AR Sandbox for Construction Planning

Traffic Simulation Feature Update

Technology Review

Prepared for Dr. Joseph Louis

Oregon State University

School of Civil and Construction Engineering

December 4, 2018

Team Augmented Construction Education

X, X, and Adam Sunderman

Abstract

The following document describes three technology reviewed components for a proposed upgrade to the Augmented Reality Sandbox located at Oregon State University. Currently the AR Sandbox has the ability to display topographic information and roadway segments within the sandbox. These features are used by students and instructors respectively to learn and teach construction planning. The proposed upgrade will refine the functionality and usability of previously built modules as well as add new functionality in the form of traffic simulation and interactive object placement.

I. Introduction

The following document rationalizes three technology options for the AR (Augmented Reality) Sandbox for Construction Planning. The three components under review are the sandbox's camera unit, the three dimensional (3D) models used for scene building, and a helper program that will be used to work with Digital Elevation Models (DEMs). These components and their functions are described in more detail after a brief overview of the AR Sandbox as a whole.

The AR Sandbox is a physical box of sand equipped with two cameras (depth and RGB), a projector and a control computer. The depth sensor camera reads height information from the sand in the box. This height data allows the projector to display a color gradient over the sands surface, much like a topographical map. Eventually, the RGB camera will read 'markers' from the sandbox. Markers will be unique physical objects placed in the sandbox by users and should be recognizable through computer vision. Once identified they will signal the system to project a particular 3D model at the markers location. These models will be used to create entire 3D scenes of roads, buildings, street signs, and other urban/transportation features. One major goal of the AR Sandbox is to run traffic simulations on 3D scenes that require no coding and no keyboard/mouse input by the user.

II. Camera and Depth Sensor Unit

A. Microsoft Kinect V2 Camera

The system currently uses the Microsoft Kinect V2 and will probably continue to do so. The Kinect V2 features a 512x424 time-of-flight (ToF) camera with active IR for reading depth and tracking objects in low-light situations. The ToF camera gives the proper depth measurements we need to create DEM's of the sandboxes terrain. It also features an RGB camera that captures video in 1080p and can output video at any resolution for the connected display. The RGB camera should theoretically work well for reading markers and object recognition. We will likely use the Vuforia SDK to handle object recognition. Vuforia works well with most any webcam so the 1080p camera should be sufficient for our needs.

The first reason for continuing to use the Kinect is that the system is currently integrated with the Kinect API. The Kinect camera and API has been used in many educational and research projects making it easy to find information on improving the units performance in specific situations. Even though Kinect devices are no longer supported and Microsoft recommends upgrading to Intel's RealSense line of devices, the Kinect API still offers the functionality we need.

The second reason is that the Kinect camera offers the best effective distance for a depth sensor at 0.5m to 4.5m. Intel's RealSense is more suited to facial recognition with effective depths of 0.2m to 1.5m. Given that the AR Sandbox is quite large, measuring over 6 feet in height, a camera with the best effective distance is going to be key for accuracy in terrain modeling.

B. Intel RealSense SR300

The RealSense depth camera from Intel is the next likely candidate for the project. The camera offers a coded light technology for depth sensing which is more accurate that the Kinect but limited in it's effective range of 0.2m to 1.5m. The RealSense cameras are mostly marketed as facial recognition devices because of this and it is likely the APIs also contain function related to these sort of tasks. The RealSense cameras do offer support with various programming languages and platforms. We would need Unity and C# support which is both included.

I see the RealSense cameras as the future for this sort of development project but at this time I believe that there needs

to be a wider range of camera models. The Kinect camera was designed for capturing large areas with multiple complex bodies where the RealSense is designed for capturing facial features. Eventually I believe there will be RealSense models that will be more suited for capturing scenes, but until then the Kinect wins.

C. Asus Xtion 2

The Asus Xtion 2 is another new depth sensing camera with similar specs to the RealSense but with better effective distance. The Xtion 2 is not limited to 1.5m like the RealSense but instead closer to the range of the Kinect at 0.8m to 3.5m. It also uses another different API and Asus bundles the open source OpenNI2 with the device. OpenNI2 does offer C++ and C# support, but that's about it. The API is far less developed than proprietary solutions and development doesn't seem to be too quick. The Xtion 2 is also the most expensive. At 300 dollars its 3x as expensive as a Kinect and 5x the cost of a RealSense.

III. 3D Model

A. Asset Store (Game Engine Dependent)

The AR Sandbox currently uses the Unity Game Engine and most likely will continue to do so. To this end the 3D models that we use for creating scenes could likely also come from Unity's Asset Store. Most major game engines offer a store where you can by just about any game asset from full 3D object models to simple textures and materials. The prices vary widely based on the quality of the item, but often reasonable models can be found relatively cheap. Since most of our models follow the theme of transportation and objects that would be found near these transport routes we can use an asset pack to meet our needs. Asset packs are collections of items from the asset store that follow a particular theme or are just general purpose for complete game levels. With minimal research we can find a pack that not only works for us now but could include extra models for later expansion.

B. Procedural Tool (City Engine)

Another approach to creating models for scenes is procedurally generated models. Programs such as ESRI's City Engine do just this on a large scale. Models made in programs like this follow some sort of predefined set of rules that are used to create semi-unique 3D models like buildings and streets.

There are multiple advantages to using a procedural tool but there is also overhead which we may not be able to afford. The advantage to using a method such as this is that the proper setup can allow for many buildings to be built using the same rule yet still having a large amount of model variation. Another advantage is that models are never stored unless desired, these models can be created and thrown away at will. The obvious disadvantage is the hit to processing power. The generation of models such as this is not cheap as far as the CPU and GPU is concerned. We already have graphics being generated by the AR Sandbox's control computer and we plan to add much more work for it. In light of this it would probably be wise to just have a collection of models on hand that we can call on when needed rather than wasting processing power on model generation at runtime.

C. Build Our Own (Blender)

Another option which is possible would be to just create our own models using a modeling program such as Blender. Most of the models in our scenes will be fairly simple. They probably won't need many vertices and they won't need complex

materials and textures. While the process of modelling is time consuming it might be worth spending a little time creating some models on our own. We could always decide to do a hybrid approach as well, where we purchase the models we feel we don't have time to build. One advantage to creating our own models is that we can suit the needs of our customer better. If there is a particular model that the customer wants we can build it to specifications.

IV. Digital Elevation Models

A. GDAL

GDAL is the Geospatial Data Abstraction Library. It is open source and provides a set of routines for creating and editing DEM files. DEM's, Digital Elevation Models, are images that using shading and pixel values to store the height information of a terrain. The depth sensor in the Kinect camera produces a style of DEM when it processes depth data. These elevation models are currently used to display topography in the AR Sandbox but they can be used to save the current state of the sand as well. GDAL is lightweight and provides the exact set of methods we would need for this type of feature. GDAL is best used for converting between various DEM formats and making changes like blending DEM's. GDAL is used in a wide variety of larger software suites deemed Geographical Information Systems or GIS's. This means for us that GDAL it is a tested and worthy tool that doesn't come with the extra, and unnecessary, features in a full GIS program.

B. QGIS

QGIS is an open source GIS that incorporates GDAL into its packages. The advantage of choosing a full featured GIS is the additional geographical tools present that could come in handy later. GIS systems offer options such importing and exporting real life terrain or adding layers for geographic features other than elevation. When exploring the idea of creating full, robust scenes we should not only be thinking about what our tools can build for us now, but what could they build for us later. A tool like QGIS could make it much easier to model real-life areas inside the AR Sandbox.

C. USNA MicroDEM

MicroDEM is a small GIS program somewhere in between QGIS and GDAL. It cuts down on some of the unneeded (by us) modules that are present in QGIS but still has more features than GDAL. MicroDEM is certainly a possible solution but still might be more than we currently need in a GIS program.

V. Conclusion

- Upgrading the current cameras on the AR Sandbox is unlikely to happen. The depth sensor technology on the market cannot offer any gain in performance that justifies swapping out the Camera and API we are currently using.
- When considering the models we use to create our scenes we will look at a hybrid approach; buy what we can, then build whatever else we may need. This will save us some time on tedious tasks but also ensure we have a full collection of models that will satisfy the AR Sandbox users and our customer.
- For dealing with terrain data and DEM's we should use GDAL, it's lightweight and has enough of the feature we need. Additionally it is scriptable and therefore accessible in Unity.

VI. References

i ***Microsoft Kinect V2 Documentation***. https://developer.microsoft.com/en-us/windows/kinect

ii ***Intel RealSense Cameras***. https://realsense.intel.com/

iii ***Asus Xtion 2***. https://www.asus.com/3D-Sensor/Xtion-2/specifications/

iv ***Unity Asset Store***. https://assetstore.unity.com/

v ***ESRI City Engine***. https://www.esri.com/en-us/store/city-engine

vi ***Blender***. https://www.blender.org/

vii ***GDAL***. https://www.gdal.org/

viii ***QGIS***. https://www.qgis.org/en/site/

ix ***USNA MicroDEM***. https://www.usna.edu/Users/oceano/pguth/website/microdem/microdem.htm