Oregon State University
College of Engineering

CS Capstone Software Design Description

November 30, 2017

3D Virtual Reality Painting

Prepared for

EECS

Dr. Kirsten Winters _____ _____
Signature Date

Dr. Mike Bailey _____ _____
Signature Date

Prepared by

Group 66
PolyVox

Chris Bakkom _____ _____
Signature Date

Richard Cunard _____ _____
Signature Date

Braxton Cuneo _____ _____
Signature Date

Abstract

The following is the design document for the program PolyVox, a virtual reality based three-dimensional art program. This document outlines the goals, requirements, and design decisions for the development of the program. These factors are based on the needs expressed by the project stakeholders, which are specified and addressed within the document.

CONTENTS

1 INTRODUCTION

1.1 Purpose

This Software Design Description (SDD) specifies the design plan for the VR based 3D art program PolyVox. This document expands upon the content of the PolyVox Software Requirements Specifications (SRS), and specifies how the elements of the SRS are incorporated into the program. Additionally, this document addresses the design concerns of the project stakeholders, and how these concerns will be handled in the design.

1.2 Scope

This document describes the structure of PolyVox and the implementation of its individual components. This document assumes that any reader is already familiar with the PolyVox SRS and the project as a whole. This document also includes details of elements as of yet not detailed in previous documentation, and provides context as necessary.

1.3 Intended Audience

This document details the technical design of the PolyVox application, and, as such, is intended for readers with knowledge of programming and software development methods.

2 GLOSSARY

Adding geometry	Altering the environment such that it contains additional geometry without the exclusion of any geometry extant immediately prior to this alteration.
AR	Augmented Reality; The practice of producing an overlay interface that displays and dynamically interacts with the physical world around the user.
Attribute	A class of value representable as an integer or floating point number
Attribute datum	A specific instance of an attribute
Color	A set of four attribute data, all represented by a floating point value, corresponding to red, green, and blue color channels, as well as an alpha (transparency) channel.
CPU	Central Processing Unit; The component of the computer that runs and operates the programs being run, and performs the necessary computations to do so.
Environment	The state of every instance of a geometric element explicitly represented by the system during a given instant. This includes the transformation of each particular geometric element, the size of each geometric element along each of its three axes, and the states of all attribute data present in each voxel of each geometric element.
Framerate	The measure of time between each temporally consecutive instance of a new image rendered by the system being displayed to the user.
Grid	Specifically, a finite three dimensional cartesian regular grid which is oriented relative to a three-dimensional origin by a transformation representable by a quaternion.

Geometric element	A set of all voxels contained within a grid.
Geometry	One or more instances of a geometric element.
GPU	Graphics Processing Unit; a processor specifically designed to perform the computations used to render three-dimensional computer graphics.
HMD	Head Mounted Display; A wearable display system placed over the users head and face with a display placed directly in front of the users eyes. The HMD also tracks the users head movements and sends movement data to the program.
Interchange	The program native to the CPU which is in charge of managing user interaction with the voxel state via Yggdrasil as well as maintaining CPU-side resources.
Latency	The measure of time between a user manipulating the devices they are interfacing with and the effects of these manipulations being conveyed by the output of these interfacing devices.
Modifying geometry	Altering the environment such that the state of attribute data contained by geometry in the environment has been altered without the addition or removal of geometry.
Motion Control	The practice of manipulating devices which measure and convey to a computer their position and orientation relative to a reference point.
Removing geometry	Altering the environment such that it excludes geometry without the inclusion of any geometry not extant immediately prior to this alteration.
Resolution	The number of columns and rows of pixels used in a display. In the case of VR headset, resolution is the effective resolution experienced by one eye using the headset.
Virtual Reality	The practice of placing a display in front of each eye of an individual and displaying images for each eye which, through binocular vision, convey, to the individual looking into said displays, a scene with the illusion of depth.
Voxel	An element of a grid with an associated set of attribute data including at least one instance of color.
Yggdrasil	A collection of GPU programs which collectively manage the voxel state, including updating the data represented within and maintaining the associated resources on the GPU. These programs are called by the interchange.

2.1 References

REFERENCES

[1] "Htc vive reccomended hardware specifications," https://www.vive.com/us/ready, accessed: 2017-10-25.

[2] "Google tilt brush," https://www.tiltbrush.com, accessed: 2017-10-25.

[3] "Cavepainting," http://graphics.cs.brown.edu/research/sciviz/cavepainting/cavepainting.html, accessed: 2017-10-25.

[4] "Unreal engine — blog," https://www.unrealengine.com/en-US/blog, accessed: 2017-11-01.

[5] "Unity - game engine," https://unity3d.com/, accessed: 2017-11-01.

3 BODY

3.1 Stakeholders

3.1.1 Dr. Mike Bailey

Dr. Bailey was brought onto the project after being approached by Dr. Kirsten Winters, who inquired about designing a three-dimensional art program in VR. Dr. Baileys primary stake in the project is the development of VR and graphical technology, with less concern for specific feature sets. His primary design concern is development of a stable and sufficiently robust graphics engine compatible with VR.

3.1.2 Dr. Kirsten Winters

Dr. Winters is responsible for the inception of the program, and brought the concept of a 3D art program to Dr. Mike Bailey. Her initial vision of the project is, by intention, rather vague. As such, her main design concerns are higher-level functionality, such as the general ability to create three-dimensional geometry using motion controls, as well as program optimization (in order to ensure a comfortable user experience).

3.1.3 Intel

In recent years, Intel has been supporting the development of VR and AR applications, going as far as forming a VR-centric department, the Intel VR Center of Excellence, in an effort to push VR into mainstream popularity. With this goal in mind, Intel has agreed to aid the project and supply resources, with the intent of producing a viable VR product. Given these factors, Intels primary design concerns are focused around ease of use for users. These include program stability, accuracy of motion controls, functioning user interface, and a sufficient feature set, as well as comfortable user experience.

3.1.4 Development Team

In addition to the project clients, the development team has a stake in the success of the project. All team members (in addition to all clients and the OSU college of EECS) will receive equal, non-exclusive rights to the ownership of the program. As such the development team has an interest in developing a powerful and functional toolset for the program. With that in mind, the development teams primary design concerns are building a rich feature set and maintaining program stability.

3.2 Design Views

3.2.1 Interactions between the hardware and the Interchange

3.2.2 Architecture of the User Interface

3.2.3 Architecture of the Voxel State

3.2.4 Architecture of the graphics engine (Yggdrasil)

3.2.5 Architecture of the Interchange

The interchange will be the primary point of communication and manager for the disparate elements of the program. The interchange must be able to receive information supplied via the user interface and interface hardware and relay the corresponding instruction to the Yggdrasil engine to properly render the scene, and is directly responsible for directing state transition of all other components.

The Interchange must be capable of properly coordinating all other elements of the system as a whole, and is a central element in executing commands supplied via user interactions, implementing program features, and directing the graphics engines rendering operations. As such it is relevant to design concerns of all stakeholders involved in the project. In particular, it is central to the operation of the user interface and motion controls, from which it will receive user commands, and the operation of the graphics engine, which it will send commands to. As the Interchange will be the point at which the programs toolset is constructed, it will also dictate the programs available features.

Design Elements:

Unity Engine

The interchange will be built in and run on the Unity Game Engine. Unity has its own proprietary rendering and modeling systems, as well as native compatibility with motion tracking systems and duel rendering used in VR. Additionally, Unity has native scripting compatibility and will serve as the platform for developing the program tools and features.

C#

C# is one of the most frequently used languages for scripting in game engines, and is natively compatible with Unity. Most features and tools for the program will be constructed using C#.

When in operation, the Interchange will receive positional information from the user via both the HMD and VR motion controls in the form of four-vector positional coordinates. Using Unitys native VR drivers, the interchange will translate these coordinates into a position relative to the voxel state.

Design:

Additionally, the data received from the user may or may not include a user-inputted command via a button press. The button pressed will be sent to a function, which is also passed the current state of the UI (such as what brush is selected, or what menu the user currently has open). The function will process the user command to determine any possible UI state changes, as well as any changes to the voxel state the user command will perform (such as creating or destroying a voxel) based on whatever tool or UI element is currently being operated.

Any changes in the state are returned by the function as a set of commands to the graphics engine. The graphics engine will then process the commands from the Interchange. Before the render is sent to the HMD, the graphics engine will return a flag that will determine if the Interchange needs to perform additional actions, such as sending a command to the GPU to allocate additional memory. If so, the Interchange will send the appropriate commands, and the graphics engine will reattempt the render. This repeats until the flag sent from the GPU is null.

While the operations of the interchange are primarily just in service to other elements of the program, they are still vitally necessary to the systems operation. The Interchange effectively acts as the driver for the graphics engine and the motion control system, and is needed in order to develop a working feature set and comfortable user interface.

[1][2][3][4][5]