The design paradigm we chose and the nature of the application we are developing lends itself to the creation of a monolithic application architecture. While we have divided the system into components, they are mainly divided this way because of the completely separate use cases for each one. The app we are developing is browser-based and is intended to contain a multitude of games and interactive art created and manipulated using the three.js library. In order to deliver this content, we need to host the application on a server capable of distributing the files. In this way we are implementing a client-server architecture, since HTTP is used to receive the files we have created, but the communication is not two-way besides the initial HTTP GET request for the files. The apps are intended to be interacted with solely in the users browser and have no side effects elsewhere.

Each game is able to be designed independently and relies on the three js library which is included in the js folder of the repository. Any functions and objects in this library can be utilized by the games and art we create. The monolithic design architecture is a result of the fact that the user interface and logic are implemented in the same javascript file and are deeply interconnected as one would expect from a game. The separation of game logic and UI (what little there is) comes from the use of the model-view-controller design approach used in making the games. The application is designed to be self-contained and the only modularity comes from the separation of the games themselves(ie. loading them onclick), not the interactions within them. Although each game itself is modular, it could be implemented on a separate website, the application as a whole is monolithic in the way it is delivered and interacted with and each game is also a monolith.