Vulkan 1.2 Game Demo GDD

|  |  |
| --- | --- |
| 8/21/20 | Initial Creation |

**I. Purpose**

Make an AWESOME example of a game using the Vulkan rendering API. This should be fun, right?!

**II. Description**

* This is a test project to demonstrate minimal Vulkan games.
* The game is a clone of Super Mario for the NES.
* We will use SNES graphics to make a Super Mario clone with C++ 17 and Vulkan 1.2.

**III. Timeframe**

2 Weeks – 3 Weeks. Due: Sep 4 – Sep 11.

**IV. Art & Design**

* We will rip sprites from Super Mario for the SNES.
* The levels will be designed as Super Mario from the NES.

**V. Architecture**

* The system will be written in Vulkan using C++ and SDL.
* We will use SDL\_Vulkan extensions to use Vulkan.
* We will use the old integer platformer-based physics and 2D sprite animation.
* Cross platform to build. With CMAKE.

**VI. Implementation**

Day 1-5

1. **Code Iteration** 
   1. Task: Get SDL\_Vulkan Rendering system running without bugs or memory leaks.
      1. Get CMAKE to build a small program.
      2. Initialize SDL.
      3. Initialize the Vulkan swap-chain.
   2. Task: Show a blank screen.
      1. Create a textured screen Quad.
      2. Create a shader for the quad (put it right in code).
      3. Present colored quad to the screen.
   3. Task: Create the Game loop & Sprite Objects.
      1. Create a Fixed Time-Step Variable render step game loop (see game programming patterns).
      2. Create component class (see GPP)
      3. Create Render Component
      4. Create Renderer
      5. Create a sprite game object (box).
         1. Draw it as a box for now (single pixel texture).
2. **Design Iteration** 
   1. Rip 3 Super Mario first area sprites into sprites (16x16)
      1. Mario
      2. Ground.
      3. Goombas.
3. **Code Iteration**
   1. Design the screen scaling metrics (to blit a pixilated screen to the quad).
      * 1. Nearest (no) filtering.
   2. Load Sprites somehow
4. **Test**
   1. Show sprites in boxes on screen as a test.

Day 6-14

1. **Code Iteration**
   1. Task: Input.
      1. Research integer platformer physics.
      2. Get a box to move on the screen with the input controls.
      3. Space = Jump
      4. Arrows = Move
   2. Task: Collisions.
      1. Research integer platformer gravity.
      2. Get the box to collide with another ground box and stay put.
      3. Static vs dynamic objects vs backgrounds. (Props vs ..)
2. **Code Iteration – Levels, objects, etc.** 
   1. Create level class.
   2. For now, hard code level into an array.
   3. Possibly use Tiled (again).
   4. Get Goombas to move.
3. **Design Iteration** 
   1. Rip Background from Super Mario.
   2. Question Mark Block.
   3. Other Blocks.
   4. Other necessary ground sprites.
4. **Code Iteration**
   1. Get the character to slide.
   2. Get the character to bounce off Goombas.

Day 14+

1. **Future Iterations (TBD)**
   1. Sprite Animation.
   2. Overworld.
   3. Sound effects.
   4. Change up the world design.
   5. New game entirely.

**VII. Result**

Leave on GitHub to die.