My idea is to create an interactive computer algebra system (CAS) *component* that focuses its efforts on augmenting human manipulation of mathematics rather than its automation. By human manipulation of mathematics I refer first to the rewriting of mathematical expressions, like moving a term to the other side of an equation, or factoring out a common factor. However, the term is more general, and also refers to operations done to other visually represented structures, such as graphs. By "augmenting" it, I mean making these manipulations easier to do, as well as more intuitive, using software, so that one can reason better with the underlying mathematics.

The research requires a deep understanding of the mathematics involved, the ways it can be represented, manipulated and used, as well as the relevant software engineering needed to implement this. It will require research into mathematical practice, user interface design, and the logic and structures that are the foundation of mathematics and its representation. The main original contribution to knowledge would be a comprehensive review of existing and new ways of representing mathematics (mathematical structures) with interactive user interfaces, and an evaluation of how effective these are for mathematical practice, collaboration and learning. I will be building a piece of software that implements the best ideas, and will take feedback from users in the University and elsewhere to improve it.

I am currently working to do this for basic algebra, calculus, and vector calculus (in a piece of software I call AugMath [11]). However, I will later move to more complicated structures and methods: algebraic structures, like groups or lattices; categories. Interactive representations of these are useful to be able to reason with them (see for example [1], a diagrammatic proof assistant). The interactive manipulation of algebra in particular has already been studied by one author who calls it "Dynamic Algebra" [9], and who also worked on its implementation in a piece of software called Epsilonwriter [6]. He is now working on an implementation on the web, and he told me by email that he is willing to collaborate with me on the project.

Other examples of software that implement this kind of direct manipulation of algebra are LiveMath [4], Graphing Calculator [5], and more recently, Maple, which incorporated a feature called Clickable Math [7]. Direct manipulation was considered one major area that needed improvement in CAS software by the authors of the proceedings of a 1998 conference on Computer-Human Interaction in Symbolic Computation [12]. However implementations available today still underutilize the power of modern user interfaces (for animation, for example), and are rather limited in scope, with most software not covering anything beyond basic algebra and calculus. This is in contrast to the progress that has been seen in interactive geometry software, with GeoGebra [2] being the most used one, and Cinderella [3] being another powerful one.

Another piece of the puzzle to make Computer Algebra Systems as interactive, intuitive and fun to use as some of the geometry software available, is the animation of the mathematical expressions. Work on this was done by computer graphics pioneer Jim Blinn. In an article, he describes a set of principles for animating algebraic equations, which he calls "algebraic ballet" [10]. These are just natural ways of animating manipulations such as dividing by a factor, or collecting terms. He used these to animate equations in the popular series of documentaries "The Mechanical Universe", and later in "Project Mathematics!". Both of these have a record of being incredibly effective for learning complicated concepts in physics and mathematics (and I can confirm this myself, having enjoyed them when learning freshman physics concepts).

The animation of equations is made even more powerful by coupling it with other forms of visualization like graphs, diagrams or simple simulations of physical systems, which really improve understanding of the underlying mathematical derivation. The power of this is seen in the documentaries mentioned above, and also recently in a series of videos made by Grant Sanderson [8], which explain mathematical concepts in this way. I think it is only natural to combine the work that has been done in animating mathematics with the work on making it interactive (by direct manipulation, for example).

The uses of such a piece of software would be many. Probably the first uses will be in authoring interactive books or animated videos such as Sanderson's. However, the combination of interactivity with animated graphics will effectively make the mathematics software a computer game, falling into the category of serious games; think FoldIt for mathematics. This aspect may have the most important consequences: allowing for a much larger adoption of mathematical software, as well as an increased productivity relative to working with existing CAS software.

My background is in physics and mathematics. My interest in this topic has mostly grown from the frustration of doing a lot of maths in the course while increasingly being aware that computers had great untapped potential in helping me do it. Physics is one of the main areas in which mathematics is used rather than researched, and thus it is natural that with some design thinking (the kind that is found often in coders, hackers, makers, etc., but less often in physicists) one would begin thinking about ways of improving how one does math, rather than just what one does with it.

As a result of this growing interest, I began learning web development approximately a year ago, and began implementing some of the ideas I have for interactive mathematics on AugMath [11], which is currently written (mostly) in JavaScript, though I want to move to a nicer language like Coffeescript [13], and probably utilise a suitable web framework, such as react.js [14]. I also plan to use SageMath (an open-source CAS) as a backend for the software, that will extend its capabilities by offering more standard computer algebra functionality, among other things to verify and help users manipulation. A useful feature (included in Maple) that SageMath could allow is to offer hints on where to go next in manipulating an expression. SageMath also has developed a web interface (SageMathCloud, written in Coffeescript), and a way of making basic interactive GUIs (called "interacts"), which will probably combine nicely with my project.

I am interested in pursuing this research and development at the University of Oxford, as it is a powerhouse in all things scientific. I have also spoken to Dmitrii Pasechnik who has expressed interest in supervising this project, and who is a contributor to SageMath. At the moment I am constantly improving my software engineering skills, as well as learning computer science concepts that I can use in my research. Dmitrii has also advised applying for the Google Summer of Code to work on this with SageMath.

Needless to say, other reasons for choosing Oxford are that during my undergraduate years I have grown very fond of it, I like the city, and developed good connections in it.

References

[1] Quantomatic, `http://quantomatic.github.io/`

[2] Geogebra, `https://www.geogebra.org/`

[3] Cinderella, `http://www.cinderella.de/tiki-index.php`

[4] LiveMath, `http://www.livemath.com/`

[5] Graphing Calculator, `https://www.pacifict.com/`

[6] Epsilonwriter, `http://www.epsilonwriter.com/en/`

[7] Maple: Clickable Math, `http://www.maplesoft.com/products/maple/demo/player/ClickableMath.aspx`

[8] Grant Sanderson personal website, `http://www.3blue1brown.com/`

[9] Jean-François Nicaud, Christophe Viudez Implementation of Dynamic Algebra in Epsilonwriter, Aristod, `http://cermat.org/events/MathUI/13/proceedings/DynamicAlgebra-Implementation-MathUI2013.pdf`, 2013.

[10] James Blinn, The Making of The Mechanical Universe, JPL Graphics Laboratory, `http://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/19900013645.pdf`, 1989.

[11] AugMath, `https://github.com/guillefix/augmath`

[12] Dr. N. Kajler (Editor), Computer-Human Interaction in Symbolic Computation, Texts and Monographs in Symbolic Computation, 1998.

[13] Coffescript, `http://coffeescript.org/`

[14] react.js, `https://facebook.github.io/react/`