UNIVERSITÄT DES SAARLANDES
Prof. Dr.-Ing. Philipp Slusallek
Computer Graphics Group
Arsène Pérard-Gayot (perard@cg.uni-saarland.de)

16. December 2019

Introduction to Computer Graphics
Rendering Competition

Submission deadline for the Rendering Competition: 03. February 2020, 23:59 CET
Throughout the semester you have been working hard on your ray tracer. The Rendering Competition is the final step. Its goal is to show off how good your ray tracing engine is, and how you can use it to create stunning images. Your task is to:

a) Create an original scene.

b) Use your ray tracer to create an image.

c) Create a web page to describe the image and highlight the engine features.

You can either submit an entry per team (akin to the assignments), or one entry per person.

1 The Scene

Your scene should be original. You can get *inspiration* from past rendering competitions at other institutions, or from the various ray-traced images available on the web, but the final scene should be the product of your own imagination.
You are allowed to use third-party assets, such as models or textures (e.g. from Google 3D Warehouse[1], Blend Swap[2], etc.). Those assets must be publicly available for free. You can use those assets to build your original scene, but it is not allowed for the whole, or major part of the scene to be straight reused from somewhere. Alternatively, you can model everything yourself, e.g. using Blender (free for all) or 3DS Max (free for students at UdS).
All assets used in the scene must be included with the submission.

2 The Rendering

You must submit two PNG images: a low quality at around 480x270 and high quality with resolution 1920x1080 or higher. You may use different aspect ratio if it fits better your scene.
The image must be rendered by your own ray tracer. Any post-process operation you may want to apply should be implemented within your framework and not performed via some third-party image-editing software or library.
You may implement additional features in your ray tracer if your scene needs it, but it is not required. The grading will be based on the artistic merit of the image, and not the amount of additional code you have put into the ray tracer. On the other hand, of course, those additional features — if used right — can make your image more interesting.
The image does not have to be *realistic*. You can use features that do not follow the real life physics if your artistic concept demands it.
The final image must render roughly in less than 12 hours on a modern computer. It is recommended to implement binning BVH building for good performance (see the slide set "Building Good BVHs").

[1] https://3dwarehouse.sketchup.com/
[2] https://www.blendswap.com/

3 The Website

On the website you should advertise your image (why should it win the competition?) and describe your work:

- Summarize your concept. How you arrived to it.
- Shortly describe how you were building up your scene.
- Highlight your image. Which are more interesting parts or features? You may include additional images on your page.
- List all additional features you have implemented.
  - Say where the feature can be visible on your image
  - Who implemented it (if in group)
  - Where in the source code it can be found
- Give references to all third-party material.
- Any other comments you would like to include.

In addition you must include:

- Names of all participants
- Title of your scene

Keep in mind that this website will be made public on the university website and that you should be careful not to include sensitive or offensive content.

4 Submission

- Submit your entry for the rendering competition via e-mail to the address at the top of this sheet.
- The subject of the message should be of the form
  - For single person: `RC1920 MatrNum LastName`
  - For a team: `RC1920 MatrNum1 MatrNum2 groupID`
- Avoid special characters like ä ü ó ß.
- Submit two files:
  - The source code of your ray tracer together with all the assets needed for rendering: `LastName_rt.zip` or `groupID_rt.zip`
  - The web page: `LastName_web.zip` or `groupID_web.zip`
- The ray tracer code should follow a similar structure as the framework that we provided. It must build out-of-the-box using CMake without any third-party dependencies (except libpng). You may add an `assignment-rc.cmake` with a similar structure as the others in order to add further sources.
- The main program, when invoked without parameters, should render the high-quality image. No additional configuration, or downloading extra files should be required.
- Do not include executables or object files in your submission — we will compile your code.
- The web page should contain at least three files: `final.png`, `thumbnail.png`, and `index.html`.

If the files are too big to send via mail, you may put them at some external location and send us a link. Alternatively, you can hand it in on some physical medium (CDs, USB drives) — in which case compression is not necessary, but the file structure should be the same.

5 Bonus: BVH Speed Contest

This year, we introduce a BVH speed contest. This is entirely optional, and gives bonus points. Because we need to evaluate the performance of your BVH on the same machine, we provide new files that contain the skeleton of a BVH exporter in a zip archive. Be careful, as this archive contains different versions of `bvh.cpp` and `bvh.h`, so **make a copy of your BVH implementation before extracting it**.
To take part in the contest, implement the exporter in `BVH::serialize()`, and send the code along with your rendering competition submission. We will run your BVH builder on a scene of our choice (that we intentionally do not provide, featuring around 700k polygons), then export the result to a file and load it in *our* renderer to do the benchmarking. There is thus no need to optimize your shading code or intersection routines: Only the BVH structure matters. We will reward the best, second best, and third best BVH builders with bonus points equivalent to 20%, 10%, and 5% of the rendering competition points, respectively. Renderers taking more than 15mins or 16Gbytes of RAM to terminate on the tested scene will be disqualified.
To debug and test your BVH builder/exporter performance and correctness, we provide the benchmarking utility `cgray-bench`. This tool requires you to implement `BVH::deserialize()` as well, so that it can load BVHs from a file. In order to use it, simply type: `cgray-bench cow.obj` (replace `cow.obj` with the model of your choice). The tool will try to load a BVH file named `bvh.bin`, and if that file cannot be found, will build a BVH and export it in `bvh.bin`. If you change the input scene file, make sure to delete the BVH file before re-running the tool.