**Things which the rendering engine should have**

Todo:

* Finalize the clang-format documentation
* API for games
* Create a full screen GUI for Chronos for easy rendering
  + graph editor using <https://github.com/thedmd/imgui-node-editor>
  + keyframe editor <https://gitlab.com/GroGy/im-neo-sequencer>
  + gizmos <https://github.com/CedricGuillemet/ImGuizmo>
  + file explorer dialog <https://github.com/dfranx/ImFileDialog>
  + look at this <https://github.com/Flix01/imgui/wiki/ImGui-Addons-Branch-Home>
  + <https://github.com/Raais/ImStudio> - use this as inspiration for creating an gui
  + 2d gizmos <https://github.com/Half-People/ImGui-2D-HArrow>
* Write a tutorial on text rendering in vulkan
* Solid colours and deleting shapes
* Animations and rigging
  + Shapes should have be manipulated around a point
  + Keyframe animation with graph manipulation
* Need to add unit tests and figure out the cmake situation for that

Counting number of lines : dir -Recurse -Include \*.cpp,\*.hpp | Get-Content | Measure-Object -Line

**Features to add now (V0.0):**

* add ability to set solid colours to shapes
* add ability to delete shapes
* saving settings to a file
* ability to animate shapes
* post processing implements using Vulkan
* Able to animate characters using rigging and skeletal animation
* Get a way to use the thing
* Complete the README
* Add comments and make things presentable

**Features to add later (V0.1):**

* add ability to change textures from Gui
* add ability to add offsets to textures
* ability to toggle V-Sync
* render circles
* Render heath bars

**Features to add later (V0.2):**

* GPU accelerated particle physics
* multiple textures support
* Multi-threaded rendering
* instancing
* pause rendering when nothing is happening and duplicate frames

**Features to add later (V0.3):**

* direct storage
* texture compression
* use VMA for memory allocation
* mesh shaders
* add pipeline cache
* Convert renderpasses to subpasses

**After creating this engine:**

* physics
* input system
* sound
* making the gamE

**In order to implement post processing techniques:**

**Formulas/Algorithms for post processing:**

* Saturation

https://www.geeksforgeeks.org/program-change-rgb-color-model-hsv-color-model/

* Contrast
* https://www.geeksforgeeks.org/changing-the-contrast-and-brightness-of-an-image-using-python-opencv/
* Gamma
* Offset
* Highlights
* Blue correction
* Vignette
* Exposure
* Chromatic aberration

**Implementation:**

* + 1. Pass it through until we get final buffer
    2. Take that buffer and put ti thorugh a pipeline
    3. Modify the fragment shader of that pipeline to apply post processing effects