Cinder

SparkyPotato

Apr 20, 2022

GETTING STARTED

Cinder is a modular raytracer that aims to provide true-to-life, physically based rendering. This documentation is a guide for using Cinder to render images, and is also a guide on how to extend the raytracer to add your own features.

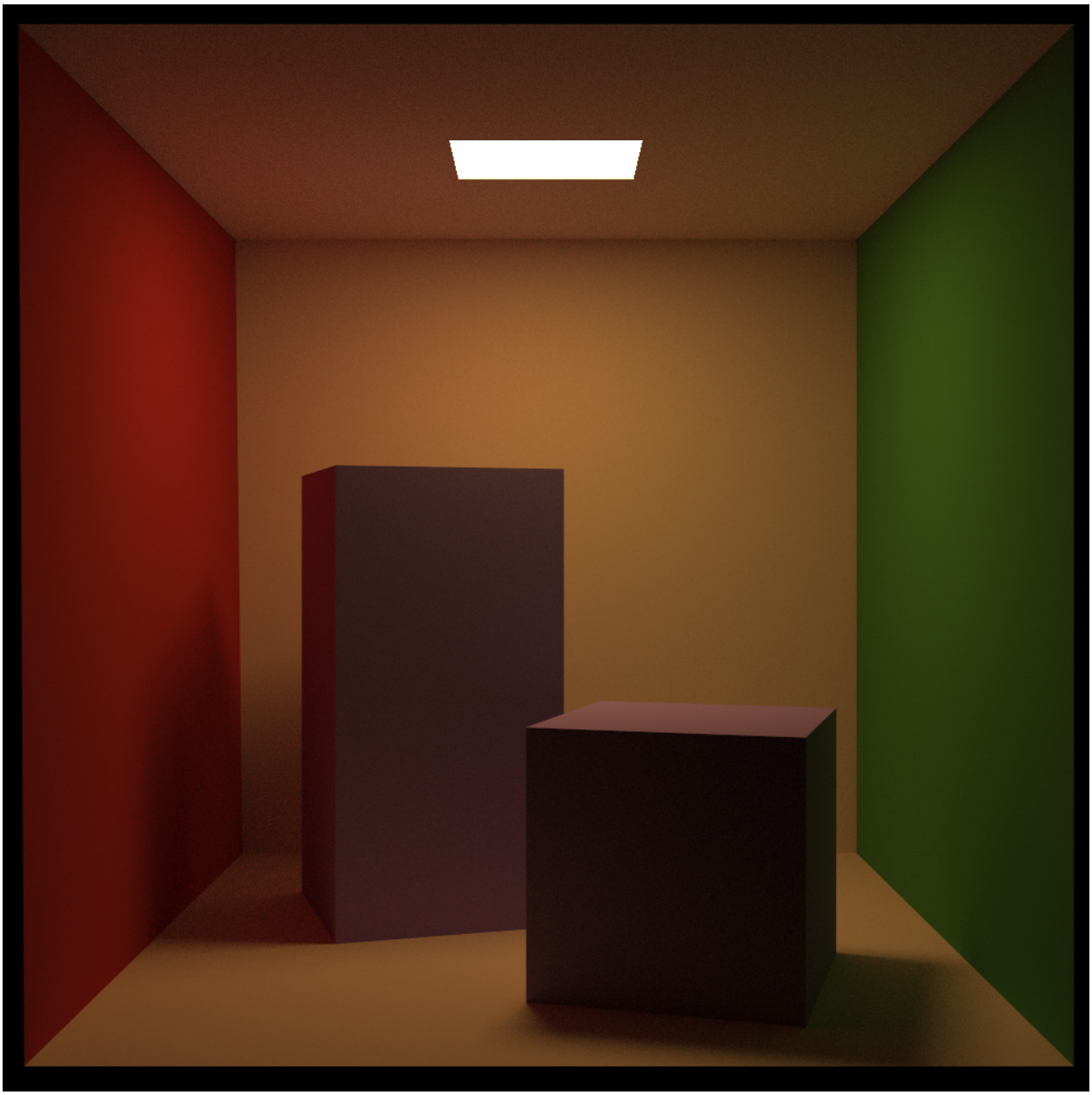

Fig. 1: Cornell Box rendered with path tracer, at a resolution of 1200x1200, with 128 samples per pixel.

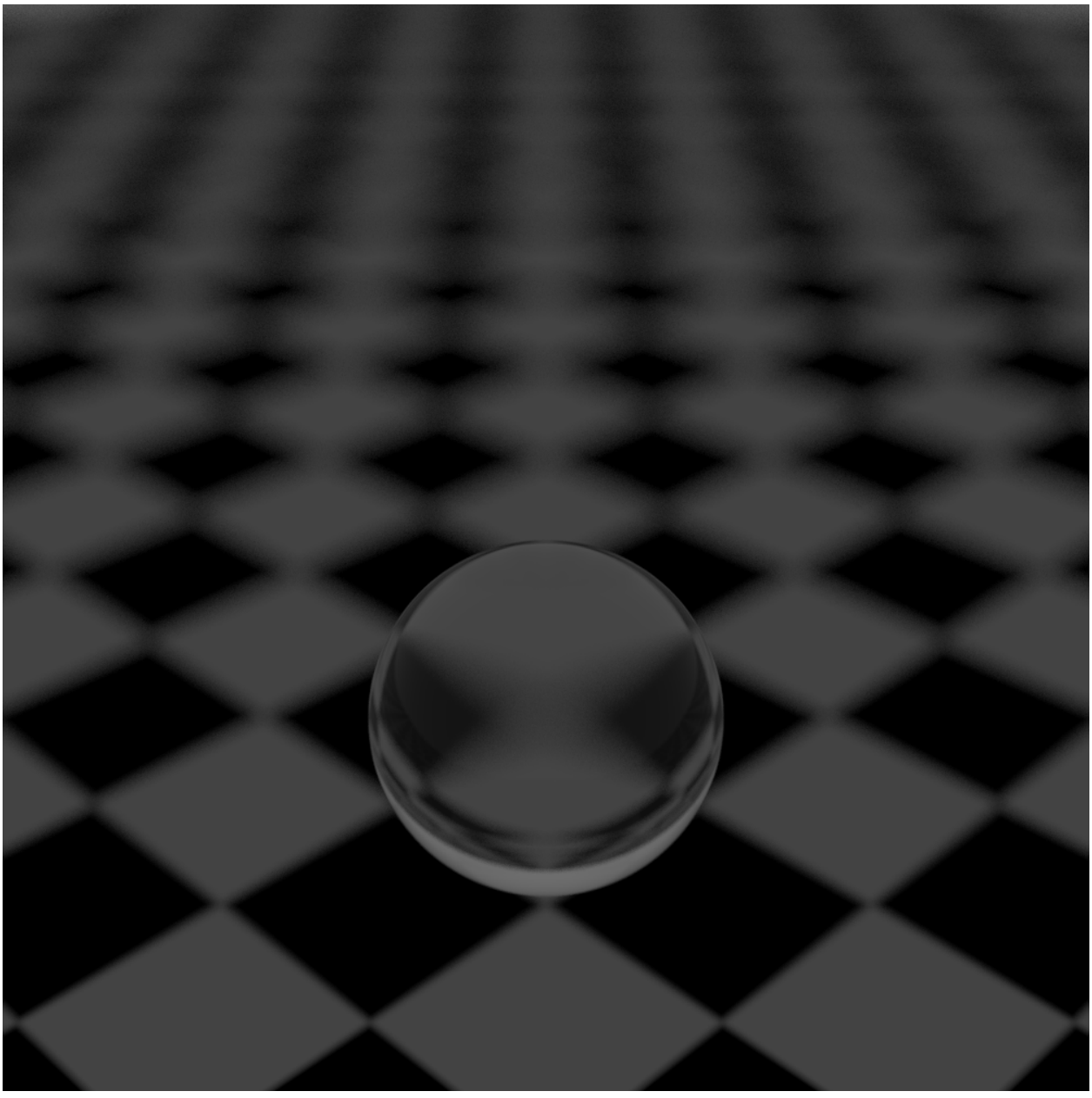

Fig. 2: Depth of field rendered with Whitted tracer, at a resolution of 1200x1200, with 128 samples per pixel.

CHAPTER

ONE

SETTING UP CINDER

1.1 Dependencies

Compiling Cinder requires the following dependencies to be installed:

- *A suitable C++ toolchain*
- *CMake*
- *Ninja*
- *Python*
- *Doxygen*
- *Sphinx*

Doxygen and Sphinx are only required for building the documentation (what you're seeing now). See *Building*.

1.1.1 Getting a C++ toolchain

1.1.1.1 On Linux

You should already have GCC pre-installed.

1.1.1.2 On Mac

Open a Terminal instance by searching for it in Spotlight. Then enter the following command:

```
xcode-select --install
```

1.1.1.3 On Windows

1. Download Visual Studio 2019 Community from here.
2. In the installer, make sure to select the 'Desktop Development with C++' component.

1.1.2 Getting CMake

Download and install the correct binary distribution of CMake for your platform from here.

1.1.3 Getting Ninja

Download and install Ninja from here.

1.1.4 Getting Python

Download Python from here.

1.1.5 Getting Doxygen

Doxygen is not required if you do not want to build the documentation. Download the right version of Doxygen from here.

1.1.6 Getting Sphinx

Sphinx is not required if you do not want to build the documentation.

1. Open a terminal window. On Mac, search it in Spotlight, and on Windows, search for 'Command Prompt'.
2. Enter the following command (requires Python):

```
pip install -U Sphinx
```

1.2 Cloning

If you have Git simply call:

```
git clone https://github.com/SparkyPotato/Cinder --recursive
```

Or else, you can head over to GitHub, and download the repository by clicking on 'Code' and then 'Download ZIP'.

1.3 Building

Note: If you do not wish to build the documentation, you do not need to install Doxygen or Sphinx. To disable documentation building, add the option `-nodoc` while invoking `Build.py`.

1. Open another terminal window, and navigate into the `Cinder` directory you just cloned. On Windows, you might need to open a Developer Command Prompt. Just search for this in the Start Menu.
2. To quickly get up and running, just call:

```
python3 Build.py
```

If you want to know all the options you can set in the build script, call:

```
python3 Build.py -h
```

Note: If an instance of Visual Studio is opened, CMake generation could fail. Just keep running the build script until it works, or close Visual Studio.

CHAPTER

TWO

YOUR FIRST RENDER WITH CINDER

Cinder uses an easy to understand and write YAML format for project and scene serialization. You can go through all the scenes used in this documentation in the `Cinder/Projects` folder.

We will be rendering the Cornell Box in this introduction to the Cinder CLI Tool.

2.1 Cinder Projects and Scenes

To render something with Cinder, you need to have two things: a project, and a scene. The two are closely related, but both have different uses.

2.1.1 Projects

A project is used to setup *how* you want Cinder to render scenes. It decides what type of renderer to use, and what kind of images to output. It also contains a list of scenes to be rendered, and the output image of each of them.

2.1.2 Scenes

A scene is exactly what the name suggests. It consists of a camera, objects, their consisting geometry, materials, and lights.

2.2 Rendering the Cornell Box

The Cornell Box is a famous test which determines the accuracy of rendering software, as it is a real place, and has all the relevant data captured. You can check the data out at Cornell.

Rendering the Cornell Box with Cinder is a fairly simple task.

Our first step is to find the Cinder binaries. If you used the Python build script, you should find the Binaries in `Build/{Config}/Binaries`, where `{Config}` is the configuration you chose while building (this is Release by default). If you built it directly with CMake, they will be in `{BuildDir}/Binaries`, where `{BuildDir}` is the CMake output directory.

> **Warning:** If you're on Windows and decide to move the Cinder executable outside the directory, ensure that you also move `CinderAPI.dll` along with it, and keep them both in the same folder. On Linux, ensure you do not delete or move `libCinder.so` out of `Build/{Config}/Libraries`

Assuming the binaries are in their default directory, you just need to run the following command in a terminal to begin rendering. Ensure that the terminal's working directory is the root of the Cinder project.

```
Build/Release/Binaries/cinder Projects/CornellBox/Project.yaml
```

Note: You will have to replace all the path separators ('/') with '\' on Windows.

After the render is complete, you should see a file `CornellBox.png` in `Projects/CornellBox/`.