ELEC 4309 Senior Design I
Augmented Reality Sandbox
Software Requirements Document
Revision: 00
Revised: 2017-DEC-02

## Definitions:

- CSV: a data format which is specified by comma separated values.
- Operating System: the base system running on the platform that allows users interface (Windows or Linux).
- Firmware: an embedded program on a platform that will not change, written in C or Python or both.
- Software: the program that is executed on the platform, written in C or Python or both.
- Hardware: devices that are controlled by software and interact with the physical world.
- RTOS: Real time operating system.
- Augmented Reality: is a live direct or indirect view of a physical, real-world environment whose elements are "augmented" by computer generated or extracted real-world sensory input.
- Media: the material that is used to partially fill the volume of the enclosure.
- BSP: Board Support Package

## Problem Statement:

- To develop software that can capture topographical data by use of an RGB+D camera in real time.

## Project Goals:

- To control various pieces of modern hardware using Python.
- To apply machine learning & computer vision concepts by exploiting the hardware/software interface of a host platform to analyze a system in real time.
- To aggregate data and display concurrently in a real time system.
- To develop a system whose latency is below the detectable threshold for human eyes, making it appear as if changes are happening simultaneously (I.E. augmented reality).

## Constraints/Restrictions for Simplicity:

The following is a list of assumptions or constraints that we are using to implement our software in a controlled environment:

- Camera will be fixed and mounted above/over a region that can be manually changed (media) to show efficacy
- Media will be easy to modify and shape to show the changing contours in real time, simulating panning of a region
- There will be a secondary process to show the processed data, which will be displaying topographical contour lines on region from a projector mounted above/over the region
- Only the topographical data collected from within the sandbox will be analyzed, everything outside of this region will be ignored. This will either be implemented in software or by the placement of the hardware.

## General Operation:

This device is an augmented reality sandbox used to demonstrate the capabilities of real time topographical mapping of a static environment captured by a camera. The structure will have a large internal volume that is partially filled with a fine sand-like material that is free from any/all debris. The structure will be supported by a mechanically sound “cart” with 4 wheels for easy moving of the entire setup (two caster, two rigid bi-directional). Above the sandbox will be a cantilevered structure that supports both a RGB+D camera and a digital projector both of which are fixed to the structure. Some minor built in adjustments may be implemented to allow for easier troubleshooting of the system during development but will be fixed and rigid for demonstrations. Cabling will be run from the sandbox to an adjacent laptop computer which will be running the software controlling the system. Once the system is powered on, the camera will take a depth measurement of the entire sandbox (z-axis height of the sand at all locations contained within the structure only, the program will ignore any data retrieved from outside of this area).

The data will then be parsed into levels of similar height which will be determined from the z-axis step size for grouping of similar data. The data will then be assigned a color based on its groupings and a line segment will be drawn linearly between each of the data points in the same group of that particular color. This will occur for all groupings of data for each color (red to blue). Once all the data has been parsed, the final contour image will be projected back down onto the media in the structure. This process will be in a loop routine and occur over and over again until user input is received to stop. The timing of this loop for processing the data is the most critical aspect since we require that this appear as if it is happening in real time. At any time during this process if the user wishes to capture the current environment (I.E. the current topology of the sand), a keypress can be made on the computer which will save the entire data set to a file. This will then be immediately processed using either Matlab or Python and will 3D plot the data onto the screen which the user can then save. From this point, the saved data set can then be sent over to a connected 3D printer where it will initialize a print of the saved workspace from ABS material. Due to useable printer platform sizes, the 3D print may be scaled to a fraction of the original data taken.

| Requirement Number/Requirement Specification | Requirement Description |
|---|---|
| **1. Hardware** | |
| 1.1 Platform | Platform must support required peripherals (see section X). |
| 1.2 RGB+D Camera | Camera must be capable of taking depth data with a minimum resolution of 1cm. |
| 1.3 Digital Projector | Digital projector must have HDMI connection and have a minimum display resolution of 1080p. |
| 1.4 Cabling | Cabling will be purchased through an approved supplier and will not inhibit the overall data speed transfer rates set by the platform or other hardware. |
| 1.5 3D Printer | 3D printer must be capable of taking in X, Y, Z coordinates or point cloud information and printing in ABS material with a minimum resolution of 0.5 mm. |
| 1.6 RTC | Real time clock will be on the platform to be used as a time stamp for any offloaded or saved data sets taken from the device. |
| **2. Software** | |
| 2.1 Operating System of platform | Operating system will be up to date to manufacturer's current distribution. |
| 2.2 Operating System of 3D Printer | 3D printer will be up to date to manufacturers approved firmware version |
| 2.3 BSP's | BSP's will be used to hasten development time if they are currently supported by the manufacturer. |
| 2.4 3D Plotting Software | Matlab or Python will be used to plot the changes of the sandbox at set intervals determined in the software. |
| 2.5 Software Include Files | Any utilized libraries will be declared at the beginning of the program. For Python, this will be in the form of *import x as y*. For C, this will be in the form of *#include xyz.h.* |
| 2.6 Header Files | Header files are allowed but must have a descriptive header block at the beginning of the header which describes the following: header name, date of creation, author, and a verbose explanation of use. |
| 2.7 Functions | Functions are allowed and highly encouraged. Every function must have a descriptive function block preceding the function itself and must at the minimum contain the following information: function name, date of creation, author, and a verbose explanation of use. All functions must have a defined function prototype which must be at the top of the main program, below the included files. |
| **3. Language** | |
| 3.1 Embedded Software Language | All software will be written in either C or Python or a combination of the two. |
| **4. Protocols** | |
| 4.1 HDMI | HDMI will be used to display the information from the platform to the projector and onto the surface. |
| 4.2 USB 3.0 | USB 3.0 will be used to transfer data from the RGB+D camera to the platform |

| | |
|---|---|
| 4.3 SPI | SPI will be used if required to communicate to peripherals through full duplex. |
| 4.4 I2C | I2C can be used as an alternative solution if there are a large number of hardware that are more easily addressable in this fashion. |
| **5. Platform** | |
| 5.1 Power | Platform will have a power supply sufficient to power itself and must be able to plug into AC mains 110-125V. |
| 5.2 Peripherals | The platform must have the following peripherals: HDMI, USB 3.0, SPI & I2C. |
| 5.3 Performance | The platform must have at least one core. |
| 5.4 Processor Speed | The platform must have a minimum processor speed of 1 GHz. |
| **6. Overall Dimensions** | |
| 6.1 Exterior Dimensions (Prototype) | The overall outside dimensions of the prototype will be 12” x 12” x 6” ± 0.5” (L x W x H). |
| 6.2 Interior Volume (Prototype) | The overall interior volume of the prototype will be no greater than 864 $in^3$ (12” x 12” x 6”). |
| 6.3 Exterior Dimensions of the final design | TBD. |
| 6.4 Interior Volume of final design | TBD. |
| **7. Media** | |
| 7.1 Particle size | Material selected must be finer than 1mm. |
| 7.2 Gradient | Particle size must be fine enough to produce a smooth gradient on the surface. |
| **8. Speeds** | |
| 8.1 Refresh rate | The refresh rate is tied directly to the speed of which we can collect data from the Kinect and display the data with the projector. One cycle of this is our defined refresh rate. |
| 8.2 Program execution time | A program timer will be implemented to determine how fast the program is executing. This will be the quantitative determinant for how to improve the programs speed. Optimization strategies will be developed if required based on that information. |
| 8.3 Latency | There is not a quantitative value for this requirement. This will be more of a UX requirement as the system must update in real time to accomplish the augmented reality aspect. Therefore, there cannot be any detectable lag in the system from the user’s perspective. |
| **9. Power** | |
| 9.1 Main power | Power will be 110-125 VAC provided from mains via a wall outlet. |
| 9.2 Battery power | This revision of the device will not be designed to operate off of any battery power. |
| 9.3 Device Power Supplies | Each of the 3 main pieces of hardware that is required for this project (Laptop, Xbox Kinect, Projector, 3D printer) all have their own provided power supplies which plug directly into a 3-prong wall outlet and provide the required DC voltage and current necessary to adequately power the |

| | |
|---|---|
| | devices. For additional information on the power consumptions of these devices see the power analysis document. |
| **10. Z-axis Step Interval** | |
| 10.1 Z-axis step interval | Will be a #define or similar global variable that can be adjusted in one location and will change everywhere in the program. |
| **11. Topographical line color palate** | |
| 11.1 Topographical line color palate | This will red to blue ranging from the minimum to the maximum and the color transition from one line to the next will be determined by the Z-axis step internal (Requirement #10). |
| **12. Optimization** | |
| 12.1 Program optimization | Techniques such as loop unrolling and the use of intrinsics may be used to speed up the execution time of the program if required or desired for the appearance of real time. |
| **13. 3D Printer** | |
| 13.1 Usable printer area | Due to the constraints of the 3D printer's usable workspace, the rendered print may need to be a scaled down version of the original data set. In this case, a secondary data set will be generated from the original which will be the reduced data necessary to fit on the printer's platform. |
| **14. Cost** | |
| 14.1 Total Cost of the Project | TBD. |
| 14.2 Total investment from the department of electrical engineering | $600. |
| 14.3 Additional Funding | $100 (provided by the chair of the Department of Electrical Engineering). |
| 14.4 Total prototype cost | $958.22 |
| 14.5 Total estimated production costs | TBD. |
| **15. Data** | |
| 15.1 Topographical Data set | Each captured data set will contain all data points prior to being sorted. Each subsequent data captured will over-write the previous data. This will keep the database from "blowing up" in size and filling up digital space. |
| 15.2 Topographical Data set to output file | When the user triggers the dataset to be saved to an output file, this will be time stamped and saved to the working directory. This data is never destroyed. The user could trigger many "save data to output file" and each will be saved uniquely to be reviewed or analyzed at a later time. |
| 15.3 Output file naming convention | When the user triggers the program to save the current dataset to an output file, the filename will be the date and time the user triggered the save event. |
| 15.4 Output file data format | CSV (comma separated values) or .xlxs (Excel file format). |
| **16. User Interaction** | |
| 16.1 Start Program | When the program is initialized, to start the program press the "ENTER" key on the keyboard. |
| 16.2 End Program | To end the program, press the "ESC" key. |
| 16.3 Save data to output file | To save the last data to an output file, press the "S" key. |

| | |
|---|---|
| 16.4 Send data to 3D printer | To send the data to a 3D printer, press the "P" key. |
| 16.5 Send data to be 3D Plotted | To print the topographical map of the last data set, press the "M" key. |
| **17. Software Best Practices** | |
| 17.1 Function naming convention | Functions will be named in a case sensitive manner in which the first word is lowercase and the following word first letter is uppercase. There will be no underscores used in function naming. Functions must be named such that their function is obvious.<br>Example of correct syntax: void exampleFunction(void)<br>Examples of incorrect syntax: void example_function(void)<br>void ExampleFunction(void) |
| 17.2 Variable naming convention | Functions will be named in a case sensitive manner in which the first word is lowercase and the following word first letter is uppercase. There will be no underscores used in function naming. Variables must be named such that it is easy to understand what purpose they serve. Ambiguous names are not allowed.<br>Example of correct syntax: int testVariable<br>Examples of incorrect syntax: int val1<br>int maximum_value_found |
| 17.3 Function return values | If a function is going to return a numeric value for success or failure they will be done in the following way:<br>For a success: return 0<br>RETURN_SUCCESS<br>For a failure: return -1<br>RETURN_FAILURE |
| 17.4 Main program organization | The main program will be in the following form (from top to bottom):<br>Title block<br>Include Files<br>Preprocessor Directives<br>Main function<br>Supplemental Functions |
| 17.5 If/then statements | If then statements are not allowed to have a full pass through. This means that if an if/then statement is used that the full if/then/else must be implemented to catch any issues that could arise from the conditions not being met in the if/then.<br>Example:<br>`int i = 5;`<br>`if (i == 1)`<br>`{`<br>`        printf("IF CONDITION\n");`<br>`}`<br>`elseif (i == 2)`<br>`{`<br>`        printf("ELSE CONDITION\n");`<br>`}`<br>`else` |

| | |
|---|---|
| | {<br>        printf("CATCH CASE CONDITION\n");<br>} |
| 17.6 Case statements | Case statements are allowed. |
| 17.7 While loops | While loops must be allowed to have one or more exit conditions. This means that the main program cannot run in a while(1) loop. |
| 17.8 For loops | For loops are allowed but should be used with care as this program is time sensitive and these loops do not execute in one clock cycle. |
| 17.9 Conditional expressions | Conditional expressions must have verbose parenthesis used to make it perfectly clear what is being compared or evaluated. |
| 17.10 Ternary operator (?) | Ternary operators are not allowed. |
| 17.11 Arrays | Arrays will be used to store the topographical data, it is recommended that numpy be used as it has some built in acceleration aspects already in place for optimization. All arrays must be correctly allocated for their size. |
| 17.12 Memory Allocation | Memory allocation must be done for this application as the amount of data being collected will be of a fixed size and will be updated frequently. The same block of memory will be used to store the topographical data on repeat executions. This will keep the program from wasting the computers resources. |
| 17.13 Print statements | Print statements are only allowed for debugging. They will not be allowed to be included in the final production version of the software as they will negatively affect the programs execution time. If output is required during normal operation, use an output file. |
| 17.14 Output files | Output files have two basic purposes: data storage and debugging. Data storage is a requirement of the software and must be done to the specified format. All output files must be closed after the information has been written. |
| 17.15 Condensed Code | Clever code is not syntactically incorrect, but as it does not aid in any development across multiple users, it is not allowed. Verbose, easy to follow code is required which will allow all programmers to easily understand an modify any sections of code. |
| 17.16 Program Title Block | Regardless of C or Python, the program header block will take the following form and will be in a block comment:<br>Program Name:<br>Program Author:<br>Date of Creation:<br>Revision:<br>Revision Updates:<br>Program Description: |
| 17.17 Program Function Block | Regardless of C or Python, the program function block will take the following form and will be in a comment block:<br>Function Name:<br>Function Author:<br>Date of Creation:<br>Function Parameters: |

| | |
|---|---|
| | Return Parameters:<br>Function Description: |
| 17.18 Program Header Block | Headers will be used for functions that are frequently used and therefore will contain some repeated information as the function block. The following header block is for C only and will take the following form and will be in a comment block:<br>Header Name:<br>Header Author:<br>Date of Creation:<br><br>Function Name:<br>Function Author:<br>Date of Creation:<br>Function Parameters:<br>Return Parameters:<br>Function Description: |
| 17.19 Comment Block | Comment blocks are defined as comments in the program that extend more than one line. This is done in the following way:<br>C:<br>/*************************<br>* This is a multiple line comment<br>**************************/<br>Python:<br>‘’’This is how<br>to do multiple line commenting<br>in Python’’’ |
| 17.20 Single Line Comment | Single line comments will be done in the following way:<br>C: //This is a single line comment in C<br>Python: #This is a single line comment in Python |
| 17.21 White space | White space between lines of code is not a requirement. Excessive space is not allowed, however double spacing your code for easier reading is allowed and up to the programmer’s discretion. |
| 17.22 Visible function/program separation | Frequently it helps to break up the program into visible sections. This is allowed in the following ways:<br>C:<br>//---------------------------------------------<br>Python:<br>#--------------------------------------------- |
| 17.23 Brackets | Brackets are used and required explicitly for all functions, loops, if statements, etc. They will be done in the following ways:<br>void exampleFunction(void)<br>{<br>        //Function contents<br>} |

| | |
|---|---|
| | `If (variable == 1)`<br>`{`<br>`        //Do this`<br>`}`<br>`elseif (variable == 2)`<br>`{`<br>`        //Or this`<br>`}`<br>`else`<br>`{`<br>`        //Then this`<br>`}`<br><br>The following is an example of what is not allowed:<br>`void exampleFunction(void) {`<br>`        //Function contents`<br>`}` |
| 17.24 Revisioning | The program title block has a section for revision. This is to be used when developing the program. When the program is in its prototype phase and is not functioning as required, the revision used will be alphabetic only. Ex. Revision: A. Once the program reaches the point when the program is functioning as required and changes made are to improve functionality, speed, interfacing, etc., then the revision will be a two-digit numeric only. Ex. Revision: 00. When saving a new revision, update the section that describes what changes were made since the previous revision. Use revisions liberally, as they are a great way to keep a program history of what changes have been done and allows for easy roll back if necessary. |
| 17.25 Multiple Source Files | Multiple source are allowed but must be saved in the same directory as the main file. When using multiple sources, they must be verbosely explained and contain the same title block and description of operation as any other files. In general, unless the program becomes exceedingly long, it is best to make use of header files before using multiple source files. |