Google Summer of Code 2025

Liquid Galaxy Master Web and Flutter Application. Google Earth version

INDEX

1. Introduction. What is this guide, and how does it work?

Welcome to GSoC and congratulations on being selected! Now that you have to focus on starting your project, this document aims to be a guide to make this whole process a lot easier. The code archive will give you a structure on how your application should look like as well as some coding tips, and this written guide serves as a complement to that same code archive. Here you will find information on Liquid Galaxy, Flutter, and Flutter applied to Google Earth in case you have never used them. You will also find tips and advice on how to start your project, as well as how to solve different problems that you may encounter.

2. Basic concepts

2.1. What is Liquid Galaxy?

The Liquid Galaxy is an open-source project founded by Google that enables users to navigate Google Earth, view videos and photos, create interactive tours, and visually display GIS (Geographic Information System) data. It is made up of a 7-screen display, connected together to provide users an immersive panoramic experience. Created in 2008 by Google employee Jason Holt, Liquid Galaxy started out as a panoramic, multiscreen viewer for Google Earth. Since then, it has become a tool used for more than just exploring the Earth, as it is now also used for data visualization, virtual tours, videos, photos, and more. Here's how it works:

- **Each screen is powered by its own computer.** The Liquid Galaxy hardware consists of 3 or more computers driving multiple displays, with usually one computer for each display.
- **The Liquid Galaxy applications are developed using master/slave architecture.** The master screen controls the view, and the other screens (the "slaves") are set up to follow it. When the user moves around on the master screen, the other screens adjust their views to stay in sync, creating a smooth, panoramic effect. This coordination is handled through messages sent over a network using UDP.

Even though Liquid Galaxy often uses Google Earth, it is a separate open-source project. This means developers can create open-source applications to display many types of content in an immersive panoramic environment. As a fun fact, Google Earth itself is not open-source software, although it is free to use.

Companies, nonprofits, and universities often use Liquid Galaxy to present information in a more engaging and interactive way. Google also regularly uses Liquid Galaxy to show and promote its own geospatial technologies at trade shows and exhibits.

2.2. What is Flutter?

Flutter is a cross-platform UI toolkit designed to allow developers to reuse code across different operating systems, such as iOS, Android, web, and desktop, while also allowing apps to interact directly with the features of each platform. The goal is to help developers create high-performance apps for many platforms using mostly the same code, instead of writing separate code for each system. Flutter is open source, meaning anyone can use and contribute to it.

Flutter is known for its fast development. During development, Flutter apps run in a Virtual Machine (VM) that offers a feature called *Hot Reload*, which allows developers to instantly see the changes made in the app without the need for a full recompile. When the app is ready to be released, Flutter apps are compiled directly to machine code or to JavaScript (if targeting the web), so they run efficiently. Flutter is also known for its flexible UI and its native performance, as Flutter runs smoothly on both iOS and Android. This is because Flutter uses its own high-performance engine, and its widgets include built-in support for platform-specific behaviours like scrolling, navigation, fonts, and icons.

Flutter apps are written in a programming language called Dart. When an app is built, Flutter turns the Dart code into machine code to make sure it runs efficiently. Flutter is built using different layers, each one with its own role. Working from the bottom to the top, we have:

- **Basic foundational classes and services.** They provide reusable building blocks for the rest of the system. These include features like animation or painting.
- **Rendering layer.** Controls layout and display. It allows building a tree of visual objects that can be manipulated dynamically and that automatically updates as the app changes.
- **Widgets layer.** It lets developers create the user interface using small building blocks called widgets. Each object from the rendering layer has a corresponding class in this layer. Widgets can be combined and reused. This is also the layer where the reactive programming model is introduced.
- **Material and Cupertino libraries.** These provide a comprehensive sets of controls that, using the widget layer, implement Google's Material or Apple's iOS design languages.

Flutter uses a reactive programming style, which means that, instead of constantly updating the screen manually every time something changes, it is the framework that updates the interface at runtime when the application state changes. The developer describes how the UI should look based on the current app state, and Flutter takes care of updating the screen automatically when things change, similar to a smart assistant keeping the app in sync.

In contrast, most traditional UI frameworks describe the user interface once and then developers must manually update it later when things change. As apps get more complex, this can become hard to manage because changes in one part of the app can affect many others. Flutter also has a huge collection of plugins and packages that add extra functionality, such as camera access, webviews, payments, and more.

2.3. Flutter and Google Earth

With the Google Maps Flutter plugin (which you can download here), you can add maps based on Google Maps data to your application. The plugin automatically handles access to the Google Maps servers, map display, and response to user gestures such as clicks and drags. You can also add markers to your map. These objects provide additional information on map locations and allow the user to interact with the map.
Using Google Maps is incredibly useful, as it is built with reliable, comprehensive data for over 200 countries and territories, and it relies on accurate, real-time location information.

3. Start your project

3.1. GitHub setup

- First of all, make sure “master” (or “main”) is set as the default branch. Second of all, avoid pushing anything directly into it! This is because…

3.1. Code structure

- Make sure you have installed the “dart” extension.

Rough structure of what your project should look like:

```
your_Project/
|
|---- lib/
|      |----- main.dart
|      |----- models/
|      |
|      |----- providers/ (optional)
|      |
|      |----- config/
|      |
|      |----- screens/
|      |            |----- mainScreen.dart
|      |            |----- secondaryScreenOne.dart
|      |            |----- (...rest of secondary screens…)
|      |            |----- secondaryScreenFinal.dart
|      |
|      |----- services/
|      |
|      |----- utils/ (optional?)
|      |
|      |----- models/
|      |
|      |----- linux/ (depending on if you want your project to be visualized in this technology)
|      |
|      |----- macOS/ (depending on if you want your project to be visualized in this technology)
|      |
|      |----- web/ (depending on if you want your project to be visualized in this technology)
|      |
|      |----- windows/ (depending on if you want your project to be visualized in this technology)
|
|---- test/
|
|---- pubspec.yaml
|---- README.md
|---- .gitignore
```

- lib/.
- main.dart. The 'main' archive serves to execute the app and show the main screen (HomeView). It is the entry point of the Flutter application.
- utils. **Utility files. They can help to maintain consistency through the app and to make it easy to manage global settings and styles, for example**.

4. Projects used to create this project

Project title + GitHub link	Contributor	Year	Technology/ Programming language
LG Ship Automatic	**Rofayda**	2024	● Dart (95.6%)

Google Summer of Code 2025

liquid galaxy Master Application

Identification System Visualization	**Bassem**		• C++ (2.2%) • CMake (1.8%) • Swift (0.2%) • C (0.1%) • HTML (0.1%)