Department: Education
Editors: Beatriz Souza Santos, bss@ua.pt & Ginger Alford, alfordg@smu.edu

Remote Teaching Advanced Rendering Topics using the Rayground Platform

A. A. Vasilakis
Department of Computer Science and Engineering, University of Ioannina, Greece
Department of Informatics, Athens University of Economics and Business, Greece

G. Papaioannou
Department of Informatics, Athens University of Economics and Business, Greece

N. Vitsas
Department of Informatics, Athens University of Economics and Business, Greece

A. Gkaravelis
Department of Informatics, Athens University of Economics and Business, Greece

***Abstract*—Rayground is a novel online framework for fast prototyping and interactive demonstration of ray tracing algorithms. It aims to set the ground for the online development of ray-traced visualization algorithms in an accessible manner for everyone, stripping off the mechanics that get in the way of creativity and the understanding of the core concepts. Due to the COVID-19 pandemic, remote teaching and online coursework have taken center stage. In this work, we demonstrate how Rayground can incorporate advanced instructive rendering media during online lectures as well as offer attractive student assignments in an engaging, hands-on manner. We cover things to consider when building or porting methods to this new development platform, best practices in remote teaching and learning activities and time-tested assessment and grading strategies suitable for fully online university courses.**

IN THIS WORK, we share our experience on the transformation of an advanced computer graphics course with extensive hands-on coursework to an online course, with the help of the Rayground platform. "Advanced Computer Graphics" is a new elective course of the undergraduate syllabus (6 ECTS) at the Department of Computer Science and Engineering of Ioannina University in Greece that had to be offered online because of the COVID-19 pandemic. The course was attended by approximately 50 forth-year, full-time, undergraduate students in the winter semester of academic year 2020-2021. The course mainly covers 3D graphics techniques for realistic image

 Published by the IEEE Computer Society

synthesis and assumes that students are familiar with advanced programming skills as well as basic computer graphics knowledge, where modern OpenGL and GLSL shaders knowledge is adequately covered.

It aims to examine state-of-the-art CGI pipelines and established algorithms alongside proven computer graphics software systems (e.g. Blender). The course addresses concepts such as local and global illumination and related methods, real-time visual effects and ray-tracing-based methods for offline and real-time image synthesis.

Our goal was to abstract the programming environment and diverse hardware requirements by structuring the entire course around Rayground, a Web-based educational programming environment for prototyping ray tracing methods. By building educational material on Rayground, students can intuitively learn and easily prototype light transport algorithms via GPU-accelerated ray tracing, with relatively inexpensive hardware, such as laptops, tablets, and even smartphones.

Ray tracing is an important topic in computer graphics but, due to its complexity and lack of tools to demonstrate its functionality, it is discussed briefly during the last couple of lectures in the major undergraduate course “Introduction to Computer Graphics”, as is the common practice in many computer graphics courses [1]. As ray tracing establishes itself both in production and real-time rendering as a viable and more accurate replacement for approximate methods, in this follow-up course we have elevated its importance. Moreover, following the rendering pipeline abstraction model of the computer graphics course offered by the department of Informatics at the Athens University of Economics and Business, we divided the rendering pipeline into four generic stages: a) geometry setup, b) sampling/token generation, c) shading and d) compositing, with a specific note on the re-entrant nature of any of these stages. This allowed us to independently map the rasterization or ray tracing paradigms to this pipeline and clearly separate common topics, such as material properties, local shading, geometry representation, texturing, image-domain sampling and antialiazing. Rayground further assisted in this abstraction process by offering an accessible alternative platform for in-class algorithms and principles demonstration,

Figure 1. The Rayground interface, with the preview window (left) and the shader editor (right).

while providing a simpler solution for the practical and accurate experimentation with certain concepts, such as visibility testing and materials.

Rayground: Online ray tracing platform

Rayground, developed by the Computer Graphics group of the Athens University of Economics and Business, is an interactive education tool for richer in-class teaching and gradual self-study that provides a convenient introduction into ray tracing programming [2]. It was built using WebRays, a WebGL-based ray intersection library for the web [3]. Rayground abstracts the functionality of the underlying ray tracing stages to an extent that still preserves the main concepts taught in a computer graphics course as well as eases the development of advanced visual effects in student projects. It has not been designed to be a complete replacement of teaching computer graphics with modern shader-based programming, e.g. OpenGL/WebGL [4], [5], but rather as a complementary educational resource that harmoniously enriches the teaching environment. Rayground, hosted at https://rayground.com, does not rely on any browser plugins and thus runs on any platform that has a modern, standards-compliant browser. The graphical user interface of Rayground is designed to have two discrete parts, the preview window and the shader editor (**Figure 1**). Visual feedback is interactively provided in the WebGL rendering context of the preview canvas, while the user performs live source code modifications. Rayground is open, cross-platform, and available to everyone.

With a minimal registration (personal email), students have full access to the platform’s functionality. Students can create any number of new projects from scratch or clone an existing one

from a variety of public ray tracing projects available on Rayground. This is also helpful for teachers who want to provide a minimal base project to their students, which the complete assignment will be built on. To ensure academic integrity, Rayground gives the ability to each student to hide their projects during the assignment period in order to avoid their work being plagiarized from their fellow students. Finally, source code can easily be exported from the platform and be submitted as per assignment instructions.

Teaching and Learning Activities

Remote Teaching

Rayground has been successfully used to interactively demonstrate the concepts of advanced image synthesis during 3-hour weekly lectures (12 in total), enriching and complementing the theory presented in static slides. Students were encouraged to actively experiment on their own computers at certain periods during remote teaching, further transforming the learning process into a more immersive and engaging one. As reported by the students themselves (see evaluation below), the learning experience was highly enhanced, since most of them successfully correlated the presented concepts with their practical implementation and results. Students clarified with ease the recursive nature of ray tracing and how rays are generated and scattered differently though the virtual environment, based on a variety of object materials. Furthermore, the relationship between sampling and noise produced in the image is effectively explained by previewing the effect, when dynamically tuning a specific parameter inside a programmable stage (**Figure 2**).

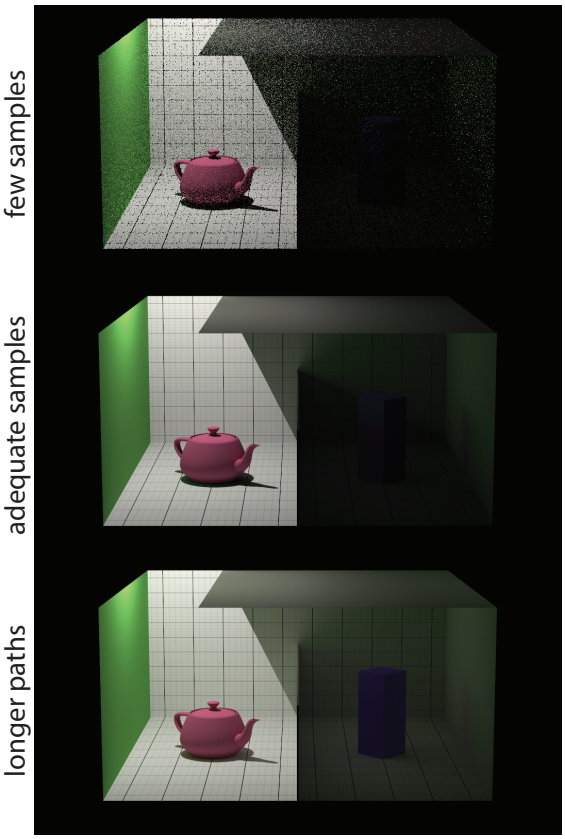

Figure 2. Interactive remote learning example: The random sampling in path tracing causes noise to appear in the rendered image (top). The noise is removed by letting the algorithm generate more samples (middle). Increasing the ray depth results in a dramatic visual difference (bottom).

Remote Labs

As is very frequently the case, the computer graphics lectures were supported by a series of practical exercises in a one- to two-hour lab per week (6 in total), where students were able to remotely apply the theory learned and experiment individually using simple Rayground projects. The contents of the lab sessions were normally aligned with the corresponding week's lectures as described above. Based on the number of registrations, one or two teaching assistants had to be deployed for this matter. Sessions started with an introduction to the topics that would be discussed, followed by a practice exercise for the remainder of the course. The lab assistant was present during the exercise, guiding each student, when necessary, and presenting an indicative solution at the end. This project series was uniquely designed with the aim to help students gradually understand how a ray tracer works in detail, following a modern ray tracing programming model.

Resources

To encourage more teachers to incorporate Rayground into their courses, the teaching material built during this course to support remote learning as well as to promote autodidactic engagement and collaboration are provided freely at the educational section of the Rayground Github account: https://github.com/cgaueb/rayground/tree/master/education.

Assessment and Grading Methods

Assessment Strategy

Student assessment in this particular course is not done via written examinations, which tend to be problematic in remote courses, but rather relies solely on the students' final assignment. The Rayground assignment was therefore mandatory and students were given a two-month period to submit their work in teams of up to two members. The project was assigned after midterm, when teaching of the fundamental concepts about geometry modeling and image synthesis had been sufficiently covered. The students were expected to provide:

- A written report that contained detailed description about their implementation (using pseudocode, reference to sources, mathematical formulations, etc.), figures or snapshots from specific viewpoints, that prove the fulfillment of each requested task and especially for any particularities, problems, special conditions, test data, or additional information they have used during their experimental evaluation.
- The complete source code written for Rayground, associated with the solution to each requested task. This could be automatically exported from the online platform.

The theme of this year's assignment was to create an image of an indoor billiard room via path tracing, where the practical design considerations and content creation for the 3D scene were left to the judgment, imagination and creativity of the students. **Figure 3** illustrates the high-quality synthetic image generated from the best student work that excelled in both visual fidelity and content creation creativity.

The basic ray tracer features were designed with an increasing implementation difficulty in mind, and students were allowed to approach them in an independent and non-sequential manner. A complete progressive path tracer had to gradually be implemented from scratch. The assignment was divided in five core tasks in order to avoid confusion and simplify time management. Each task contained several related problems to be solved. The points corresponding to each task were not uniformly weighted (shown in brackets). Bonus points (highlighted with *italics*) were also awarded for certain tasks, enabling students to earn more than the maximum possible points for the assignment.

Figure 3. Rendered result from the best student Rayground project, excelling in both creativity and technical implementation.

- Background (1) : flat color, gradient color, (*animated*) patterns via procedural texturing.
- Camera (2): orthographic & perspective primary rays, antializing. *depth of field*.
- Modeling (2): main room with light sources, billiard table and indoor details, wall openings (via *ray-traced constructive solid geometry*).
- Local Shading (2): Lambertian, Phong, Blinn-Phong & *Cook-Torrance* illumination models, hard and *soft* shadows from spot and *area* lights, multiple light sources.
- Global Illumination (3): merging all the above in a Whitted and/or Stochastic Path Tracer. *Next-event estimation, multiple importance sampling and Russian roulette termination*.

Examination

At the end of the semester, an oral examination took place for the students, whose project's score was at least 5/10 points to prevent and detect any form of plagiarism and thus ensure fairness. Members of each group were examined separately for 10 minutes each, with the aim of (a) highlighting the ability to apply the knowledge acquired during the learning process and (b) demonstrating the appropriate skill and competence in the use of the new paradigm for solving computer graphics problems. Moreover, group performance had to be translated into individual

grades. If a task was not adequately answered from a student during the oral examination, the corresponding points were not counted towards the student's total assignment grade.

Excluding a couple of cases (see below), project grading went beyond our expectations. Students were able to successfully match previously learned concepts to the new paradigm. The majority (75%) met most of the project goals/requirements, providing high-quality work and showing fairly independent effort during the project. Additionally, a small portion of them (15%) showed evidence of excellent work above normal standards (see Figure 3).

Concerns

We have noted cases where students did not adopt a learning and creative stance, but rather only sought to secure the marginal requirements for each task at hand. To this effect, we observed that a couple of student teams relied heavily on code scavenged from internet resources, without proper information filtering and rationalization. In cases, they resorted to partial code replication from other students' Rayground projects.

Evaluation

Students were given a questionnaire to fill in related to the usefulness and practicality of Rayground as an interactive tool for (remotely) teaching computer graphics and more specifically, ray tracing. From the 40 students who actively participated and contributed to the evaluation, more than 75% commented on the very positive effect it had on their understanding of difficult concepts in theoretical lectures and labs, praising the use of interactive tools like Rayground in the online course. On the other hand, many students mentioned that the provided API documentation was too difficult to understand and work with (25%), taking long to learn and thus adversely affecting the completion of the programming assignment. We attribute this learning difficulty to the poor understanding of practical computer graphics concepts and most importantly, (parallel) programming via GLSL shaders, skills adequately covered in the preceding introductory Computer Graphics course. For completeness, we will offer additional online tutorials and introductory material, coupled with fundamental ray tracing theory, to ease the learning curve for beginner graphics developers, newcomers and graphics enthusiasts.

Conclusion

Although building knowledge around teaching of fundamental visualization concepts is challenging because of the open and broad nature of visualization and its wide application to different domains and audiences, we believe that the Rayground platform has the potential to function as an online hub for the whole community, as ray tracing ideas and methods become more widespread. To this end, we are eager to improve the core technology (e.g. WebGPU implementation) as well as to provide additional features (such as global textures and 3D scenes loading support, social/community features like blogs, comments, tags, etc.) that would enhance usability and user experience for students and teachers alike. We welcome instructors that want to take advantage of this online teaching platform, adopt the accompanied learning materials and resources, and get inspired by our assessment strategies to make 3D visualization knowledge and practical skills available to a wider audience.

Acknowledgment

Many thanks to the editors for their constructive comments and all the students for their honest commitment, effort and evaluation. This work was funded by the Epic MegaGrants grant program from Epic Games Inc.

REFERENCES

1. D. G. Balreira, M. Walter, D, W. Fellner, "What we are teaching in introduction to computer graphics", *Eurographics 2017 - Educational Papers*, April 2017.
2. N. Vitsas, A. Gkaravelis, A. A. Vasilakis, K. Vardis, G. Papaioannou, "Rayground: An online educational tool for ray tracing", *Eurographics 2020 - Educational Papers*, May 2020.
3. N. Vitsas, A. Gkaravelis, A. A. Vasilakis, G. Papaioannou, "WebRays: Ray tracing on the web", *Ray Tracing Gems II, ch. 18*, August 2021.
4. Ed. Angel, "The case for teaching computer graphics with WebGL: A 25-year perspective", *IEEE Computer Graphics and Applications 37.2 (2017)*. pages 106-112.
5. G. Ridge, D. Terzopoulos: "An online collaborative ecosystem for educational computer graphics", *Web3D '19*, July 2019.