DOD vs OOD in Engines

for

Game Creation

1 OVERVIEW

For my project I am going to compare 3 engines I have developed, 2 of them have being written using OpenGL, but each of them with different code structures, one in DOD and the other in OOD. The idea of this project is create a new render engine one using Vulkan as a GPU api and making use of the DOD code structure and then compare its performance with the others.

The main idea is create a render engine using Vulkan, make profiling test and then compare it with the previous engines created. *Phantom*, the engine’s names, will give support to forward, deferred, gltf , obj and maybe fbx. Having enough time making use of ImGui, a tiny editor will be created, but not to with much priority. The libraries I am going to use are: Vulkan SDK, glm (for the mathematics) , gltfw (window creation) , tinyobj and tinygltf.

I will be making my work as modular as possible so that all the actual simulation code will be able to be reused with a different rendering system, this will allow me to use it for future games.

# 2 RATIONALE

The main motivation behind my choice of project is based on my own preference of programming areas which are Graphics and Engines, therefore I have chosen a project that requires me to look into a new programming pattern, which is cache friendly and is being used more now days. It will also allow me to get more experience with graphics programming using a much different rendering system than I have used before, this will help me to be much more confident when using new systems.

# 3 TECHNICAL OVERVIEW

For my project I am going to be creating a render engine which aims to be as an approach to DOD as possible while still being a true render engine and not just imitating previous engines at github. For this project I will be creating an environment within Vulkan which will allow me to create a small testing environment, it will not contain very much geometry, this is because I am not testing these areas. Within the testing environment I am going to be creating a scenery simulation, loaded using gltf or obj, the scene will make use of all the features developed for the engine, which are in project deliverables.

I will then render the scene in the engine using Vulkan’s API. I will create as accurately as possible how a DOD engine is implemented, as the main purpose of the project is compare difference between OOP/OOD versus DOD. I am going to implement a simple render engine, which allow me to create scenes in there and compare them with previous engines developed in OOP/OOD, so most of the time will be spent creating a proper debugger and obtain quantity and quality of samples. This sample will be used to create a document showing the difference between OOP and OOD performance, including the code style, and how different are OpenGL and Vulkan languages explaining the advantages and disadvantages of them.

As having the code style as a hotspot, I will need some volunteers’ programmers to prove that the code is simple to understand and to be used for scene creation. One of the main ideas is make it simple to the user making a proper abstraction to the programmer and consider that the person who is implementing the scene is not a graphic programmer.

Another important spot of the idea is create a good multy-thread system based on [px\_sched.h](https://github.com/pplux/px), this will take most of the time during the development of the project and will be one of the main point of our project, making a proper scheduler will give more useful tools to the user and will make easier to use it during future implementations. In addition a display list design will be implemented to recreate properly a task system.

For my project I am going to need to research in multy-thread and use of vulkan’s API, for example “[C++ Concurrency in Action,2E](https://www.amazon.co.uk/gp/product/1617294691/ref=ppx_yo_dt_b_asin_title_o02__o00_s00?ie=UTF8&psc=1)” and “Vulkan programming guide”.

# 4 PROJECT DELIVERABLES

|  |
| --- |
| Project Deliverables |
| * Create a render engine in DOD * Forward * Deferred * Shadows for directional, spot and point lights * Texture support * SSAO * PBR * GLTF and Obj * Transparencies * Anti-aliasing * Sky box * Component support * Lua scripting. * Multy-Thread * Render Debugger (timers) |

The time I will spend developing the engine would be about 8~12h per day approximately, however is going to be 5 days per week if all the work planned to that week was done, so in this case I will take each weekend for my free time.

In order to produce a Project Diary explaining the work I have done each week and what problems I’ve faced with solutions that have arisen I am going to use blogger provided by Google: [dodenginefyp.blogspot.com](http://dodenginefyp.blogspot.com/)

The deliverable created will be a document explaining the differences between DOD, OOP, Vulkan and OpengGL. This paper will show a performance graphs and showing all the data gathered during the app development from my partners, this paper will show how the code has being changing and explaining the reasons of the changes.

# 5 PROJECT PLAN

Here is a brief overview of how I will be dividing my time over the course of this year outside of my scheduled hours. I have left Sunday open so that I can use that day to do any extra work that I was not able to complete that week on the designated day.

**Task List**

Here is a list of tasks that need to be completed over the course of my completion of the project, tasks are numbered and I have indicated which tasks each task depends on having been completed before they can be started, this way I can see which tasks I can be working on at any given time so that if I get stuck working on one task I can switch to another in the meantime whilst I passively think about how to continue with the first task.

|  |  |  |  |
| --- | --- | --- | --- |
| **Task Number** | **Task Name** | **Brief Description** | **Depends Upon** |
| **1** | Project Gen | Using genie create the script to create the project in visual and add all the libraries. | - |
| **2** | Window creation | Making use of glfw create a window. | - |
| **3** | Render a triangle | Create a simple debug class to render the triangle and debug if all is working well | 3 |
| **4** | Buffers | Create the structure to connect Vulkan buffer to our engine. | - |
| **5** | Frame Buffers | The same as the buffers | 3,4 |
| **6** | Textures | Create a simple class to give texture support. | 3,5 |
| **7** | Forward render | - | 3,4,5 |
| **8** | Deferred render | - | 3,4,5,6 |
| **9** | Lightning | - | 7 or 8 |
| **10** | Shadows | Create the shadows for directional, spot and point lights | 6,10,7 or 8 |
| **11** | Scene creation with gltf and objloader | Create the scenes using gltf and load models using obj to. | - |
| **12** | PBR Material | Create the shader to support pbr. | - |
| **13** | SSAO | - | 8,7 |
| **14** | Lua | Create a scripting to our components as Unity | - |
| **15** | Multy-thread | - | - |
| **16** | Anti-aliasing | - | 7,8 |
| **17** | Sky box | - | - |

# 6 Ethics

All the libraries are open source and can be used for application development and only not claim that you wrote the original software.

# 7 Methodology

I am going to use a scrum methodology making each weak sprints and using the blog as a control of the worked implemented.

* SCRUM
* Blogger

# 8 LITERATURE REVIEW

* *Game engine architecture*: learn more about current engines architecture in the industry.
* *Real-Time Rendering, Fourth Edition:* learn about the new techniques created during last years in render.
* *OOP Is Dead, Long Live Data-oriented Design:* explain the advantages of DOD versus OOP/OOD.

# 9 BIBLIOGRAPHY

-Overvoorde, A. (2019). *Introduction - Vulkan Tutorial*. [online] Vulkan-tutorial.com. Available at: <https://vulkan-tutorial.com> [Accessed 15 Jan. 2019].

- Khronos Group (no date) *Vulkan Reference Pages*. Available at: https://www.khronos.org/vulkan/ (Accessed: 15 January 2019).

-Gamesfromwithin.com. (2019). *Data-Oriented Design (Or Why You Might Be Shooting Yourself in The Foot With OOP) – Games from Within*. [online] Available at: [http://gamesfromwithin.com/data-oriented-design](http://gamesfromwithin.com/data-oriented-design%20) [Accessed 15 Jan. 2019].

-Dice.se. (2019). [online] Available at: <https://www.dice.se/wp-content/uploads/2014/12/Introduction_to_Data-Oriented_Design.pdf> [Accessed 15 Jan. 2019].

-YouTube. (2019). *CppCon 2018: Stoyan Nikolov “OOP Is Dead, Long Live Data-oriented Design”*. [online] Available at: <https://www.youtube.com/watch?v=yy8jQgmhbAU> [Accessed 15 Jan. 2019].

-Gregory, J. (2018). *Game engine architecture*.

-Akenine-Möller, T., Haines, E. and Hoffman, N. (2014). *Real-Time Rendering, Fourth Edition*.