# EE267 Final Project Proposal

Mariana Neubauer
mhneub@stanford.edu

Shannon Kao
kaos@stanford.edu

## 1 Goal

In this project we hope to achieve reasonably accurate positional tracking in order to make a simple immersive drawing application for the headset. We plan to put an IMU on a rod and have it act as a marker that will draw a line on 3D space (or on a 2D canvas that is placed in a 3D environment). If the accelerometer data cannot provide accurate enough positional tracking to make the drawing experience enjoyable, we will include a webcam on the headset and use Visual SLAM to help localize the the rod.

## 2 Positional Tracking

We will first implement accelerometer-based positional tracking, and evaluate the accuracy of the system. We will then try camera-based or depth-based methods to augment or replace the initial IMU-based tracking.

### 2.1 Accelerometer-based

Our initial attempt at positional tracking will be based on the accelerometer data obtained from our IMU. We will first use the orientation calculated in assignment 5 to remove the gravitational component of the acceleration. Then the acceleration data can be used to calculate velocity, which in turn will be used to calculate position offsets. At each stage of this double integration, we will experiment with different signal filters in order to remove unnecessary noise and bias.

### 2.2 Camera-based

We also plan on exploring camera-based positional tracking methods. We hope to attach a camera to the front of the HMD, and place a marker on the end of our brush rod. The program, knowing the color and size of this marker, should be able to track its position. We will utilize a visual SLAM library (eg OpenSLAM, ScaViSLAM) in order to do this image-based tracking. Additionally, we may consider augmenting this camera data with depth sensor information, if we can locate a depth camera.

## 3 Immersive Environment

The main purpose of our interactive and immersive environment is to visually display the positional tracking of the rod by rendering the path we compute from the accelerometer and camera data. We have a series of milestones for the development of the environment.

1. Model the position of rod as a marker or paintbrush. We can make the model in Maya or get it online. Render the path the rod takes as a line in 3D space.
2. Turn on and off the ability to draw a line using a gesture.
3. Switch the colors that are drawn by positioning or pointing the rod at a color palette located in the immersive 3D environment.
4. Instead of rendering a line in 3D, render a 2D line on a canvas located in the 3D immersive environment. The extra challenge here is we must decide if the rod is touching the canvas in order to draw the line. If we achieve milestone 4 we do not need milestone 2.

The achievement of these milestones depends on the success of our positional tracking. Milestone 1 is the minimum we would like to achieve and milestone 4 is the stretch goal.

## 4 Rendering

The final step of the project will be to determine how to display a user's paint strokes.

### 4.1 2D Canvas

Rendering paint on a 2D canvas is relatively straightforward, assuming we have the correct positional tracking of the brush. We can, on the CPU side, create a texture image representing the 2D canvas. Then, we can simply project the 3D tracked point into the 2D pixel space, and color the appropriate pixels in the texture. This texture will be mapped to the 3D canvas object, which will display the painting. Different brush stroke or paint effects can be applied once we have the 2D line that the user draws on the canvas.

### 4.2 3D Canvas

Rendering paint in 3D space is a significantly more difficult. Each stroke will require 3D geometry in order to render it in our environment. We draw inspiration from Googles TiltBrush regarding the display and rendering of 3D paint strokes. Our first step will be to simply render the stroke as a line. The positional tracking returns a series of points in 3D space, and we will simply connect these points to represent a brush stroke. Then, to render actual paint-like strokes, we hope to try a couple different approaches. Drawing flat rectangles and texturing them with a brush stroke image is one option, and we would have to determine the orientation of the stroke. We might also consider a low-polygon version, where users can paint squares of color in a Minecraft-like style.