1. Introduction

The purpose of our project is to implement the computation of greedy triangulation using movement recognition software, specifically hand tracking. The project uses p5.js due to the fact that it facilitates the implementation of the program on websites. The idea for the greedy triangulation came from C. Levcopoulos, D. Krznaric's paper [1], which describes this concept in detail and also provides useful information for the implementation itself, in the code. An easier way to understand the base notion of the project is by examining Figure 1, which can be found on Imgur [2]. The GIF explains how, by having a polygon in any shape, it can be split into triangles, which will be turned into rectangles using Heron's Formula [3]. The rectangles are then manipulated into becoming squares with the same area, after which the Pythagorean formula will be used to create a square that will have the same area as the initial polygon.

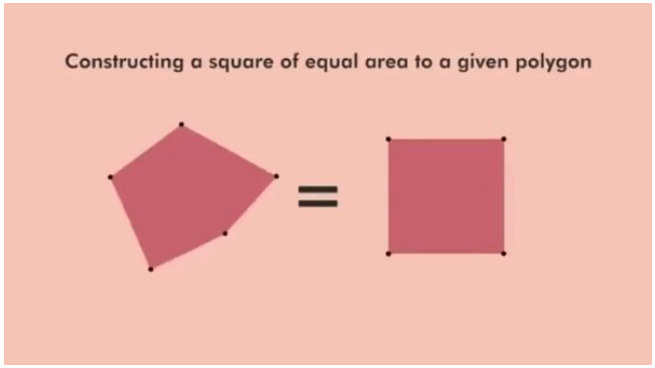

Fig. 1 - Short explanation of the triangulation concept

2. Implementation

A. Visual

In terms of visual elements, the aim was to make the project feel like an actual application. When drawing a polygon, the user can make it as complex as possible, it can be concave or convex, the results will still be accurate. Furthermore, the user can choose from different color schemes.

B. Methods

The way the code works is showcased in Figure 2 (below). The program detects whenever the following events occur: pressing, moving, dragging and releasing the mouse. If the mouse has been pressed, the method will get the coordinates of the respective point and calculate the area of the polygon after multiple points have been clicked. The area is calculated using basic mathematical formulas, as well as using inspiration from [1].

[1] C. Levcopoulos, D. Krznaric. "The greedy triangulation can be computed from the Delaunay triangulation in linear time". 1999. [Online]. Available: https://core.ac.uk/download/pdf/82620126.pdf
[2] Triangulation GIF. [Online]. Available: https://imgur.com/f80NRUP
[3] Herron's Formula. Available: https://www.mathsisfun.com/geometry/herons-formula.html
[4] MediaPipe Hands. Code: https://google.github.io/mediapipe/solutions/hands.html

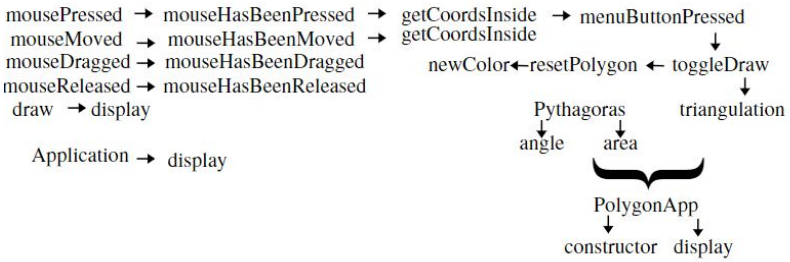

Fig. 2 - Schematic describing the code

C. Hand Tracking

The hand tracking code was provided by MediaPipe [4], which offers Machine Learning models, specifically the hand tracking one used in this project. The program can detect the palm and back of the hand, as well as track the position of each individual section of the fingers (see Figure 3). When testing out the code, we started out by printing the X, Y, Z coordinates, in order to figure out what output we were receiving.

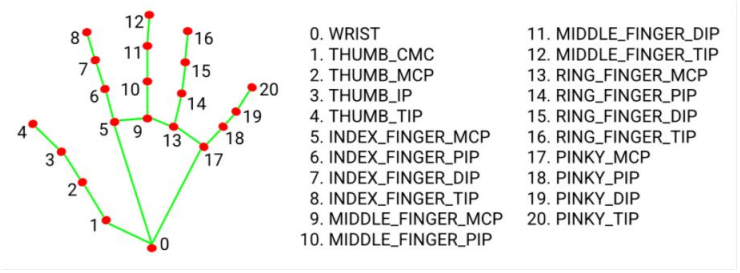

Fig. 3 - The landmarks tracked

The given code has certain parameters that can be modified. For example, the maximum amount of hands that can be detected is 2 in the default code, however, that can be changed. Another parameter that can be changed is the confidence level for detection, as well as tracking (the default settings are `([0.0, 1.0])`). Increasing those values will result in a drop in frames, as well as an increase in efficiency.

[1] C. Levcopoulos, D. Krznaric. “The greedy triangulation can be computed from the Delaunay triangulation in linear time”. 1999. [Online]. Available: https://core.ac.uk/download/pdf/82620126.pdf
[2] Triangulation GIF. [Online]. Available: https://imgur.com/f80NRUP
[3] Herron’s Formula. Available: https://www.mathsisfun.com/geometry/herons-formula.html
[4] MediaPipe Hands. Code: https://google.github.io/mediapipe/solutions/hands.html