3D Paint

By
Yuval Shildan, Ran Mansoor, Shlomit Sibony

Supervisors
Yaron Honen, Boaz Sterenfeld
Introduction

• We have created a Virtual Reality 3D Paint using Unity engine with C# scripting.

• The equipment we used includes Manus-VR gloves and HTC Vive headset and trackers.
Introduction

Since the Manus VR gloves is a new product we wanted to explore its capabilities and create an easy to use 3D-paint platform.
Technologies

- Unity, C# scripting
- HTC Vive - Headset and trackers
- Manus VR gloves
Project Objectives

We had 3 main goals:

1. Smooth and interactive painting with both gloves
2. Grabbable paintings
3. Intuitive usage of the product
Solution Procedure
Interaction with the gloves

● Since the gloves are a pretty new product, its documentation and open source Q&A can’t be easily found.

● However, Manus-VR provides us with its SDK implementation written in C#, thus we dived into the given code and explored some specific features.
Interaction with the gloves

The important instructions for using the gloves are described in our project report, including:

- fingers joints open/close values
  - an array with [0,1] values
- hands state (e.g. open, close)
  - useful enum
Unity provides us with several ways for implementing the painting feature.

- Line Renderer
  - a component that takes an array of 3D points and draws a straight line between each one.
Painting Implementation

Line Renderer

- **Pros**
  - easy to implement and use

- **Cons**
  - a game object is not created which affects the ability to edit its transform.
Painting Implementation

Creating a game object in the gloves' position for each frame.

- Cons:
  - each painting is composed of thousands of game objects which results in a poor performance.
  - the complex design makes it barely impossible to move the painting smoothly and effectively.
Building a mesh for each painting

- Mesh consists of triangles arranged in 3D space to make the impression of a solid object.
- Pros
  - only one game object is generated per painting, thus it can be moved easily by changing its transform.
Building the mesh

- Given the previous (s) and current (e) points of the user’s finger
  - \( n = s \times e \)
  - \( l = n \times (s - e) \)
- The two new edges of the mesh are:
  - \( s \)
  - \( s + l \times w \), where \( w \) is the custom width.
Painting Implementation

Building the mesh

- Each frame we
  - add the previous calculation to the mesh
  - update the value of the last point

- This technique computes the previous vector due to the next position of the finger
  - it makes the rotation of the painting smoother
Future Work

- Communication online – paint with friends
- Printing and saving the paintings
- Choosing colors
Demo

- Pictures
- Video
Demo

- Pictures
- Video