A TOOLKIT FOR PRESENTING ADVANCED MATHEMATICS IN SERIOUS GAMING

This work was sponsored by the Office of Naval Research under GRANT11899718:
From Stern to STEM: A Pilot Program for the Recruitment and Education of Veterans
PI - Dr. Anthony W. Dean

BY: KATIE SMITH & JOHN SHULL

OLD DOMINION UNIVERSITY

IDEA FUSION

ACCELERATING VETERANS FROM MILITARY TO STEM CAREERS

STERN 2 STEM

OBJECTIVE

One of the challenges that must be overcome during game development for mathematics is symbolic expression manipulation and display. This poster outlines the first of many phases of the development process of a serious game aimed at accelerating veterans from military to STEM careers. The first phase is the development of a toolkit that allows for the manipulation and display of symbolic expressions.

Several open source mathematics libraries written in Python programming language were selected and utilized. We then developed a toolkit in the form of a C# dynamically linked library (DLL) that provides an intuitive and unified interface for commonly used mathematics functionalities needed by serious games, simplifying installation and software development.

GAME DEVELOPMENT ENGINES

As games have increased in popularity, game engines have been produced to facilitate the development of complex games. Two of the most popular game development engines are Unity and Unreal engine. Both game development engines are popular among developers, free to use (with some restrictions), and have extensive asset stores. While Unity was chosen as the target for the mathematics game toolkit under development, the toolkit could easily be adapted to be compatible with Unreal engine, other proprietary software, and more recently Amazon's cloud platform engine free cloud platform engine Lumberyard.

$$t = \frac{d}{v\cos\theta}$$

$$d = \frac{v^2 \sin(2\theta)}{g}$$

SYMBOLIC MATHEMATICS SOFTWARE

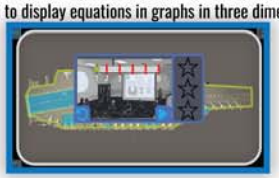
There are a wide variety of both commercial and noncommercial packages for symbolic mathematics. Commercial packages include MATLAB, Wolfram Mathematica, and Maple. Each of these products is highly capable and proficient in symbolic mathematics operations. However, commercial products are not suitable for integration with a game development engine because their source code is proprietary and therefore inaccessible. A review of commercial products was important in order to establish a set of target capabilities for the toolkit to be developed. Once an initial review of commercial products was completed, non-commercial, open source products were considered. The goal was to find a symbolic mathematics library that could perform symbolic operations as well as display equations and graphs. The tools considered were Math.NET Symbolics, Maxima, and SageMath. Math.NET Symbolics is an open source library written in F# and can be used with C# which would make integration with Unity simple. However, Math.NET has very limited capabilities and is not equivalent to the commercial options mentioned previously. Maxima is a computer algebra system written in Lisp. While the capabilities of Maxima are comparable to the commercial options, it was developed as a single application and therefore dismantling the code to use only the relevant pieces would be a time-consuming task. Finally, SageMath is a tool built on open source Python libraries that aims to be a replacement for the commercial tools mentioned earlier. Because it is built from separate libraries, it is easy to use only the necessary libraries to develop a toolkit that can be integrated with a game development engine. SymPy and MatPlotLib are two Python libraries that are used by SageMath. These two libraries allow for symbolic manipulation and equation and graph display, respectively. These two libraries were chosen as the basis for developing our symbolic mathematics toolkit..

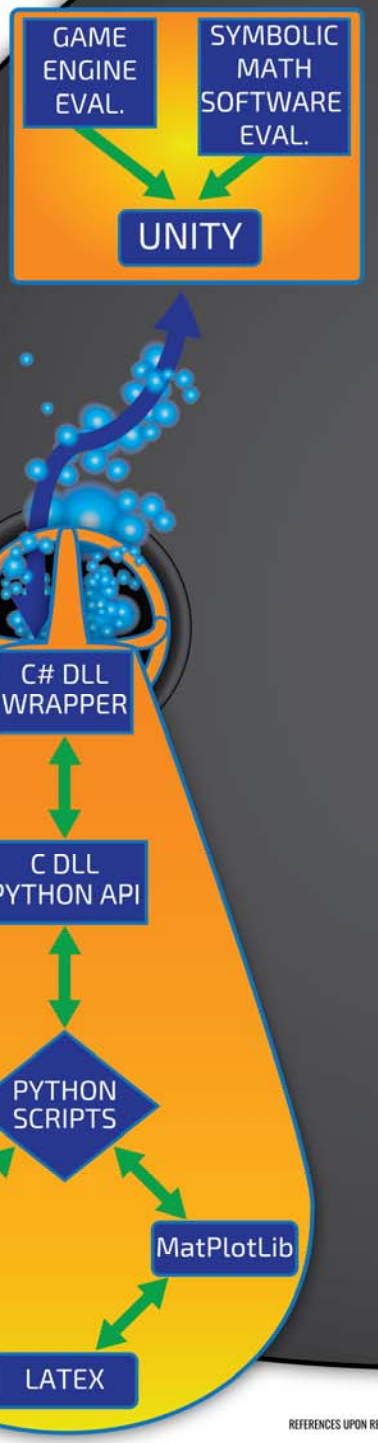
DESIGN & IMPLEMENTATION

For the development of this toolkit, several Python libraries used in SageMath were combined to achieve the desired results. The SymPy library was used to provide symbolic mathematics capabilities. The MatPlotLib library was used to provide the ability to generate displays of equations and graphs. Using the Python C API, Python functions can be called from a C program. A C DLL was written to wrap these functions so they could be called from C#. Then, a C# DLL was written to provide functionalities including equation and graph display for a set of functions.

The C# DLL provides a set of classes that handles formatting the strings that define mathematical expressions so that they are readable when passed to the Python interpreter. Functions from this C# DLL are called directly from Unity scripts. The Python interpreter then stores the equations and graphs as images that can be displayed in Unity as textures. A software architecture diagram with all dependencies is included off to the right. While the toolkit has been developed to be compatible with the Unity game development engine, integration with Unreal engine could easily be achieved by rewriting the C# DLL in C++ which is the scripting language used by Unreal engine. See some of the example images below to see how the toolkit can be implemented to display equations in graphs in three dimensional environments.

$$T = v\frac{dm}{dt}$$





REFERENCES UPON REQU