6

Building responsive UI interface

In this chapter, we will learn and build the basics of constructing a high performance user interface from scratch. A User interface plays a vital role in any application, it’s a medium through which a user not only interacts with the application but also observe various states of application via visualizations. There is an obvious question that comes to mind – what is need to building UI interface from Vulkan or any other low level API? When it comes to rendering complex 1000’s UI objects with real-time rendering visualization and interaction at the same time, the traditional API might not sufficient enough for responsive rendering rates. Also, each ui object might bring a certain amount of overhead which might not a required for the custom applications. Therefore, the learning in this chapter is not to encourage replacement of any existing GUI libraries but to overcome situation where traditional GUI encounter limitations. The other reason is to build custom UI that provides an application a dedicated fast path without indulging into external overheads with better utilization of GPU.

Building Ui interface is vast and complex topic, this chapter introduce the basic concepts and demonstrate building simple and custom UI’s without dropping the performance. We will start with geometric instancing and learn to draw multiple objects in a single command. Next, we will build relation between different rendering object with Scene-Graph, it’s a logical concept that manages a 2D/3D scene and manages transformation graph. We will manage application event’s such as mouse event and handle interaction with objects.

In this chapter, we will learn step-by-step how to usecover the following topics and by the end of the chapter you should be able to run your first Vulkan application to render a triangle on your system.

* Geometric Instancing – Rendering multiple object in single command
* Getting started with Scene-Graphs
* Transformation Graph - preserving Parent-Child transformation.
* Create complex models with transformation graph
* Implementing a scene with multiple views
* Event management Mouse interaction – hover, cl
* Optimization and scaling
* Summary

# Geometric Instancing

Geometric instancing is feature rendering multiple instances of the same object under single API call. Geometric instancing consist of two type of attributes – attributes that are constant and those changes per instance. In this example, we are rendering a Cube geometry with instancing at random 3D positions. The Cube attributes like vertices position, color are *constant*, however the spatial position of each are *different*.

Instancing is helpful in rendering same types of objects in application like particle system, vegetation, crowd simulations etc.

This recipe demonstrates rendering of 1000 cubes using geometric instancing, for this we use 1000 matrices in a VBO, each matrix contains a transformation to place a cube in 3D space. The information of the matrices are updated using range map buffer feature as discussed in the previous recipe. This allows to pass new transformation data on-the-fly at run time, the transformed data contains new rotation and translated positions.

Insert image B06926\_01.png

7OT\_03\_04.png