A 3D Animation and Effects Framework for Mobile Devices

Vipin Patel
Samsung R&D Institute
Bangalore, India
vipin.patel@samsung.com

GBCS Tejaswi Vinnakota
Samsung R&D Institute
Bangalore, India
tejas.v@samsung.com

Soumyajit Deb
Samsung R&D Institute
Bangalore, India
soum.deb@samsung.com

Manjunatha R. Rao
Samsung R&D Institute
Bangalore, India
manju.r@samsung.com

1. Introduction

Of Late, mobile devices with OpenGL ES compliant graphics processors have become all pervasive. However popular platforms like Android do not support any form of power efficient 3D content loading, animation or 3D interaction even though the entire user interface system runs OpenGL ES under the hood. In this work, we present a light weight, power efficient rendering pipeline for enabling new innovative use cases of 3D data by allowing art directed animation and effects created by an artist/UI designer. We utilize the open spec Collada format to seamlessly incorporate 3D geometry into user interface elements, skeletal animated visualizations and interesting lock screen effects in a lightweight manner.

2. System Design

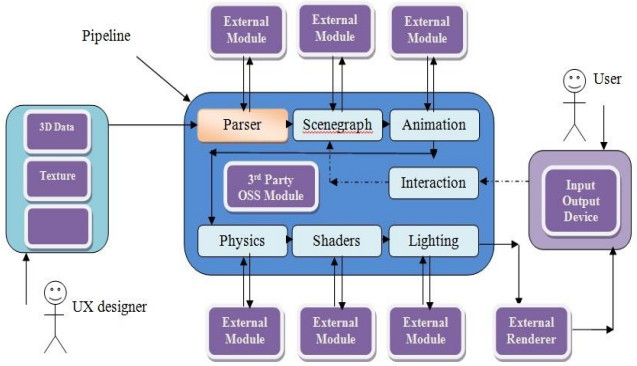
Our system was designed to primarily solve two problems – (i) Incorporating artist generated digital 3D content into every facet of mobile graphics without significant programmer intervention and (ii) Building a 3D pipeline that works upon a 'Power First' philosophy where power saving methodologies are built into the pipeline from ground up. We build a super lightweight framework to supply 3D content to any application or UI elements that may request it. A commercial engine may be too large in memory footprint, storage and power usage for such an application.

We built this system as set of pluggable loosely coupled modules around a scenegraph system. This would allow us to package only the modules we require in a particular application – saving space and reducing power consumption. The system exposes a client API to an application to load a particular model with its associated shaders and textures into a user defined scene graph. The entire system then runs at a smooth and interactive 60fps on commodity mobile devices with an OpenGL ES 2.0 compliant GPU**.** One major challenge for low latency, real-time performance was that loading XML based Collada files at runtime is quite slow if the model is larger than a few thousand primitives. To avoid this wait, we run an offline preprocessing step to convert the Collada file into a simple binary file format. The parser module runs on a desktop and may also optimize meshes to be more power and performance friendly with mobile devices. On the actual device, both power and load times are reduced significantly by this optimization.

The scene graph module is the heart of the pipeline and is responsible for most of the core functionality. The scene graph system logically organizes data into various containers or compartments segregated from each other to apply the optimal data transport path right down to the GPU. For an example, characters with skeletal animation rigs need to be evaluated very differently from repeating geometry around the world which could be instanced on the GPU for optimal rendering. The system also allows 3rd party open source modules to be integrated.

SIGGRAPH 2014, August 10 – 14, 2014, Vancouver, British Columbia, Canada.

ACM 978-1-4503-2958-3/14/08



System Architecture

3. Results

We built a wide variety of proof of concept demos using this system. The use cases are varied – ranging from a skeletal animated avatar as an overlay to assist the users, a virtual character navigating a 3D maze, a key-frame animated scene of a car crashing into a wall to an interactive lock screen effect where the user must push billiards balls into respective pockets in the table and a 3D text rendering system. The entire system was written in C++ with OpenGL ES for rendering, accessible to a Java Android application via the JNI interface. The footprint of the compiled library is extremely small – in its entirety taking just 60kB of space on the device. We also measured the power consumption, memory usage and size of the framework in various scenarios and compared it to Unity Engine below.

	Effects Engine	Unity
APK size(incl. textures)	963KB	8.77MB
Shared library (.so) size	66KB(1.so file)	180KB(3.so files)
Power consumption(avg.)	996mW	1189mW
Run time memory	8152KB	26736KB
Load Time	1.05 seconds	2.54 seconds

- Details of model used for comparison:
- Number of triangles: 7695
- Number of textures: 5
- Size of original collada (.dae) file: 1193KB
- Size of binary (proprietary): 725KB

4. Conclusions

We presented a lightweight, power efficient pipeline for rendering art directed 3d content in various possible scenarios using the open Collada format. Some future directions include incorporation of motion capture data into skeletal animation and also coming up with metrics to quantify an explicit relationship between device power consumption and rendered data.

References:

Mochowski B, Lahiri K, Power analysis of mobile 3D graphics in DAC '06: Proceedings of the 43rd annual Design Automation Conference

Mochowski B, Lahiri K, Signature based workload estimation for mobile 3d graphics In DAC '06 *Proceedings of the 43rd annual Design Automation Conference*

Unity Game Engine – http://unity3d.com