Visual Calculus

XiaoZheng Judy Xu, Matthew Beaudouin-Lafon, March Saper

Line Smoothing

If we took the line directly from the user's drawn input, it would be jagged, leading to a wildly changing derivative. To solve this problem, we must smoothen the line.

To do so, we use cubic interpolation. First, we select a small set of points from the input (*pull points*). Between two points, we will draw a cubic function such that its derivative at the pull points matches the derivative of the cubic in the leading and following pairs. The domain of the cubic function is restricted by the two poitns. This is handled by SciPy.

As a result, we get a unique smooth continuous function that we can easily differentiate and integrate. We can also use the pull points to move the curve around in a predictable way.

Cubic interpolation. The green line is the cubic function between points 2 and 3, with slopes at those points matching the slopes of the neighboring cubics. The red lines are also cubic functions, but their domain has been restricted by their defining points.

Differentiation

With the interpolation above, we can have an arbitrarily large density of points. Therefore, we can very simply and precisely get the derivative of the line numerically, by getting the slope between any two consequent point.

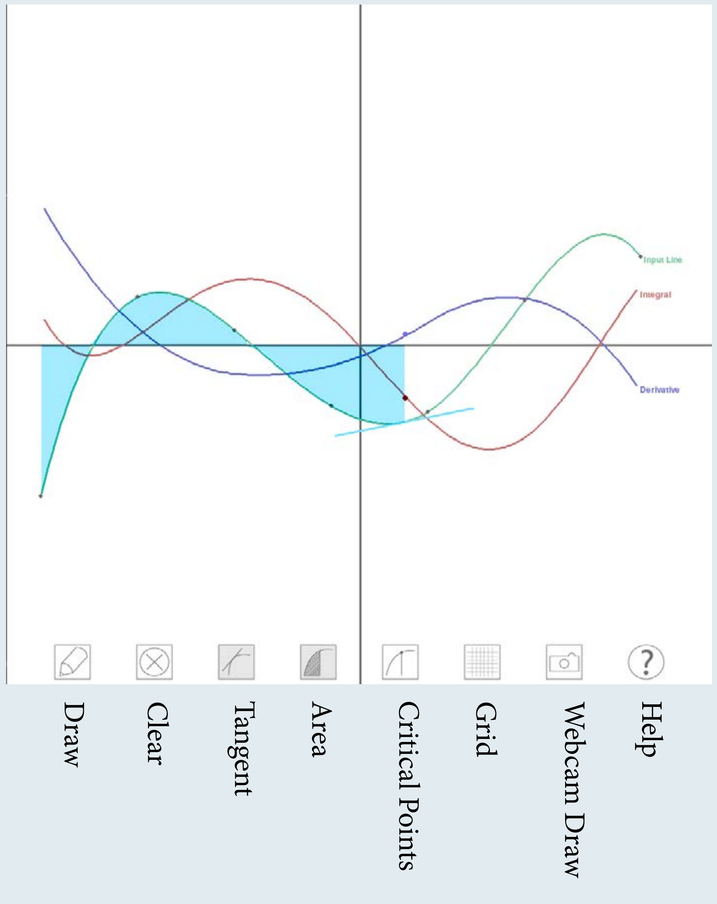
Integration

Similarly to differentiation, we can get the curve's integral using trapezoidal approximation. We find the area between two consecutive points by getting the area of the trapezoid defined by the two points and their projection onto the x-axis.

Introduction

This project aims to help individuals understand and interact with the Fundamental Theorem of Calculus in a purely visual and geometric way. We hope to help learners build and intuition for the relationships among a line and its corresponding derivative and integral through an interactive window where users begin by drawing an input curve. After the curve is drawn, it's corresponding integral and derivative are displayed and users may adjust these lines and observe how they change in relation to one another. Users may also view corresponding tangent lines and areas to connect the functions of derivatives and integrals to their properties as lines. We hope this modules will act as a supplement to traditional textbook explanations and will be a useful too for both teachers and learners.

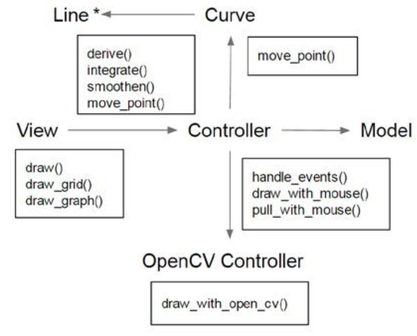
For Example:



Structure

We based the structure of our code on the model-view-controller framework, as shown in the UML diagram below. The Line class contains methods for processing lines (deriving etc.). The Curve class keeps track of the input, integral and derivative Lines in relation to one another.

Our structure could perhaps have been better. In some cases changing things in one class meant changing them in another class as well. Also, the interdependence of classes on one another created ambiguity about which class was performing which action, and caused interestingly long variable names.



UML Diagram.

Design Decisions

Our approach to this project has changed throughout the weeks. We decided that the user would draw a single line and immediately see its derivative/integral, rather then apply a derivative to a line. The latter might lead to the user repeatedly deriving the derivative, which is not what we want to highlight. Our approach focuses on how the line's geometry translates to Calculus.

Originally, we wanted to implement functionality halfway between the typical mouse/keyboard interface and OpenCV. For the sake of completeness, we then decided to craft the experience entirely around OpenCV. However, after thinking about our audience, we realized we should focus on a solid mouse-based interface as it would be the most familiar. This meant moving away from a shortcut-based interaction to having visble and actionable buttons on screen.