MAGE Game Engine

https://github.com/comp195/senior-project-spring-2022-mage-game-engine

Jordan Scharkey <j_scharkey@u.pacific.edu>

Last updated: April 17th 2022

I. System Architecture

While directly intended to be utilized by game developers, it may be helpful to start understanding game engines by examining they usage as an end-product within a game.

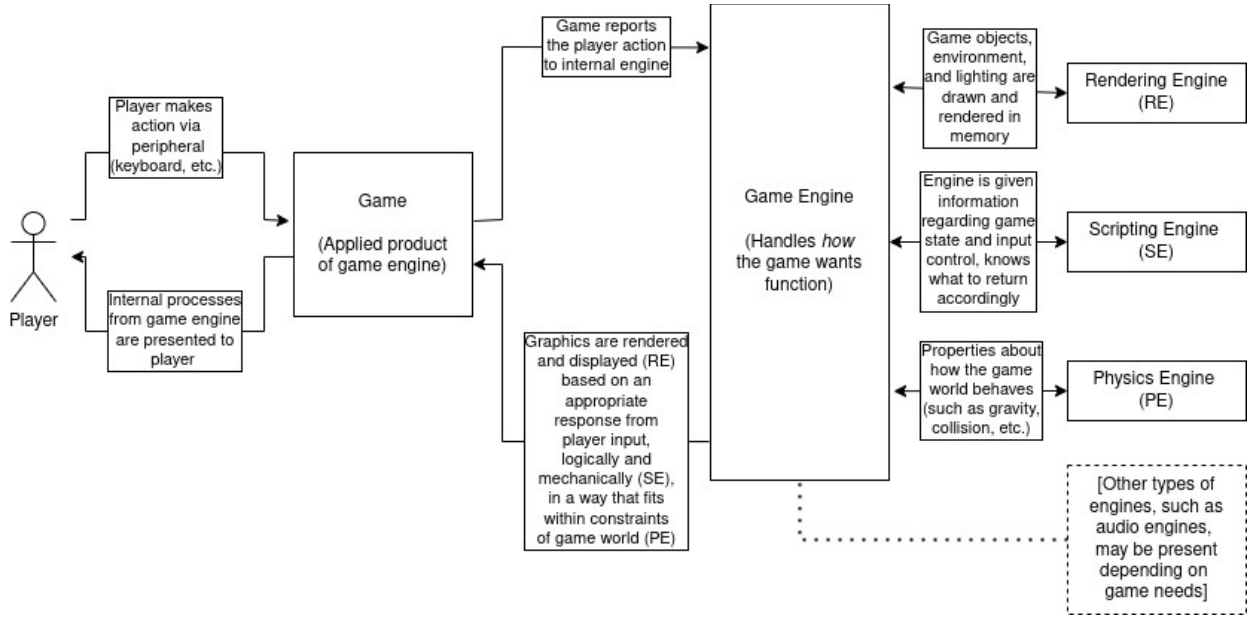

Diagram I.I – Game engine as part of a game

The diagram above shows how a game engine helps a game accomplish the experience it wishes to portray. From a game designer and game developer perspective, I liken the relationship of game and game engine to a book and its binding. An author is responsible for the contents of a book- it isn't too often that an author is responsible for the every aspect of the process of stitching the book together. In this sense, game engines are able to stitch games together for game designers and game developers.

Below, Diagram I.II represents a more accurate view of a game engie from this perspective. For a more direct view of the different intended focuses of MAGE, refer to Section III – Software Design. I inspected two other indepentently created game engines, KOHI (Vroman, n.d) and littleVulkanEngine (Galea, n.d). both available for open source viewing on GitHub, for the framework of game engine structuring.

Different games require different features, different limits, and different required optimizations and efficiencies. This is why there are a plethora of game engines, rather than a single framework that everyone uses. The needs of a online fighting game (network connection, fast and accurate graphical rendering matched with audio cues) are much different than the needs of a point-and-click adventure RPG (large scripting memory capable of accounting for different player choice).

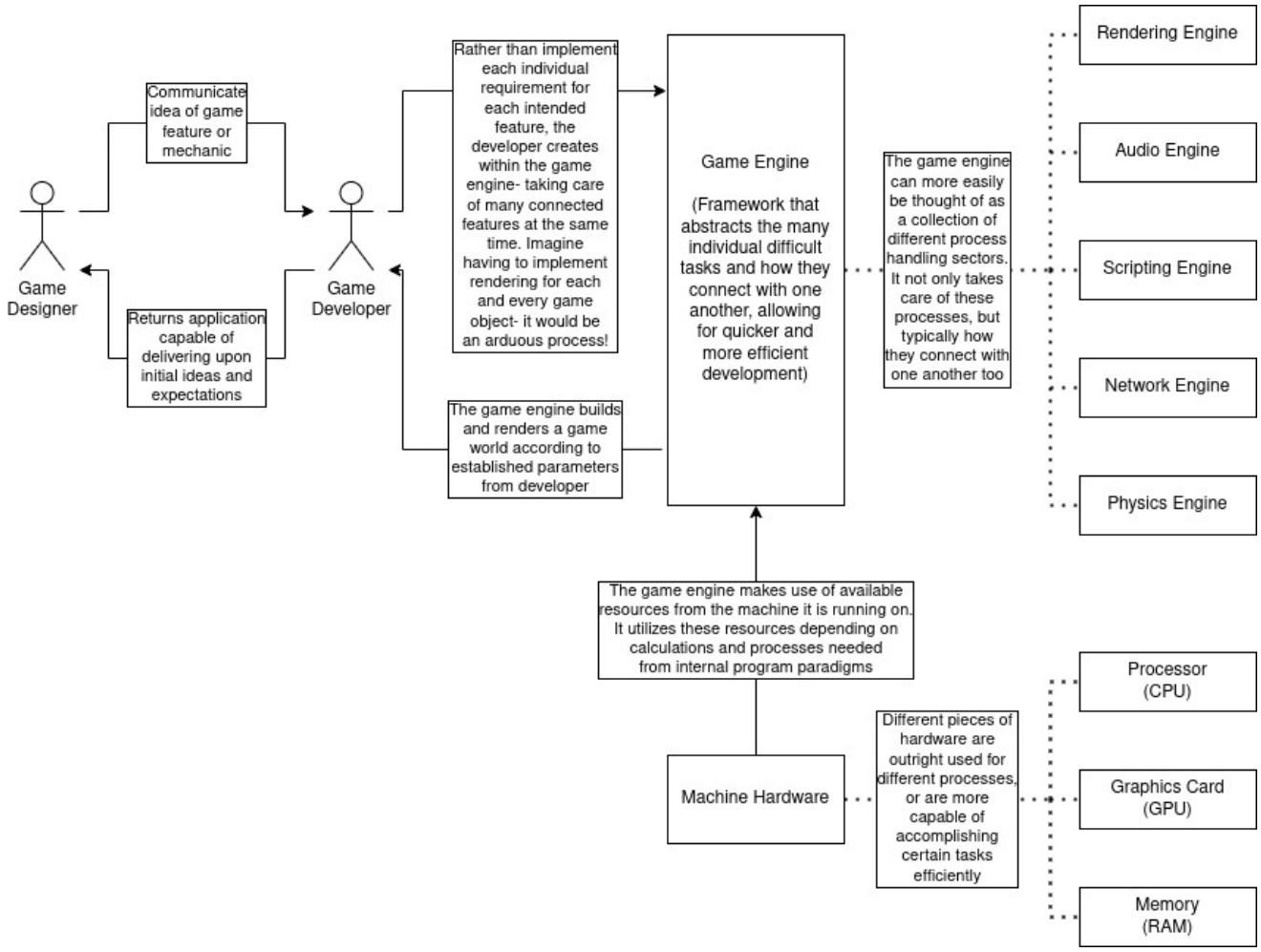

Diagram I.II – Game engine as a tool

II. Requirements

II.I - System & Software Requirements

This engine will be utilizing the Vulkan API (version 1.3) for graphical rendering. Vulkan drivers will be required for whichever operating system the machine hosts. While some aspects of the engine will have to be explicitly written for specific systems, Vulkan handles cross-platform support much more elegantly easily than other graphical APIs as long as the user is developing on Windows, Mac, or Linux platforms. (Home | Vulkan | Cross platform 3D Graphics, 2022)

I forsee myself utilizing existing open-source math libraries to take care of particular rendering equations and some basic physics engine processes. While not explicitly required by an end user, I figured it was worth noting in this section, as it is an aspect of the project outside of Vulkan that I do not see myself writing.

II.II - Hardware Requirements

The machine running MAGE will require some form of hardware capable of performing and handling grahpical calculations. While CPU-integrated graphics and smartphone-esque device support is technically plausible in this sense, it is strongly recommended to have some form of dedicated GPU where these intensive processes can be hosted.

I am currently focused on NVIDIA-based GPU support, thoughtother GPU manufacturers (such as AMD) may be supported if easily implemented within given time constraints. This is primarily because this is what I most easily have access to, but also because I expect to run into resource-intensive processes that may not be feasible on systems without this dedicated hardware. Optimizations make it possible to run on these types of machines, but may not be possible within my development window.

III. Software Design

III.I – Class and Interaction Diagram

Game engines are quite compartmentalized- sprawling many directories of multiple files, containing dozens of classes and hundreds of variables. A traditional UML class diagram of this size and caliber is not possible at this stage, as the sheer size of including every minute detail would take more time than helpful and likely not even be accurate following debugging processes. As such, I've decided to take a much more abstract approach that focusses on particular subsections of the game engine product. I believe this should satisfy the planning and organization purposes of a class diagram in a much more efficent and easy-to-understand method at this level of development.

This diagram (Diagram III.I, shown on next page) will go into more detail regarding what each process of the game engine is responsible of accomplishing, as well as how it interacts with other parts of the program. While I briefly mentioned the individual "engines" within a game engine in Section I, this should be a deeper dive of those ideas. For reasons as to why I choose these particular functions to focus on, see Section III.II.

III.II – Design Considerations

For design choices of what I would like to focus on with my game engine, I wanted to prioritize the future creation of RPGs. As such, I've sacrificed more stellar graphics (capable through methods like ray-tracing) and higher refresh rates (achiveved by alloting more memory towards graphical buffers) to free more space for something along the lines of a scripting engine to handle more options of player choice. I've also gone with the decision to initially create a 3D engine as opposed to a 2D engine, as I've heard that adding 2D to 3D engines is much easier than the other way around.

I am currently not planning for an audio engine, as I feel the limited development time can be better utilized elsewhere. I am not implementing any sort of network engine, as I am more concerned with the future creation of singleplayer experiences than multiplayer ones. I also have limited knowledge in this area, and feel that too much time would be spent researching this area to properly implement it. The basic physics engine will likely be extremely barebones, handling not much more than collision.

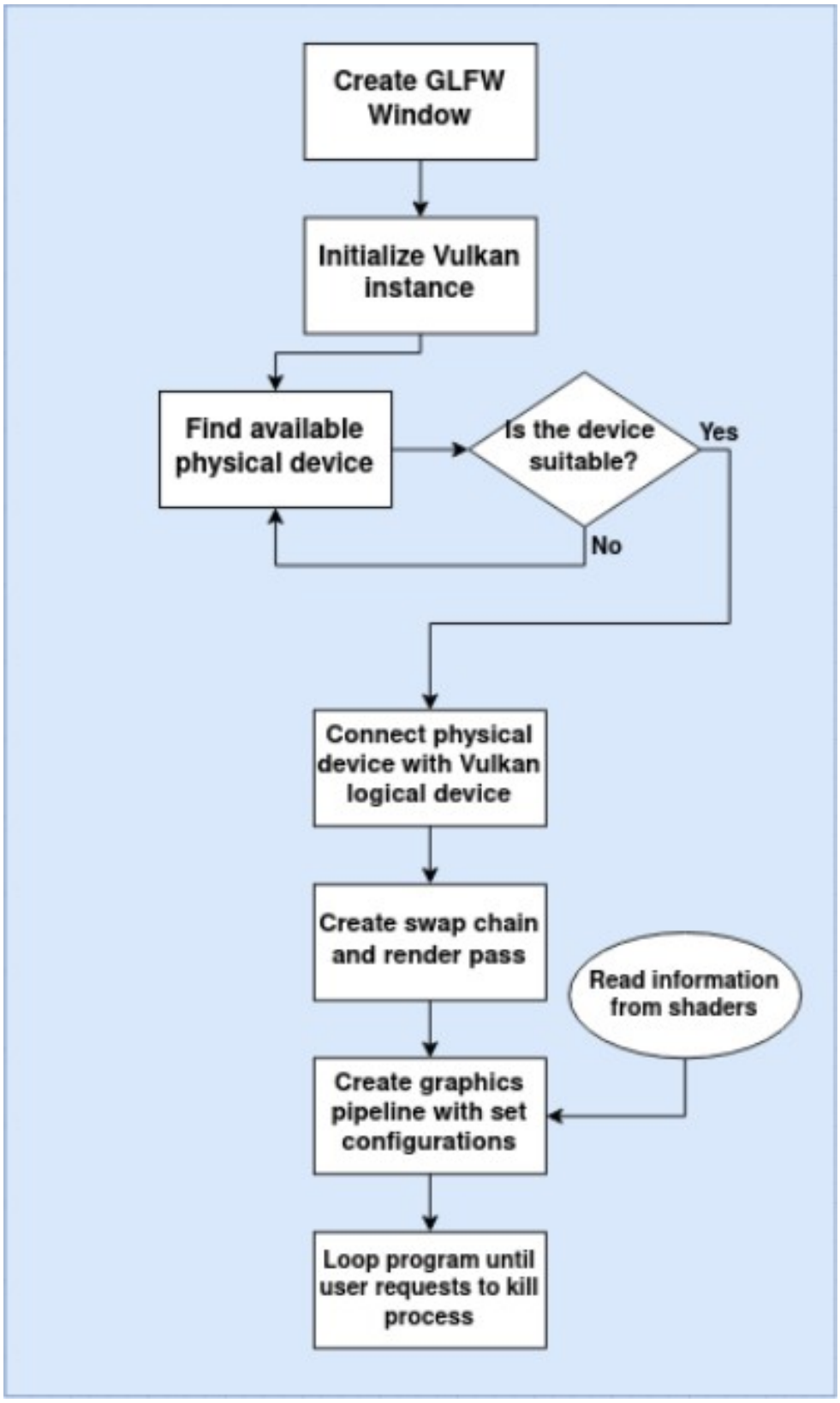

Diagram III.I – This diagram was updated April 17 to reflect the diagram presented in my Senior Project Day poster, as well as more accurately reflect the final product

IV. User Interface

There is currently no planned GUI along the lines of Unity and Unreal for this project, as it falls outside the scope of what I hope to achieve within the time alloted. Users will interact with the engine through their respective IDE and some sort of terminal (within their IDE or not) for compilation, building, and execution.

Dedicated pop-up windows that hold data and analytics may be implemented for testing and troubleshooting if optimization issues become apparent.

References

2022. [online] Available at: <https://vulkan-tutorial.com/>.

Galea, B., n.d. littleVulkanEngine.

Vulkan.org. 2022. Home | Vulkan | Cross platform 3D Graphics. [online] Available at: <https://www.vulkan.org/>.

Vroman, T., n.d. KOHI Game Engine.