**Touch to Win**

Game Design Document

**Goal**

Build a JavaScript game around the LeapMotion sensor interacting with WebGL (specifically using Babylon.js this time). This final project will demonstrate what I have learned in this Advanced Game class and in my CS courses. The specific idea behind this game is to give me experience using the LeapMotion to control JavaScript (specifically WebGL content), and more generally to explore how the LeapMotion sensor could more readily be integrated into future student’s game projects.

**Overview**

Use floating canvas windows in a scene to display content and act as “buttons”. Use a 3D model of a hand “rigged” and following the motion of my real hand, to touch the canvas windows. I will use WebGL to display the floating canvas windows. I will use a pre-built hand model, built in ThreeJS, that connects easily with my LeapMotion for the hand. Any other game logic I write will likely be in vanilla JavaScript. Needless to mention, the WebGL content, and therefore the game, will be displayed in an html page that will render on the browser.

**Game Description:**

Each game starts with a set of canvas window’s (each a unique color). Each level, starts with the colored canvases and flashes different canvases in a certain order. The user needs to use their hand to “touch” the canvases in the correct order and under a certain time limit.

**Game Assets:**

**Hand Model:** Using a pre-built/rigged hand model built in ThreeJS (see link: <https://github.com/leapmotion/leapjs-rigged-hand> : Apache-2.0 license)

**Floating Canvas Windows:** Using code from a WebGL example on Three.js website (see link: <https://threejs.org/examples/#webgl_multiple_canvases_grid> : MIT license)

**UI:**

**Game Rules:** Rules for the game and start menu

**Game Play:** A set of floating canvases with an “infinite”-like space behind the canvases and the 3D hand model.

**Game Object Mechanics:**

**Hand Model:** The LeapMotion SDK does its best to track every finger and the position of your lower arm. Thus, the hand model should fairly-realistically map to whatever you are doing with you hand.

**Floating Canvas Windows:** Each canvas lights up (glow from behind) individually whenever the pattern is being shown and/or the user touches that canvas.

**States and Transitions:**

Launching my game will direct you to the start screen. This will provide a list of instructions for game play. Then, you can push start to start the game. Once you have started the game, the first thing you see will be a sequence of canvases flashing on a off in some pre-defined/random order. Once this sequence of flashing is over, it is your turn, as the player, to use your hand to touch the correct canvases/colors in the correct order and the correct number of times. You will have a time limit, which will define when you have completed you attempt at copying the sequence. As you complete each level, you will be given the opportunity to be take to the next levels, which will be successfully harder (increasing in both speed and complexity of the sequence). Once you are done playing the game, you can push quit on the screen at the end of any level.

**Milestone plan**

**Control hand model with LeapMotion:**

**Import floating canvases into “room” with hand model:**

**Build interactivity (“touch” feature) between hand model and canvas windows:**

**Build game logic + connect with screens:**

**Tuning:**

Have the canvases float in space in a curved fashion. Score keeping. Customizable canvas positions.