Development of a Entity-Component-System Based Virtual Machine

An Entity-Component-System (ECS) is a design pattern frequently used when making games. This tutorial facilitates the building of an ECS Virutal Machine, USER VM, such that users create games and programs in an ECS Environment. USER OS accompanies USER VM to provide a interface and tools to the user so that they can create, manage, and play the games that they create.

This tutorial focusses on designing, developing, and testing these systems, in hope to have a working environment for the user in the future.

As of May 15th these features of the VM and OS were implemented

Features

- USER VM:
  - Kernel:
    - Entity Component System Program Management
      - Load A Program
      - Save State of a Program
      - Run a ECS Program
      - Pass Arguments to Program
      - Inject Entities into Program
    - File Management Services
      - Load Files
      - Save Files
    - OS Boot System
      - Starts an OS Program on Start-Up
      - Injects Rendering Entities OS Program
  - OS:
    - UI Entity System Library
      - Base Library for UI and Sprites
        - Display Sprites and Transform them
          - Color, Size, Rotation
        - NinePatch, Text, and Texture Support
        - Table Position System
      - Input Systems
        - Mouse Event Systems
        - Keyboard System
        - Drag System
    - Program Management
      - Compile and Run a program in a draggable frame (window).
    - Desktop Interface
      - Program Shortcuts
      - Taskbar
      - Image Background
      - Clock Display
      - Program Frames (Windows)

This tutorial did not exist without challenges. Some included tough programming challenges and some included design and ideology challenges

Notable Challenges:

- Runtime Compilation of Java Files
  - Solution: Groovy Class Loader and Class Management
  - Resource: Groovy Documentation
- Entity Component System Event Systems
  - Solution: ECS Design removes the needs for global events. Design by case by case
  - Resources: A long conversation with Professor Matt Lepinski + various forum posts
  - Note: This challenge solved many incomming problems: anything involving events (Drag System)
- Program Arguments:
  - Some programs need program arguments to run, finding a way to do this without violating ECS ideology was difficult
  - Solution: Entity Injection into Engines by other Systems outside of the injection point
- Rendering Custom Viewports:
  - Manipulating the Viewports of running programs so that they stay within their own frame was difficult
  - Solution: Viewport Component and Systems
- Sharing Render Resources between Programs
  - To efficiently render the program a single Sprite and Shape Renderer should be shared between the OS and ALL programs on the VM.
  - Solution: Entity Injection (See above)

There will be much more challenges to come!
Because there is much more to do!

Resources Used:

- Code:
  - Libgdx Documentation https://libgdx.badlogicgames.com/
  - Ashley ECS Documentation https://github.com/libgdx/ashley/wiki
  - Groovy Documentation https://groovy-lang.org/
- Art/Assets
  - Aseprite Pixel Art Editor https://www.aseprite.org/
  - Libgdx Texture Packer https://github.com/libgdx/libgdx/wiki/Texture-packer
- IDEs
  - Netbeans IDE
  - Visual Studio Code