The application use mainly two java libraries: JavaFX and JOGL. JavaFX is used for the user interface to display pages. The JOGL library has two components: glugen-rt and jogl-all; This library is used in the benchmarking part for rendering the objects and extracting OpenGl driver information like frames per second and resolution in order to calculate the score.

The main components of the software are the UI and benchmark package. The application is started from the Main class in the main package.

At first, an Initial Setup page is loaded from main method. The flow of the UI is controlled by the PageLoader in the UI.SceneLoaders package. Each FXML page is assigned a controller which deals with the actionable nodes in the FXML page. If a button like “BEGIN” or “SETTINGS” is actioned, then the controller calls the load method of the PageLoader to switch to the next page.

The PopupLoader is managing the help pop up in the main page.

When the nose is pressed on the main menu, a page opens to select the desired test. Each button calls for a method in main which loads the generator with a specific flag, depending on the chosen test.

The Generator, which implements the Runnable interface, then initiates a JOGL object and an Event Listener object and starts a new thread which runs a loop designed to stop after approximately 20 seconds. The loop deals with the refresh and refresh rate of the rendering, which is set to 100000 frames per second, to really stress the GPU.

The JOGL object, which implements the JOGLInterface, has the purpose of handling the OpenGl components. One of the components is the GL Canvas. The GLCanvas is the panel in which content is displayed and is contained by a RenderFrame – a customized container which extends JFrame. Also, here we extract information about vendor and version of the graphics drivers. To the GLCanvas is assigned the EventListener.

The EventListener, which implements the GLEventListener, has the purpose of rendering objects. Its display() method, which “draws” the vertexes and connections between them, is called by the loop in the Generator by the repaint() method. In each new frame, a new position of the object is calculated by the GL2 specific translation methods and displayed, creating the 3D animation. This adds further stress on the GPU.

Also, in the EventListener class we have the reshape() method which calculates the perspective matrix which has the role of displaying the 3D content on to a 2D plane. We use an orthogonal perspective matrix, which means that object do not get smaller as they get further from the ‘near’ coordinates.

The renderObj() method calculates the vertex position depending on the selected object and is called by the display() method.

Each 3D object has the functionality of ‘drawing’ itself using the GL2 methods.

After the 20 seconds passed, the getScore() method is called, which calculates the score and displayed further in the score page.

Each test is recorded and saved in a database under the blue or yellow team and displayed in the Hostory page. The CatFight class has the role of going through the database and calculating the total score of each team and deciding the winner at the moment of display. This information is displayed in the CatFight page.