Sumin Wang

Old-Town Dotonbori Viewer on Android Device

Purpose:

Today, the Dotonbori district is a famous tourist attraction and night spot in Osaka that is easily recognized by its canal-side location, restaurants, large billboards, cartoonish statues, neon signs, and food stands. But its original function when it was built in 1621 was an entertainment center, when it hosted dozens of Bunraku, Kabuki, Takeda, and Karakuri theaters, and cafes and restaurants to support the pleasure-seekers who came. Unfortunately, this cultural treasure was destroyed during World War II, but Japanese researchers have, based on old photos, recreated a 3D visualization of old-town Dotonbori focusing on five theaters. This 3D model is complete with building details, accurate scaling and location, and textures. The purpose of the old-town Dotonbori viewer is to display the 3D model on an Android device in an augmented reality, so that one may go to the Dotonbori district and with an Android device view the model superimposed upon the current landscape.

Design:

The first initiative was to use Unity3D to create the application, but it turned out that Unity3D is a licensed software and is not accessible without a fee. So the plan turned to writing the program on the actual Android framework. The first step was to manipulate 3D objects on screen based on the sensor inputs from the Android device. The next step was to import the Dotonbori model into the application. If those were finished in a timely manner, further features could be added to increase the back-end efficiency, improve the design of the system, or increase the user experience.

Procedure:

1. *Manipulating 3D objects on screen based on sensor inputs from Android device.* The first step was to extract sensor data from the Android device and analyze the accuracy, precision, and error. The type of useful data available were orientation, accelerometer, gyroscope, rotation, geomagnetic, and GPS. The second step was to choose a way to render 3D images on screen, which fell de facto to OpenGL. A test application was created in which a simple object was placed on screen, orientation data was generated from the sensor input, and the object was rendered on the screen based on the yaw and pitch of the Android device. At first, location data was not used because the GPS data afforded to Android devices was often intermittent and coarse. Other options were considered, incluing setting up an indoors location tracker and using a framework based on a WiFi network. In the end simple built-in GPS sensors were used. Since the area to walk over would be small, a system was set up to translate GPS coordinates onto a Cartesian plane, using a central GPS location as the origin. In order to test the GPS-based functionality, buildings on the Toyonaka campus of Osaka University were modeled in Google Sketchup using the Google Earth pipeline and uploaded into the Android application along with their respective GPS data.
2. *Import Dotonbori model into application.* Upon researching methods of importing 3D models into Android applications, a useful and lightweight framework that came in source code format named min3D was found. Min3d was particularly well-suited for this project because it is built over the OpenGL API and provides object-oriented graphics functionality in Android projects; it also has features for parsing and rendering 3D data from 3ds files, which is rather popular in industry and is one of the formats the Dotonbori model comes in. In fact, many existing applications such as Autodesk, Blender, and

Google Sketchup can export exising 3D models into 3ds format. Uploading 3ds models became instantaneously feasible. However, when the Dotonbori model was uploaded, it was discovered that in the Android system each application is allowed a maximum of 32 to 48 MB of heap space; if one is using Honeycomb or above one can request an increase to 128 MB. The Dotonbori model contains 2.7 million polygons. In practice it was discovered that about 1/10 of the Dotonbori model could be rendered at a time without generating an OutOfMemoryException. But this was without even the functionality of textures. The min3d framework has a documented bug with rendering multiple textures per object in 3ds files that is difficult to remedy without extensive knowledge of the OpenGL framework. Textures used with 3ds files are also tricky in the sense that the 3ds format was invented in the DOS era and any files it links to is expected to have a name with a maximum of 8 characters.

Results:

The final result came in two pieces. The first piece was an augmented reality application that could view models of buildings from the Toyonaka campus of Osaka University in a location-based augmented reality. The image feed of the models was superimposed upon the actual camera feed and the angle of the virtual camera was matched to the angle of the Android device. The image feed was relatively stable; the noise from the sensors had been filtered out with a queue buffer and the intermittent updates from the GPS sensor did not seem to make the image change at an unsightly pace when the tester walked at a normal pace. The second piece was a virtual reality application that could view a portion of the Dotonbori model based on the orientation of the Android device. There were no textures and only directional lighting

Conclusion:

The project could not hit the milestone hoped for. To complete the project, textures would be added, and the Dotonbori model would be divided up into sections and only a few sections would be rendered at a time based on which ones should be visible from one's current location. Then the objects in each section would be assigned GPS locations so that the application could be tested at the Dotonbori district.

Further steps:

To further increase the user experience, the lighting could be based on time; during the day time it could depend on the location of the sun and at night it could come from point sources on the actual model, resembling lanterns and lamps. Also, to increase the robustness of the entire system, a script could be added to preprocess all texture files so that their names are acceptable for 3ds files. More research could even be done in the area of 3D file formats in the interest of memory usage optimization.

References:

http://developer.android.com/reference/packages.html

http://www.google.com/events/io/2011/sessions/memory-management-for-android-apps.html

http://code.google.com/p/min3d/

http://support.google.com/sketchup/bin/answer.py?hl=en&answer=167461

http://www.khronos.org/opengles/documentation/opengles1_0/html/

http://www.leolol.com/drupal/tutorials/3d-graphics-jogl-opengl-etc/jogl-lesson-5-cameras-movement-opengl

Norman Lin. *Advanced Linux 3D Graphics Programming.*

http://www.movable-type.co.uk/scripts/latlong.html

Dave Shreiner and Bill the Khronos OpenGL ARB Working Group. *OpenGL Programming Guide: The Official Guide to Learning OpenGL, Versions 3.0 and 3.1* .

http://stackoverflow.com/