
- Vector FindDiffuseBounceDirection (Vector &normal, unsigned int randSeed) const

  Find the bounce direction after a ray hits an object based on a normal vector of a surface Ray must reflect within 90 degrees of normal vector.
- Vector FindRefractionDirection (Ray &ray, Vector &normal) const

7.4.1 Detailed Description

Material class to gather material information and handle ray interaction with objects

7.4.2 Member Function Documentation

7.4.2.1 FindDiffuseBounceDirection()

```
Vector Material::FindDiffuseBounceDirection (
            Vector & normal,
            unsigned int randSeed ) const
```

Find the bounce direction after a ray hits an object based on a normal vector of a surface Ray must reflect within 90 degrees of normal vector.

Parameters

ray	
normal	
randSeed	

Returns

Vector

7.4.2.2 FindRefractionDirection()

```
Vector Material::FindRefractionDirection (
            Ray & ray,
            Vector & normal ) const
```

Calculates the odds of reflection and based on those odds plays dice if ray will be reflected. If reflected, returns a zero vector, and new ray direction will be decided by other methods. Otherwise finds refraction direction using Snells'law.

Parameters

ray	
normal	

Returns

Vector

7.4.2.3 FindSpecularBounceDirection()

```
Vector Material::FindSpecularBounceDirection (
             Ray & ray,
             Vector & normal,
             unsigned int randSeed ) const
```

Find the bounce direction after a ray hits an object based on a normal vector of a surface Ray must reflect within 90 degrees of the normal vector.

Parameters

ray	
normal	
randSeed	

Returns

Vector

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

- src/utils/material.hpp
- src/utils/material.cpp

7.5 MaterialBuilder Class Reference

```
#include <materialbuilder.hpp>
```

Public Member Functions

- MaterialBuilder ()
- Material **BuildMaterial** () const
- void **SetColor** (sf::Color color)
- void **SetRoughness** (double roughness)
- void **SetSpecularIntensity** (double specularIntensity)
- void **SetSpecularColor** (sf::Color specularColor)
- void **SetEmission** (const Vector &emission)
- void **SetN** (double n)
- void **SetName** (const std::string &name)

7.5.1 Detailed Description

Builder design pattern used for keeping Material class encapsulated and immutable

7.5.2 Constructor & Destructor Documentation

7.5.2.1 MaterialBuilder()

```
MaterialBuilder::MaterialBuilder ( )  [inline]
```

Constructor with default values

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

- src/utils/materialbuilder.hpp

7.6 object::Object Class Reference

```
#include <objects.hpp>
```

Inheritance diagram for object::Object:

Collaboration diagram for object::Object:

Public Member Functions

- **Object** (Vector origin)
- **Object** (Vector origin, const Material &material)
- Vector **GetOrigin** () const
- Material **GetMaterial** () const
- sf::Color **GetColor** () const
- Vector **GetIntersectionPoint** (const Ray &ray)
- virtual double GetIntersectionDistance (const Ray &ray)=0
- virtual Vector Normal (const Vector &point)=0

Protected Attributes

- Vector **origin_**
- Material **material_**

Friends

- std::ostream & operator<< (std::ostream &os, const Object &obj)

7.6.1 Detailed Description

Abstract class (interface) for objects

7.6.2 Member Function Documentation

7.6.2.1 GetIntersectionDistance()

```
virtual double object::Object::GetIntersectionDistance (
            const Ray & ray )  [pure virtual]
```

Calculated distance between a ray and the object

Parameters

ray	

Returns

double

Implemented in object::Triangle, object::Sphere, and object::Parallelogram.

7.6.2.2 Normal()

```
virtual Vector object::Object::Normal (
            const Vector & point )  [pure virtual]
```

Calculates normal vector at a given point on the object

Parameters

point	

Returns

Vector

Implemented in object::Triangle, object::Sphere, and object::Parallelogram.

7.6.3 Friends And Related Function Documentation

7.6.3.1 operator<<

```
std::ostream& operator<< (
            std::ostream & os,
            const Object & obj )  [friend]
```

Writes to stream

Parameters

stream	
vector	

Returns

std::stream

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

- src/world/objects/objects.hpp
- src/world/objects/objects.cpp

7.7 object::Parallelogram Class Reference

```
#include <parallelogram.hpp>
```

Inheritance diagram for object::Parallelogram:

Collaboration diagram for object::Parallelogram:

Public Member Functions

- **Parallelogram** (const Vector &origin, const Vector &a, const Vector &b, const Material &material)
- double GetIntersectionDistance (const Ray &ray) override
- Vector Normal (const Vector &point) override

Additional Inherited Members

7.7.1 Detailed Description

Parallelogram class that inherits from Object

7.7.2 Member Function Documentation

7.7.2.1 GetIntersectionDistance()

```
double object::Parallelogram::GetIntersectionDistance (
            const Ray & ray )  [override], [virtual]
```

Calculates intersection of ray and parallelogram if it exists

Implements object::Object.

7.7.2.2 Normal()

```
Vector object::Parallelogram::Normal (
             const Vector & point )  [override], [virtual]
```

Calculates normal vector of parallelogram at point given by vector point

Implements object::Object.

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

- src/world/objects/parallelogram.hpp
- src/world/objects/parallelogram.cpp

7.8 PathTracer Class Reference

```
#include <pathtracer.hpp>
```

Public Member Functions

- **PathTracer** (sf::Vector2u &dimensions, Scene &scene)
- sf::Image **GetLatestImage** ()
- void **UpdateRenderContext** (sf::Vector2u &dimensions, Scene &scene)
- void Renderer (sf::RenderWindow &window)

Static Public Member Functions

- static void **Draw** (sf::RenderWindow &window, sf::Image &image)

7.8.1 Detailed Description

The core class of this application which defines the rendering logic

7.8.2 Member Function Documentation

7.8.2.1 Renderer()

```
void PathTracer::Renderer (
             sf::RenderWindow & window )  [inline]
```

This function runs inside a thread and continuously renders an image based on the rendering context.

Parameters

window	

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

- src/pathtracer.hpp

7.9 Ray Class Reference

```
#include <ray.hpp>
```

Collaboration diagram for Ray:

7.10 RenderContext Struct Reference

```
#include <pathtracer.hpp>
```

Public Member Functions

- **RenderContext** (sf::Vector2u &dimensions, Scene &scene)
- **RenderContext** (sf::Vector2u &dimensions, Scene &scene, int maxBounces)
- sf::Color **GetPixel** (sf::Vector2u pos)
- void **SetPixel** (sf::Vector2u pos, sf::Color color)
- sf::Image **GetImage** ()
- Scene **GetScene** ()
- sf::Vector2u **GetDimensions** ()
- sf::Color **AverageColor** (sf::Vector2u pos, sf::Color color)

Public Attributes

- std::atomic< int > **frameCount** = 0
- std::atomic< int > **maxBounces** = 12
- std::atomic< bool > **cancelled** = false

7.10.1 Detailed Description

This struct is a thread-safe way of representing the data needed to render a frame.

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

- src/pathtracer.hpp

7.11 Scene Class Reference

```
#include <scene.h>
```

Public Member Functions

- **Scene** (Camera &camera, std::initializer_list< std::shared_ptr< object::Object >> objects)
- Camera & **GetCamera** ()
- std::vector< std::shared_ptr< object::Object > > **GetObjects** () const
- Vector **GetAmbientLightDirection** () const
- void **SetAmbientLightDirection** (Vector direction)
- void **AddObject** (std::shared_ptr< object::Object > object)
- void **RemoveObject** (std::shared_ptr< object::Object > object)

Friends

- std::ostream & **operator**<< (std::ostream &os, const Scene &scene)

7.11.1 Detailed Description

Class for implementing a scene It bundles the camera and objects together

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

- src/world/scene.h
- src/world/scene.cpp

7.12 object::Sphere Class Reference

```
#include <sphere.hpp>
```

Inheritance diagram for object::Sphere:

Collaboration diagram for object::Sphere:

Public Member Functions

- **Sphere** (const Vector &origin, double radius)
- **Sphere** (const Vector &origin, double radius, const Material &material)
- double GetIntersectionDistance (const Ray &ray) override
- Vector Normal (const Vector &point) override

Additional Inherited Members

7.12.1 Detailed Description

Sphere class that inherits from Object

7.12.2 Member Function Documentation

7.12.2.1 GetIntersectionDistance()

```
double object::Sphere::GetIntersectionDistance (
             const Ray & ray )  [override], [virtual]
```

Calculates intersection of ray and sphere if it exists

Implements object::Object.

7.12.2.2 Normal()

```
Vector object::Sphere::Normal (
             const Vector & point )  [override], [virtual]
```

Calculates normal vector of sphere at point given by vector point

Implements object::Object.

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

- src/world/objects/sphere.hpp
- src/world/objects/sphere.cpp

7.13 object::Triangle Class Reference

```
#include <triangle.hpp>
```

Inheritance diagram for object::Triangle:

Collaboration diagram for object::Triangle:

Public Member Functions

- **Triangle** (const Vector &origin, const Vector &a, const Vector &b, const Material &material)
- double GetIntersectionDistance (const Ray &ray) override
- Vector Normal (const Vector &point) override

Additional Inherited Members

7.13.1 Detailed Description

Triangle class that inherits from Object

7.13.2 Member Function Documentation

7.13.2.1 GetIntersectionDistance()

```
double object::Triangle::GetIntersectionDistance (
            const Ray & ray )  [override], [virtual]
```

Calculates intersection of ray and triangle if it exists

Implements object::Object.

7.13.2.2 Normal()

```
Vector object::Triangle::Normal (
            const Vector & point )  [override], [virtual]
```

Calculates normal vector of triangle at point given by vector point

Implements object::Object.

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

- src/world/objects/triangle.hpp
- src/world/objects/triangle.cpp

7.14 Vector Class Reference

```
#include <vector.hpp>
```

Public Member Functions

- **Vector** (double x=0, double y=0, double z=0)
- double **x** () const
- double **y** () const
- double **z** () const
- Vector operator+ (const Vector &b)
- Vector operator- (const Vector &b)
- Vector operator+= (const Vector &b)
- Vector operator* (double k)
- double Len ()
- Vector Norm ()
- double operator* (const Vector &v2)
- Vector operator% (const Vector &v2)
- bool operator== (const Vector &v2)
- bool operator!= (const Vector &v2)
- double Dot (const Vector &v2)
- Vector CrossProduct (const Vector &v2)
- double Distance (const Vector &v2)

Friends

- std::ostream & operator<< (std::ostream &os, const Vector &v)

7.14.1 Detailed Description

Our own implementation of a vector class

7.14.2 Member Function Documentation

7.14.2.1 CrossProduct()

```
Vector Vector::CrossProduct (
             const Vector & v2 )  [inline]
```

Cross Product using named function

Parameters

vector	

Returns

Vector

7.14.2.2 Distance()

```
double Vector::Distance (
             const Vector & v2 )  [inline]
```

Calculates distance between vectors

Parameters

vector	

Returns

double

7.14.2.3 Dot()

```
double Vector::Dot (
             const Vector & v2 )  [inline]
```

Dot Product using named function

Parameters

vector	

Returns

double

7.14.2.4 Len()

```
double Vector::Len ( )  [inline]
```

Calculates length of a vector

Returns

double

7.14.2.5 Norm()

```
Vector Vector::Norm ( )  [inline]
```

Calculates normal vector

Returns

Vector

7.14.2.6 operator"!=()

```
bool Vector::operator!= (
             const Vector & v2 )  [inline]
```

Compares if vectors are NOT equal

Parameters

vector	

Returns

bool

7.14.2.7 operator%()

```
Vector Vector::operator% (
            const Vector & v2 )  [inline]
```

Cross Product using operator %

Parameters

vector	

Returns

Vector

7.14.2.8 operator∗() [1/2]

```
double Vector::operator∗ (
            const Vector & v2 )  [inline]
```

Dot Product using operator ∗

Parameters

vector	

Returns

double

7.14.2.9 operator∗() [2/2]

```
Vector Vector::operator∗ (
            double k )  [inline]
```

Vector multiplication by scalar

Parameters

scalar	

Returns

Vector

7.14.2.10 operator+()

```
Vector Vector::operator+ (
            const Vector & b )  [inline]
```

Vector addition

Parameters

vector	

Returns

Vector

7.14.2.11 operator+=()

```
Vector Vector::operator+= (
            const Vector & b )  [inline]
```

Vector addition assignment operation

Parameters

vector	

Returns

Vector

7.14.2.12 operator-()

```
Vector Vector::operator- (
            const Vector & b )  [inline]
```

Vector subtraction

Parameters

vector	

Returns

Vector

7.14.2.13 operator==()

```
bool Vector::operator== (
             const Vector & v2 )  [inline]
```

Compares if vectors are equal

Parameters

vector	

Returns

bool

7.14.3 Friends And Related Function Documentation

7.14.3.1 operator<<

```
std::ostream& operator<< (
             std::ostream & os,
             const Vector & v )  [friend]
```

Writes to stream

Parameters

stream	
vector	

Returns

std::stream

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

- src/utils/vector.hpp