CEng 49x - Computer Engineering Design

Project Proposal Form

Important Notes

1. Please read carefully, and follow the instructions below to fill in this form.
2. A project could be proposed by (i) a student or a student group, (ii) a company, or (iii) a faculty member of the department by filling in this form and submitting it to 49x-proposal@ceng.metu.edu.tr by e-mail. For a project proposal, there might be a sponsoring company supporting the project and providing some form(s) of resources for the project.
3. Each project will be carried out by a group of 4 students over the course of 7.5 months, which amounts to 30 person*months. It is very important that your project's workload is around 30 person*months. Please make sure that you have at least a rough justification about the workload of the project.
4. If your proposal might contain a patentable idea or any type of intellectual property, please first make sure to follow the appropriate steps (apply for a patent, etc.) before sending your idea to us. Once this form is received from you, the instructor(s) and the department has no responsibility regarding the intellectual properties of your project/idea.
5. All sources and documentation developed for this course are assumed to be public domain (GPL, CC or similar license) by default. If you need any exception for license and disclosure of project work, please specify this in detail in “Intellectual Property” section of the form.
6. Please note that source codes, documents and issue tracking will be kept in department servers. No restrictions can be requested for limiting faculty and assistants access to student work.
7. Instructions to fill in this form are given in italic fonts and in parentheses. To provide an input for a section of the form, delete the instruction and provide your input in place of the deleted instruction. In the final form that you will submit, there shouldn’t be any instructions left over.
8. If you feel that a particular instruction is not relevant to your project proposal, please use a proper explanation for this, rather than ignoring the instruction.
9. The final form should not exceed 5 pages including everything (even this page). Please use Arial, Normal, 11pt fonts and single line spacing.

Acronym and Title

(GRT) GPU based Ray Tracing

Target

(Please mark one of the following options. If you choose the second option, provide the names and email addresses of the students.) [X] This proposal can be announced to all student groups. It can be assigned to any student group. [ ] This proposal is restricted to the following students/groups.

Proposer Information

Names(s):	*TaleWorlds Entertainment*
Email(s):	*Murat Türe* *murat.ture@taleworlds.com*

Supervisor

(Each project will be supervised by a faculty member of the Department of Computer Engineering, METU, throughout the academic year. Please mark the following options regarding the supervisor. Please only choose the first option if the supervisor has agreed. If you choose the second option, you can give a list of faculty members as suggestions.) [] The project will be supervised by _____________. [X] The project can be supervised by any faculty member. Suggestions: Assoc. Prof. Dr. Ahmet Oğuz Akyüz

Project Description

Until a few years ago, ray tracing was seen as only an offline rendering tool for movies and animations. Recently, with the improvements over the GPU acceleration structures and algorithms and the better hardware, vendors started to explore the ideas of how they can integrate the ray tracing into the real time engines. Microsoft developed a ray tracing API called DXR which runs in GPU and resides inside DX12. It can create highly optimized acceleration structures and provide high level interfaces for the ray tracing methods. This highly elevated how the Ray Tracing is perceived by the developers and paved in the path to use real time ray tracing in Games and Rendering Applications.

At the launch of DXR, two of the mostly used game engine vendors(EPIC and DICE) published videos about how they integrated ray tracing into their pipelines. Depth of field, soft shadows, global illumination, and glossy reflections are some examples where the ray tracing vastly improved the visual quality.

In this project, the students can use an external game engine. Scene loading, object management and the basic frame rendering will be done by the engine. Firstly, they will develop the baseline ray tracing API just like DXR. It will contain the acceleration structures and basic shaders to handle the ray tracing requests. Then, they will use it to better solve the Global Illumination which is one of the most researched open Computer Graphics problem. With the latest sampling methods and acceleration structures for the GPU, small to medium sized scenes with the features outlined above can be rendered in real time with high tier GPU's. At the end, the renderer will be a hybrid one which uses rasterization for direct lighting and RTX pipeline for diffuse ambient.

Tentative Plan

General Purpose GPU Ray tracing API
A general purpose ray tracer should be implemented in the GPU. In compute shaders or pixel shaders, api should support ray tracing functionality which can return hit point geometric properties like position, normal and material properies like diffuse, gloss or transperancy. Scene graph which supports instancing for fast position and rotation update should be sent to GPU. This will be needed for optimized dynamic meshes.
(4 - 6 person*months)

GPU BVH Implementation
There are many acceleration structures which can benefit from the hardware architecture of the GPU's. This should also support dynamic meshes. Dynamic meshes should be updated fastly to get real time performance.
(6 - 8 person*months)

Monte Carlo Global Illumination
Now as the renderer is fast, students can implement a monte carlo solver using the API that they have already developed to compute diffuse ambient in real time. Area lights and punctual lights should be supported. This value should be composited with the direct lighting to compute the final light values of the pixels. To test the final results, off the shelf offlien renderers can be used.(for etc Mitsuba) (6 - 8 person*months).

Sampling Mechanisms (Cosine sampling & Light sampling)
Monte Carlo Estimators uses random samples to generate the final result. The amount of noise in the final image is highly dependent on the sample count and their contribution. Students should add better sampling mechanisms to increase the final quality of the image with the same number of samples.
(3 - 5 person*months)

Sample scenes imported to the Engine
Sample scenes that can be gathered from various sites should be imported and used as showcase scenes. Some parts of these scenes should be dynamic.
(2 – 3 person*months)

Similar Products/Projects

Microsoft DX12 DXR
Microsoft added the new feature of Ray Tracing to the DX12 pipeline. It can harness the computational power of the GPU's. As stated above, Unreal Engine and Frostbite were the ones to utilize this library. This was showcased at the GDC 2018. The videos can be found below.

Frostbite : https://www.youtube.com/watch?v=LXo0WdlELJk
Unreal : https://www.youtube.com/watch?v=J3ue35ago3Y

Also Frostbite engine showcased a workflow in which the artists can see the final image with Global İllumination while editing the scene. It can be seen in the following video:

https://www.youtube.com/watch?v=rhIGBCSv02M

Contributions, Innovation and Originality Aspects of the Project

The DXR API is a very general interface. If the students make the design choices for the Ray Tracing engine with respect to the Computer Graphics problem addressed, performance can be much better. Also, the number of hybrid renderers are very low which can utilize both rasterization and ray tracing to get the best of both worlds. These can also be used for prototyping for offline rendering applications like movies and animations.

Secondly, NVIDIA supports DXR and develop hardware to further increase the performance of GPU Ray Tracing. They showcased their latest GPU line, codenamed Turing, which was specially developed for this task and has much better performance at Ray Tracing with respect to the old hardware. This is a clear indication that the future of rendering will involve much more Ray Tracing. Experience on this front will be very important for the Computer Graphics developers in the future.

Success Measures

The success of the final product will be the combination of:
- The quality of the lighting should be improved vastly after the chosen ray tracing technique added.
- The runtime performance should be near to real time(20 - 30 fps). The chosen acceleration structure technique should be developed for the GPU's.
- Scenes chosen to show at the demo, should be matched with the Computer Graphics problem chosen to improve.
- Small, medium and big scenes with varying lighting environment should be tested and the effect of these changes to the performance should be justified with respect to the algorithms used.

Project Development Environment

Any operating system or development environment can be used. However, Windows, C++, and Visual Studio would be the best for the debugging of the GPU and the engine.

For the engine, Unity or Unreal can be used. Simpler, lightweight engines which can be found on GitHub can also be used

External Support

Support for literature survey can be provided. Also, information about debugging environments and procedures can be proved.

Developers experienced in this subject can and will attend the weekly meetings. (Murat Türe)

Major Risks and Risk Plan

The biggest issue with GPU programming is the lack of good debugging tools. Small bugs can be very hard to trace and fix. Worst case, it won't take too much time to port their code to the CPU and test there.

References

Veach, Eric. Robust monte carlo methods for light transport simulation. No. 1610. PhD thesis: Stanford University, 1997.

Popov, Stefan, et al. "Stackless kd-tree traversal for high performance GPU ray tracing." Computer Graphics Forum. Vol. 26. No. 3. Oxford, UK: Blackwell Publishing Ltd, 2007.

Parker, Steven G., et al. "OptiX: a general purpose ray tracing engine." ACM Transactions on Graphics (TOG) 29.4 (2010): 66.

DXR Tutorials https://github.com/NVIDIAGameWorks/DxrTutorials

DX12 Documentation: https://docs.microsoft.com/en-us/windows/desktop/direct3d12/what-is-directx-12-

NVIDIA RTX:
https://developer.nvidia.com/rtx