Team Grapid Project Architecture

Yoonah Lee, Shah Nafis Rafique, Matt Smitherman

The **Model-View-Controller** (MVC) architectural paradigm allows data and it's representation to change independent of one another. MVC allows for multiple views and presentation formats when there are multiple ways to view and interact with data or when future interaction and presentation requirements are unknown. MVC architectures are also commonly used to model web-based applications.

After learning how vital third party software has become to modern software engineering, we researched various frameworks that implement the MVC pattern. To our surprise, most modern frameworks in our target programming language (JavaScript) implement slightly altered versions of the MVC design pattern to achieve a form of data-representation that allows for reactive, dynamic views without sacrificing sound engineering principles such as data encapsulation.

As our project is a web-based graphic design application with ambiguous future requirements, and in light of the aforementioned research, we made use of a variant of the MVC pattern, the MV* pattern, to design our program architecture as follows:

View

1. **Canvas View:** The main program window which displays GUI elements and scene objects defined by the user.
    - The Canvas View is composed of custom vue.js components that define:

-The area of the screen which accepts webGL draw calls

-The reactive elements that bind user input to variables which are then passed to child vue components and models.

Model

1. **Transformation Model:** Applies affine transformation functions to scene object vertex arrays, and perspective transformations to the viewport.

2. **Shader Models:** The Shader models create vertex and fragment shader objects that are then passed which exist in GPU and render scene model updates to view.

3. **Scene Object Model:** Creates and deletes user defined objects, updates model data on view requests, and sends new data to the shaders to be rendered. Objects in the scene object model may have color or texture attributes, and must have a vertex array.