CS CAPSTONE DESIGN DOCUMENT

DECEMBER 4, 2018

AR SANDBOX FOR CONSTRUCTION PLANNING

PREPARED FOR

OREGON STATE UNIVERSITY

DR. JOSEPH LOUIS

PREPARED BY

GROUP 44

RAJA PETROFF
ANDREW SOLTESZ
MARK SPROUSE

Abstract

The purpose of this document is to describe how we plan on implementing the required functionality of the AR Sandbox. This means describing how the various display modes, such as calibration, depth, design, and cut and fill, will work, as well as the lower level graphics systems that are necessary for these display modes to function. Finally, we discuss the data generated and stored by the software, as well as the actual physical components of the AR Sandbox itself.

CONTENTS

List of Figures

1 INTRODUCTION

1.1 Purpose

This document describes the design and architecture for the AR Sandbox. Based on the requirements set in the corresponding Software Requirements Document, the design document elaborates on how the system is made.

1.2 Scope

This document expands on the features defined in the Software Requirements Document, by describing the implementation and data flow required to achieve the desired functionality. This document will cover the development timeline, testing protocols, and the API of the system.

1.3 Glossary

- Augmented Reality Sandbox (AR Sandbox): A physical sandbox with a depth sensor and projector that displays graphics on the sand, such as roadways and ground topology.
- Augmented Reality: A term referring to any technology that superimposes computer generated imagery on the real world. In this case, the term refers to the graphics projected onto the sand. When a user manipulates the sand, the graphics displayed on the sand will change.
- Projection: The image cast on the sand's surface by the projector
- Depth Sensor: A digital imaging device which uses a grayscale image to represent the distance from the sensor to the nearest surface.
- SSD: Software Specification Document

2 SYSTEM OVERVIEW

The AR Sandbox for Construction Planning is an augmented reality sandbox with additional functionality built in for civil engineering applications. At any point, the sandbox will be set to one of four different modes. These modes and how they behave are outlined in this section, after the system components are discussed.

2.1 System Components

Figure 1 shows the different hardware and software components of the AR sandbox and how they interact. This diagram serves as a general overview of the entire system, with each component being explored in greater detail later in the document.

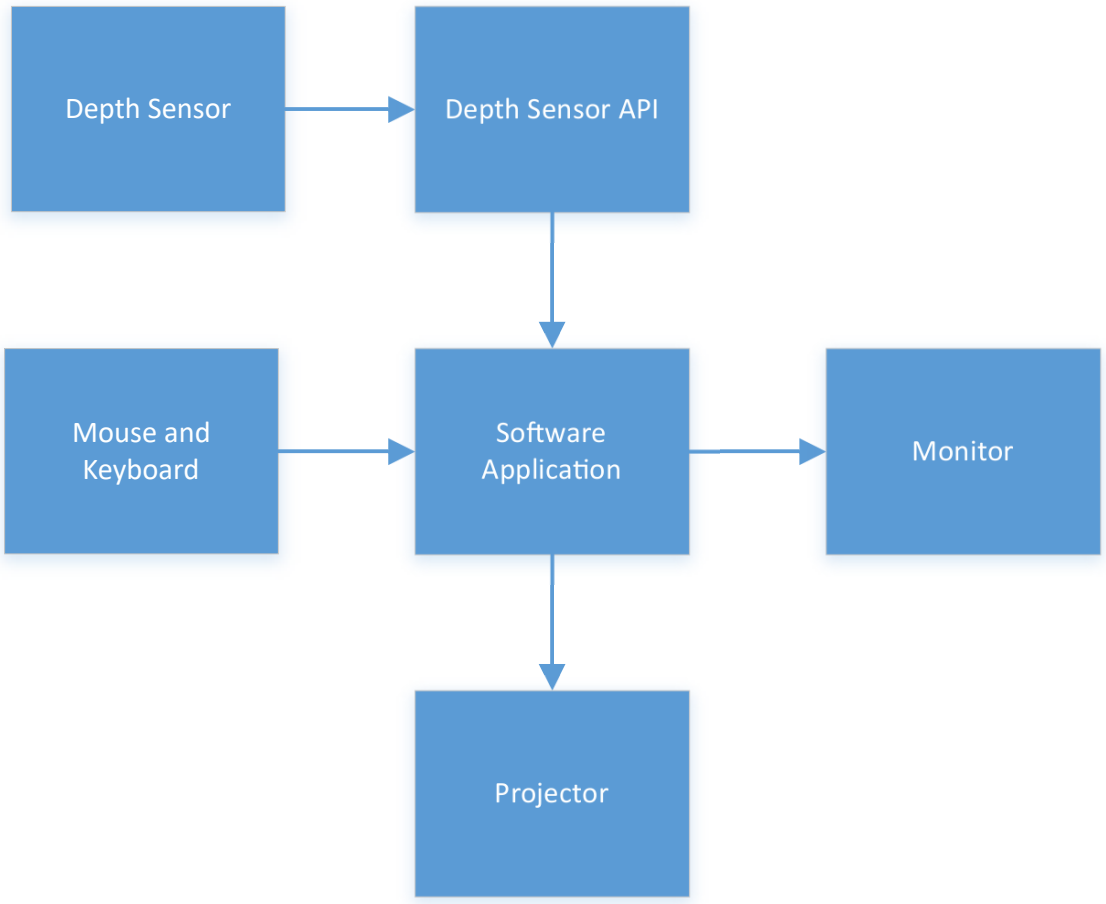

Fig. 1. An overview of the system's components and how they interact

2.2 System Modes

2.2.1 Calibration Mode

Calibration mode is used to calibrate the depth sensor and projector, mainly for use during the initial setup process for the sandbox. A summary of features is listed below:

- **User switches to Calibration Mode** Current bounds of the projection area are shown on the sand.
- **User uses the computer terminal to adjust bounds of projection area** A bounding box appears when the mouse is moved, this box dictates the area defined as the sandbox.
- **User calibrates depth sensor by placing an object with flat sides on the sand** Using height sensor data, the outline of the object is projected onto the sand. The computer terminal is used to move this outline around until it lines up with the physical object.
- **User changes the current display mode** The current display information shall be replaced with the information that corresponds to the display mode selected.

2.2.2 Depth Mode

While in Depth Mode, the program will visually represent the height of the sand using color and contour lines. Below is an overview of Depth Mode's features:

- **User observes sandbox** The different heights at different points along the surface of the sand shall be visually represented via a projection. This projection will consist of a 2-dimensional representation of the height projected onto the 3-dimensional surface of the sand.
- **User physically manipulates the sand, changing the landscape in the box** The visual representation projected on the sand shall adjust in accordance to the new height of the sand in the areas that have been altered.

2.2.3 Design Mode

Design mode is used to control the shape of the road segment that is projected onto the terrain. A summary of features is listed below:

- **User switches to Design Mode** A segment of road is displayed.
- **User uses the computer terminal to edit the path of the road** The user shall be able to adjust the position and curvature of the road. The user shall be able to place additional manipulation points along the road which can be used to further refine the path of the road. A edit/reset button shall be present to revert the road back to its previous path. The User shall be able to undo unwanted changes.

2.2.4 Cut & Fill Mode

Cut and Fill mode is used to visualize the volumes of material that must be added or removed in order to achieve the desired road path. A summary of this mode's features is outlined below:

- **User observes sandbox** A road shall be displayed traversing the surface of the sand. Along this road, the amount of sand that would be required to be taken away or put in place in order to achieve the desired alignments along the road shall be visually represented. In addition, the total haul quantity, average haul distance, average haul grade, estimated haul duration, and the construction equipment to use shall be displayed.
- **User physically manipulates the sand, changing the landscape in the box** The representation along the road and the additional information described above shall adjust in accordance to the new height of the sand.

3 System Architecture

The purpose of this section is to provide the model of our AR Sandbox System. The Architectural Design subsection demonstrates how this system is comprised of various components which interact with one another; these relationships and interactions are explained and displayed with diagrams and figures. The Decomposition Description subsection goes into further detail about each of these components and how they are implemented. Lastly, the Design Rationale subsection provides reasoning behind some of the design choices presented previously.

3.1 Architectural Design

Figure 2 gives an overview of the software architecture and shows how the different components, such as the heightmap and vertex/fragment shader takes the input from the depth sensor and renders it into a terrain mesh.

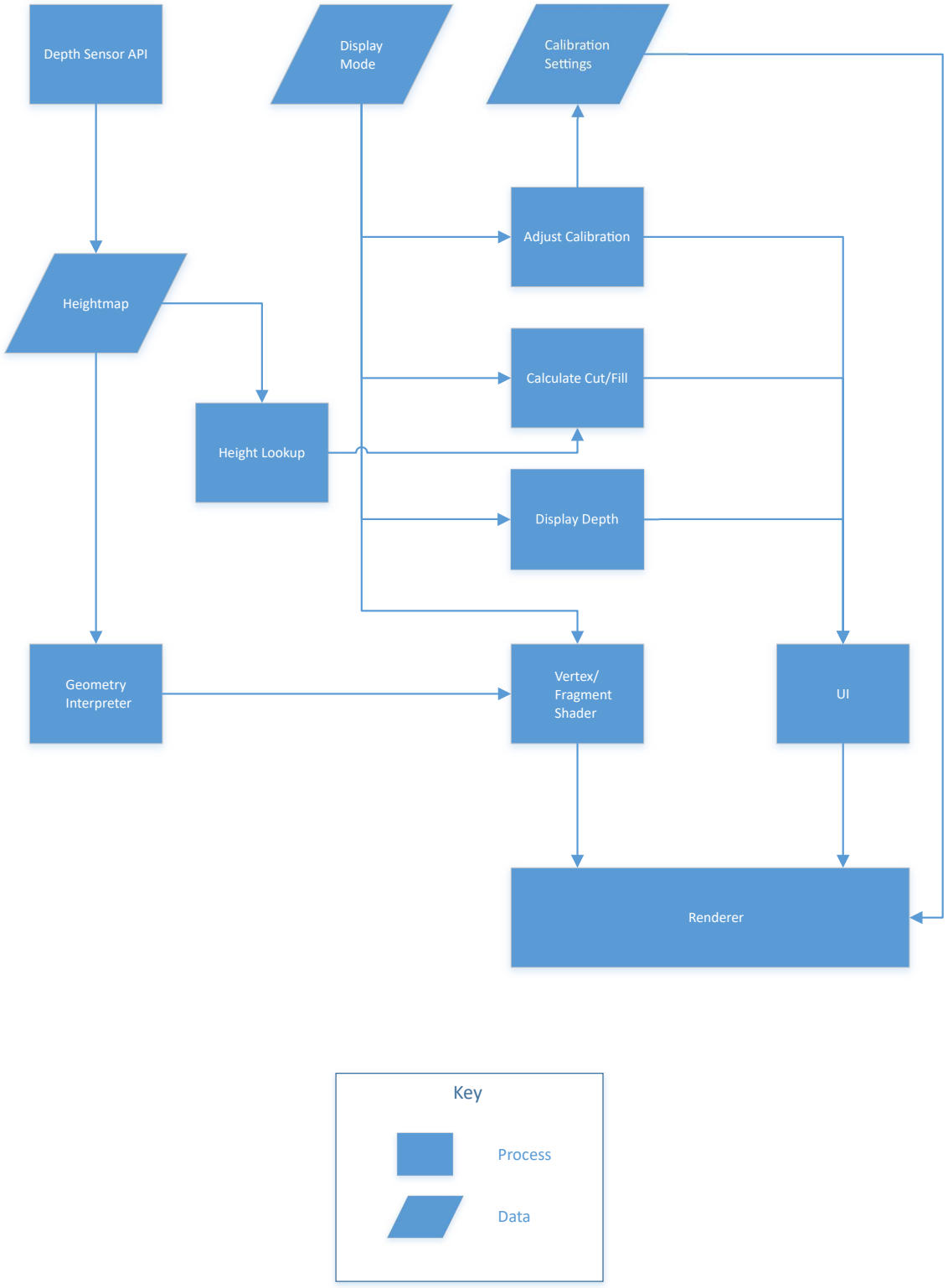

Fig. 2. System Architecture

3.1.1 Depth Sensor API

The Kinect's API allows for the Kinect sensor to interface with the rest of the sandbox system. It provides access to the sensor's depth image data, raw infrared image, full-color image, and has built-in body tracking functionality [1]. For

this system, the API's primary functionality will be providing the sensor's depth image data which will be stored as the heightmap.

3.1.2 Geometry Interpreter

The Geometry Interpreter takes the heightmap data from the depth sensor and converts it into a three dimensional mesh. It is responsible for performing any downsampling required to make the mesh renderable in real time.

3.1.3 Height Lookup

The height lookup module is responsible for fulfilling requests for height data at a given point on the terrain mesh. This is accomplished by correlating the point in 3-space to a pixel on the heightmap, and multiplying the resulting greyscale (normalized to 1) value by the maximum height of the terrain.

3.1.4 Adjust Calibration

Adjust Calibration is used to adjust system settings in order to ensure the sensor and projector are properly aligned with each other as well as the surface of the sand. These setting include adjustments in area the depth sensor is looking in and the area covered by the projection. This module continually interacts with the UI as it prompts the user with the setting to be adjusted and then stores these settings in the Calibration Settings.

3.1.5 Calculate Cut/Fill

Cut and fill refers to the process of creating a level path where a road or highway can be built. The cut or fill of a road will be calculated by determining the height of a part of the sand where the road should be, then displaying a colored view of the road. For example, if the road is below a certain height, the it's displayed using a shade of blue. If it's above a certain height, the road is displayed using a shade of red.

3.1.6 Display Depth

Display depth is used to visualize the height or altitude of an area. If a part of the sand is above a certain height, then it's displayed using a shade of red. If the sand is below a certain height, then is displayed using a shade of blue. The intensity of the color depends on how far the height of the sand is from a predetermined height.

3.1.7 Vertex/Fragment Shader

This program utilizes two pairs of vertex and fragment shaders written in the OpenGL Shader Language (GLSL). The first is responsible for coloring the terrain mesh in order to visually represent height information. The second is used to color road segments to represent cut and fill data. The visual appearance of these shaders will change depending on the current display mode.

3.1.8 User Interface

The user interface allows the user to manipulate the design of the road or change the display mode. The user will interface with the software via a standard computer terminal with a mouse and keyboard. The user can also manipulate the sand which will change the topography outputted by the software.

3.1.9 Renderer

The rendering subsystem is handled by the Unity engine, and is accessed indirectly through shaders and Unity's UI system.

3.2 Design Rationale

Since the software is being built using the Unity Engine, some of our design choices such as the discrete UI subsystem and choice of shader language are limited to what Unity supports. Use of a bitmap image to represent height is determined by the output of the depth sensor. The decision to separate different display modes into distinct modules allows for extensibility should another developer wish to add additional modes.

4 Physical Architecture

4.1 Architectural Design

Figure 3 shows the physical design of our AR sandbox. The sandbox is a mobile box filled with sand and has an arm with a projector and depth sensor connected to an adjacent computer terminal. The components related to the physical architecture of the sandbox are described below.

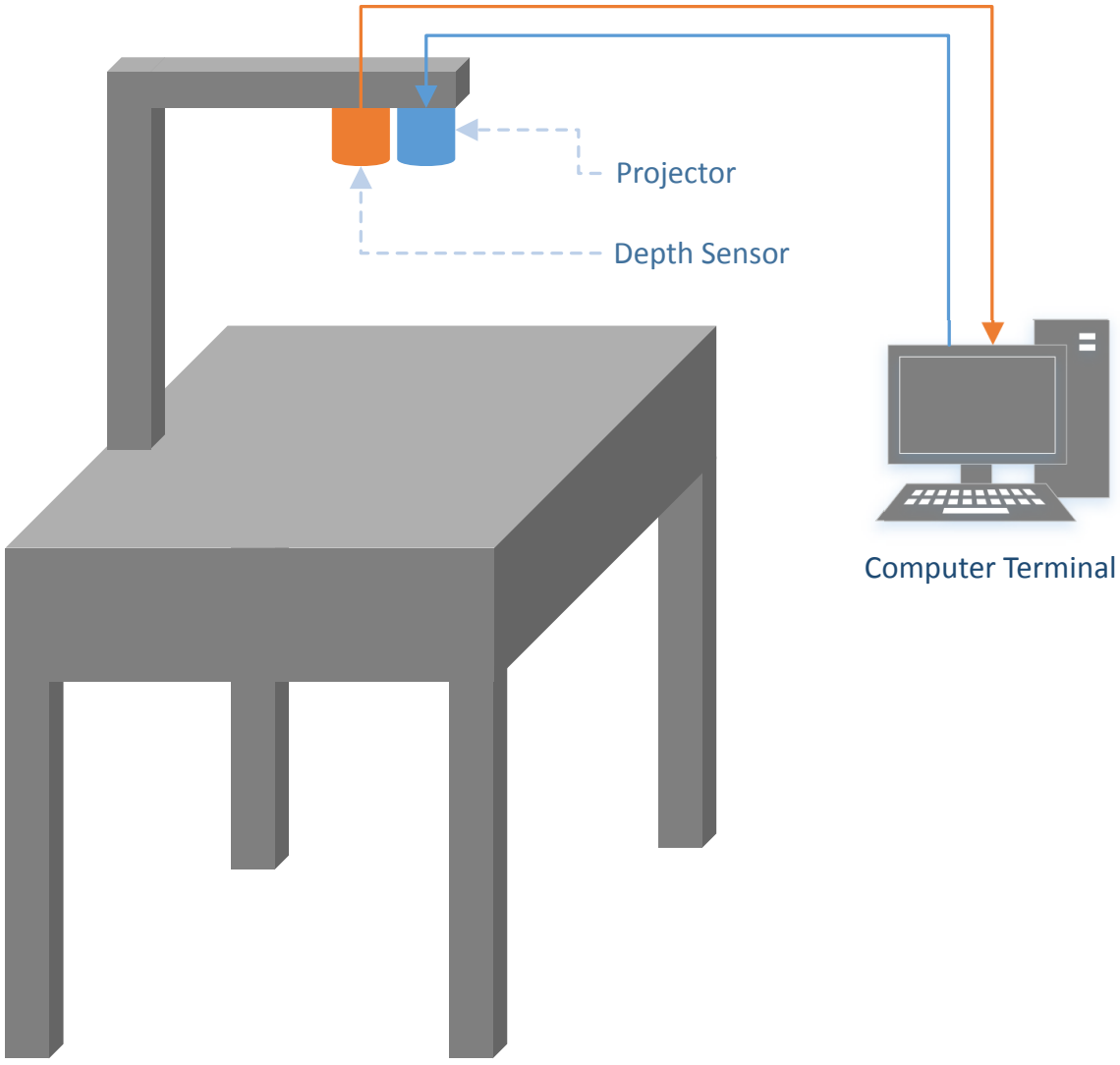

Fig. 3. Physical Architecture

4.1.1 Depth Sensor

The depth sensor that will be used is a Kinect Sensor by Microsoft. The Kinect comes equipped with a 1280x960 full-color camera, and a Infra-Red light emitter and sensor along with an array of microphones and a tilt motor. For this project's purposes, the Kinect's IR sensor and emitter will be primarily used and will allow us to capture a depth image. This Infra-Red system offers a workable range of 0.8m to 3.5m from the target with a data precision of 1cm [2].

4.1.2 Projector

The Projector that will be used is a AAXA M5 Mini Portable Business Projector. This 6in X 6in X 1.8in box only weighs 1.9 pounds and offers a 1280x800p picture at 900 lumens. Allowing for a screen up to 150in, this projector allows for a very bright and clear image for its size.

4.1.3 Computer Terminal

To power the AR sandbox, we will be using an HP Z240 full tower desktop computer with 16 GB of RAM, a 1TB spinning disk, and an Intel core i7 processor with a base speed of 4.2 GHz. We will install an Nvidia GTX 10XX series graphics card to handle rendering.

4.2 Design Rationale

When choosing the Kinect, the most important factors were the accuracy the sensor and its API. The Kinect is one of the most accurate depth sensors on the common market and the API created by Microsoft has been created specifically for developers to use when utilizing the sensor for their applications.

The most important factors of the projector are its size/weight and its brightness. Because of the way the projector is going to be mounted above the box, a smaller and light projector was necessary for proper stability. With the majority of projectors at this size providing a brightness of 50-100 lumens, the 900 lumens is phenomenal.

In regards to the computer terminal, it is necessary to use a full tower desktop since a majority of graphics cards are too big to fit in a small form factor machine, or lack the hardware to mount in such a computer. Using an Nvidia GTX graphics card will give us the best results for rendering real time graphics at the lowest cost when compared to a similar workstation graphics card.

5 Data Design

This section discusses the specifications for the data that will be utilized by the system. The data types and methods for data storage are covered.

5.1 Data Description

The main piece of data utilized by this system is the height data received by the depth sensor API. This data is represented both as a three dimensional mesh and as a greyscale bitmap image. This is because the mesh must be downsampled in order to render efficiently, which, if it were the sole source of height data, would decrease the accuracy of any calculations. The benefit of coupling the mesh to a heightmap is twofold: first, the original data accuracy is maintained, while still reaping the benefits of improved rendering, and secondly, obtaining the height at a specific point can be made much faster since this information can be obtained through a simple texture lookup rather than through mathematical calculation.

5.2 Data Dictionary

5.2.1 Heightmap

The heightmap is a greyscale bitmap image, in which each pixel represents a relative height. Each pixel is represented with 8 bits, yielding 256 shades of grey. In order to correlate a pixel to a y-value in 3-space, its value must be multiplied by a constant maximum terrain height. This is because the heightmap represents the *average* height, where the lowest point is represented by a value of 0, and the highest point is represented by a value of 256.

5.2.2 Terrain Mesh

The terrain mesh is generated by the geometry interpreter, which converts the heightmap into a three dimensional mesh. This mesh is what is rendered by the game engine and what is colored by the shader.

REFERENCES

[1] "Kinect api overview." [Online]. Available: https://msdn.microsoft.com/en-us/library/dn782033.aspx

[2] M. R. Andersen, T. Jensen, and P. Lisouski, "Kinect depth sensor evaluation for computer vision applications," 2012. [Online]. Available: https://pdfs.semanticscholar.org/0338/87697677f41b3c2cc72872ed1a09cc49d649.pdf