## Introduction

The website for displaying 3D models has been written in python 2.7 using the [Django](https://www.djangoproject.com/) web framework and using [WebGL](http://www.khronos.org/webgl/) to display the 3D models. Django provides a consistent program structure and automates a lot of the boiler plate code, including database access and administrative pages.

The website is designed as a single page site, with a menu bar down the left hand side to select models from. The models are arranged in a tree structure under administrator-defined headings and sub-headings.

## Program Structure

There are five main folders containing relevant code. The /dajax and /dajaxice contain the files for [Dajax](https://github.com/jorgebastida/django-dajax), which provides asynchronous presentation logic. The /docs folder contains documentation for the project. /media contains the actual 3D models, and /three\_d\_viewer contains the presentation. Django implements the [model-view-controller pattern](http://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller). The logic containing the models and view controllers is in \three\_d\_viewer, and the views are in \three\_d\_viewer\templates\three\_d\_viewer. For in depth descriptions of the contents of that folder, see the Django documentation.

## Code Quality Analysis

Code quality is monitored by using running various tools on the code. Unit testing is performed against the user created code. The tests are in /three\_d\_viewer/tests.py. Django generated code is not unit tested, as it has been tested before it is released. Code coverage is analysed when unit tests are run to ensure that all relevant code is being tested. Code quality and conformance to python standards are monitored by running the [pep8](https://pypi.python.org/pypi/pep8) and [pylint](http://www.pylint.org/) code analysis tools. Note that some of the pylint warnings are false positives, due to the Django framework.

## Development Environment

This section describes setting up a development environment for working on the project. More detailed instructions for installing python can be found for [windows](http://docs.python-guide.org/en/latest/starting/install/win/), [mac](http://docs.python-guide.org/en/latest/starting/install/osx/), and [linux](http://docs.python-guide.org/en/latest/starting/install/linux/).

* Install python 2.7
* Install pip to allow installation of python libraries.
* Install [virtualenv](https://pypi.python.org/pypi/virtualenv). This tool creates a python silo, where any installed libraries are not affected by any other python tools. It removes issues with a library version updating and breaking the code.
* (optional) Install [virtualenvwrapper](http://virtualenvwrapper.readthedocs.org/en/latest/), or [virtualenvwrapper-win](https://pypi.python.org/pypi/virtualenvwrapper-win) for windows to make managing virtual environments easier
* Create a virtual environment, and switch to it. If using virtualenvwrapper, use mkvirtualenv and workon respectively.
* Clone <https://github.com/sfrischkorn/rockviewer.git> from [GitHub](https://github.com/), or fork the repository if you wish to modify it.
* In the main directory of the project, install the python dependencies using pip. pip install –r requirements.txt
* The website is now ready to work on. Use python manage.py runserver to start the development webserver, which can be accessed at <http://127.0.0.1:8000>. The admin interface can be accessed at <http://127.0.0.1:8000/admin>, using the user: “admin”, and password: “password”.

## Deployment

TBD. The Django website has documentation about how to deploy a Django application.