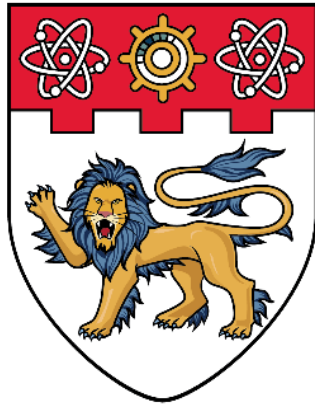


Project No: SCSE18-0105



**NANYANG  
TECHNOLOGICAL  
UNIVERSITY**

---

**SINGAPORE**

## **Displaying curves and surfaces in Unity 3D system**

**Submitted by:**

Khiew Jian Bin

**Supervisor:**

A/P Alexei Sourin

**Examiner:**

A/P Arijit Khan

**A final year project report presented to the Nanyang Technological University  
in partial fulfilment of the requirements of the degree of  
Bachelor of Computer Science  
2018/2019**

## Abstract

*FVRML* is a tool created in efforts to provide a function-based shape modeling and visualization tool. It is currently provided to students in *NTU* as part of a Computer Science coursework project, *Computer Graphics & Visualization*, and other related studies.

*FVRML* runs and relies on *BS Contact*, a software system that uses old methodologies which makes upgrading to work with newer system difficult. *BS Contact* is no longer a suitable candidate to sustain the future needs for *FVRML*, thus its' features were proposed to be ported to a new platform - *Unity3D*.

This study is to investigate the extend of *Unity3D*'s capability as a platform to support *FVRML*'s features, the advantages over *BS Contact* and ultimately to conclude whether *Unity3D* could be used to provide a new coursework software tool to *NTU* students.

This study investigates *Unity3D*'s ability to create curves, surfaces and solids using parametric and implicit functions and to display them in a virtual scene. In addition, providing an intuitive UI system to edit input parameters and a standalone software application for multiple platforms. Participants were then invited to compare *FVRML* + *BS Contact* to the new *Unity3D* software.

## Acknowledgements

I would like to thank the *School of Computer Science and Engineering, Nanyang Technological University* for allowing me to pursue in the studies of computer science and for providing help and resources to guide me on my journey.

I would like to thank my FYP Supervisor, *Associate Professor Alexei Sourin*, for giving me this opportunity and guidance to take up this Final Year Project.

I would like to thank all the *students* who participated in beta testing the new coursework software as well as the students who gave feedback and suggestions to improve the software.

## Table of Contents

Abstract.....	i
Acknowledgements.....	ii
Introduction.....	1
Background.....	1
Problem.....	1
Proposed Solution.....	2
Objective.....	2
Focus.....	3
Related software and relevant research review.....	3
Design Approach.....	4
Proposed algorithm.....	5
Implementation.....	5
User Interface.....	5
UI Header.....	6
UI Scene.....	8
UI Inputs.....	8
Object Settings (Green).....	9
Parameter Settings (Yellow).....	9
Appearance Settings (Cyan).....	9
Execution Code (Light Blue).....	10
Mesh Generation - Runtime Compile.....	13
Mesh Generation – Parametric.....	14
Mesh Generation – Implicit.....	15
Mesh Rendering.....	16
Implementation Issues.....	17
Programable Colors.....	17
Degenerative Normals.....	18
User study and Comparison with FVRML.....	19
Conclusion.....	20
Future Work.....	21
References.....	22
Appendix.....	24

Appendix A – Supported Math Operator and Functions .....	24
Appendix B – Input Format Validation .....	25
Appendix C – Regex.Replace .....	25
Appendix D – User Study Feedback Form .....	27

## **Introduction**

### **Background**

*BS Contact* is a 3D visualization software which visualizes VRML and X3D files. It provides a system that can be modified to fit customer's requirements [1]. It works on Windows and various online web browser such as Internet Explorer (IE) and Firefox.

*BS Contact* had been used and modified to allow running FVRML, a plugin that extends VRML that allows for defining any forms of object's geometry, appearance and transformation using analytical functions in parametric, implicit or explicit representation [2][3] and displaying it in a virtual 3D world.

*BS Contact* has been in continuous development since 1995 and may be considered as a legacy software that underwent multiple system upgrades to support new hardware and software inventions. Similarly, the FVRML plugin for *BS Contact* also requires upgrading to keep it operational along with the latest version of *BS Contact*.

### **Problem**

FVRML was initially designed to work with Microsoft's Internet Explorer (IE) which is now considered obsolete and is no longer being maintained by the developers of Microsoft [4]. Research had been done to investigate the possibility of upgrading FVRML to support newer online browsers such as Firefox and Google Chrome [5]. However, upgrading FVRML requires information about the current SDK implementation of *BS Contact* which is difficult to obtain and understand [5, pp 9-10].

This is due to the complex implementation of BS Contact as a system which is constructed off outdated software design methodologies.

FVRML is an essential education tool provided to students in NTU as part of the Computer Science coursework project, *Computer Graphics & Visualization* as well as other related studies. FVRML currently only works on Windows running IE and Firefox. This is inconvenient and hinders students with Mac or Linux PCs as they will not be able to run FVRML and must find alternatives to complete their coursework.

### **Proposed Solution**

In order to better support the current FVRML functionalities and upgrades to support new technological advancements and platforms, a more flexible and robust system is required that can easily handle legacy upgrades, support internal and external plugins and has a large amount of system documentation available to aid development.

### **Objective**

The aim of this project is to explore a new visualization platform alternative that can perform the same basic functionality as the current software FVRML - function based procedural object generation without having limitations of the old software. The features and the estimated time and difficulty for learning and developing on the new platform will be used to consider the new software as a candidate to replace the old and finally to provide NTU students the new software for their coursework project.

## **Focus**

This project will focus on investigating Unity 3D - a popular game development platform that is well-known for its multi-platform support as well as an extensive library of resources, collection of plugins, tools and large support from its community. Unity 3D also has multiple partnership with leading technology companies such as Microsoft, Google and Oculus which BS Contact lacks.

We will investigate the various implementations on how to display curves, surfaces and solids using parametric and implicit formulas and present an intuitive user interface in Unity 3D. We will then investigate how to develop a standalone application for end-users capable of running on Windows, MacOS, iOS and Android devices. We will also investigate any additional capabilities for Unity to support future 3D graphic visualization research.

If the project is successful, Unity 3D be chosen as a candidate to replace BS Contact for future development of FVRML.

## **Related software and relevant research review**

In the 2002 Conference on CyberWorlds, Lai Feng Min, Alexei Sourin and K. Levinski published paper “Function-based 3D web visualization” which introduced and explained function-based shape modeling using function-defined geometric nodes for VRML called *FShape* which uses the Marching Cubes algorithm by William E. Lorensen and Harvey E. Cline [6]. This was further extended upon and later known and presented as FVRML in the followed up 2005 Conference on 3D Web Technology published paper “Function-based Representation of Complex Geometry and



Appearance” by Qi Liu and Alexei Sourin as well as the 2005 Conference on Computer Graphics and Interactive Techniques in Australasia and Southeast Asia published paper “Function-defined shape metamorphoses in VRML” by Qi Liu and Alexei Sourin which explains more into the implementation of FVRML visualization pipeline.

It is to be noted that FVRML is a plugin for BS Contact while Unity 3D is a game engine which works mainly on scripts to create standalone applications. Programming mythologies of plugins differs from scripting, but the theory and algorithms used in FVRML can be applied to work in Unity 3D with the added benefit of scripting allowing greater control over implementations.

## **Design Approach**

The first task is to investigate the capabilities of Unity 3D to be able to generate curves, surfaces and solids procedurally using scripts.

The second task is to ensure that Unity3D can process analytic function defined parametrically and implicitly to generate polygon meshes within an adequate short time. The design will follow closely to the design of FVRML.

The third task is to explore how Unity 3D handles the rendering of polygon meshes and investigates the possibility of allowing users to modify the appearance of the procedural generated meshes. At minimum, the users should be to change the color and transparency of the displaying mesh.

The final task is to give the users a simple intuitive User Interface to input their analytic functions and other parameters to generate curves, surfaces and solids and display it within the application.

After the tasks is completed, demo software applications will be built to run on Windows, MacOS, IOS and Android platform.

### **Proposed algorithm**

Procedural generation of mesh using analytical functions will be done using similar methods used in FVRML.

Parametric functions will be done using the algorithm which is most commonly used to procedurally generate a 3D cube [7].

Implicit functions will be done using the Marching Cubes algorithm by Paul Bourke [8], a slightly improved version than the traditional marching cube algorithm.

### **Implementation**

#### **User Interface**

Users Interface consist of 3 parts: Header, Scene and Inputs (Fig. 1).

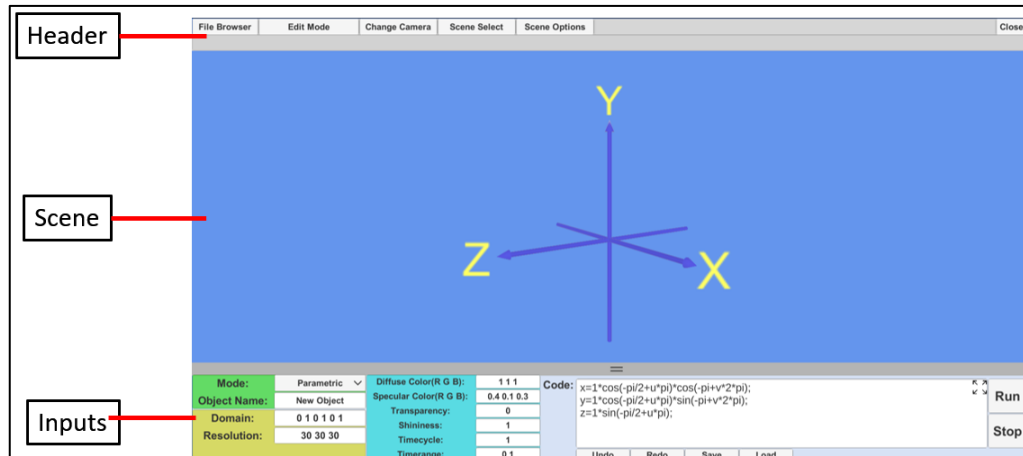


Figure 1- User Interface Overview

## UI Header

The UI Header contains selection of system and scene options provided as buttons that displays additional contextual drop-down options when clicked. The Header consist of File Browser, Edit Mode, Change Camera, Scene Select, Scene Options and Close button.

The File Browser button displays a file explorer menu which displays the current directories and files of a selected Project Path (Fig. 2). The Project Path is saved and remembered for each session and can be changed using the familiar OpenFileDialog window native to the platform. This is applicable to only Windows and MacOS. On Android, the path is automatically chosen as the allocated data storage space for the app by the android device.

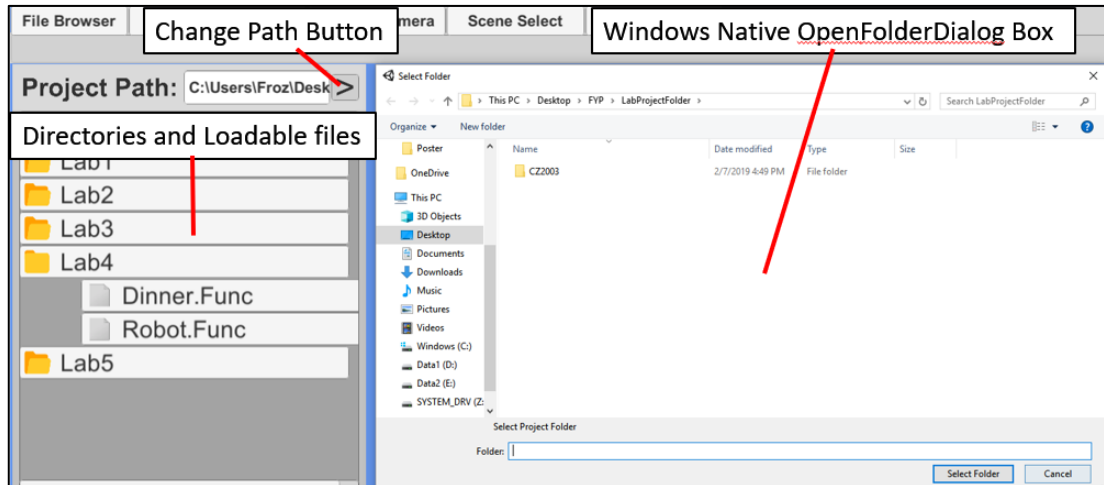


Figure 2 - File Browser

Edit Mode, Change Camera and Scene Select buttons are currently placeholders for future work to be discussed in the later sections.

The Scene Options button allows the user to toggle Wireframe Viewing Mode on the generated meshes and Show Coordinate Axis within the scene (Fig. 3).

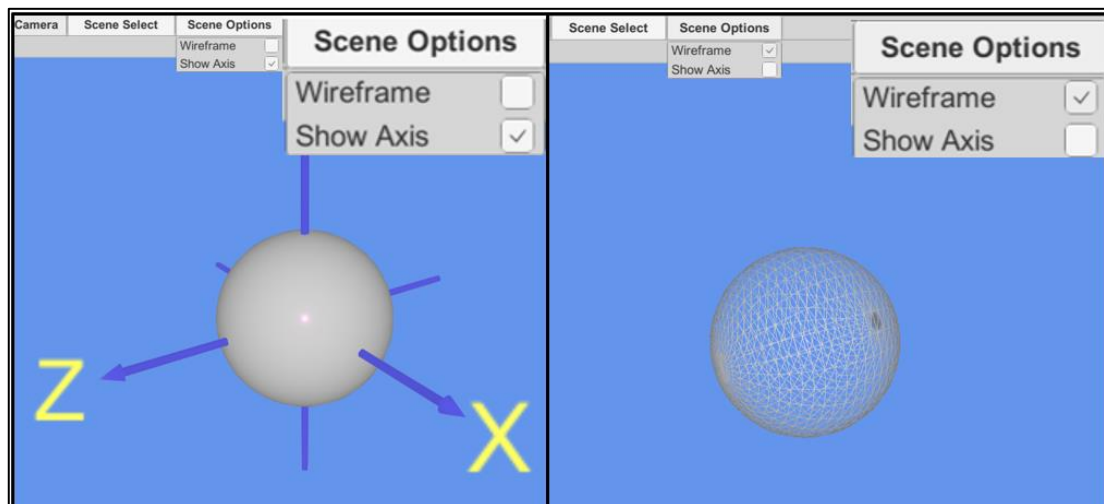


Figure 3 - Left: Enabled Show Axis, Right: Enabled Wireframe

The Close button allows the user to close the application.

## UI Scene

The UI scene displays the virtual scene through Unity's Camera system. The camera is setup to display a blue background, the coordinate axis and any procedural generated meshes. Selection of different camera interaction modes can be done using the Change Camera Button on the UI Header. Currently, only the Examine mode is available. The Examine mode allows users to rotate the camera and zoom in or out in relative to the generated object. The camera also moves and zooms automatically to re-adjusted to view the entirety of newly generated object.

## UI Inputs

The UI Inputs is a toggled panel which the user can choose to hide or show. The UI Inputs contains all the input settings used in procedural generating the mesh to be display on the UI Scene. It is separated into 4 colored sections: Object Settings (Green) Parameter Settings (Yellow), Appearance Settings (Cyan) and Execution Code (Light Blue) (Fig. 4). The UI Inputs are color coded to make it for user to contextualize and to group the UI elements according to their function.



Object Settings		Apperance Settings		Execution Code			
Mode:	Parametric	Diffuse Color(R G B):	1 1 1	Code: $x=1*\cos(-\pi/2+u*\pi)*\cos(-\pi+v*2*\pi);$ $y=1*\cos(-\pi/2+u*\pi)*\sin(-\pi+v*2*\pi);$ $z=1*\sin(-\pi/2+u*\pi);$	 		
Object Name:	New Object	Specular Color(R G B):	0.4 0.1 0.3				
Domain:	0 1 0 1 0 1	Transparency:	0				
Resolution:	30 30 30	Shininess:	1				
		Timecycle:	1				
Parameter Settings		Timerange:	0 1	Undo	Redo	Save	Load

Figure 4 - UI Inputs

### Object Settings (Green)

Users are given a selection of switching between two input modes, Parametric and Implicit to choose which formula type the user wants to use. Switching to either modes will make the Parameter Settings UI display only the relevant inputs to define the user's parametric or implicit formula (Fig. 5).

<b>Mode:</b>	Parametric ▾	<b>Mode:</b>	Implicit ▾
<b>Object Name:</b>	New Object	<b>Object Name:</b>	New Object
<b>Domain:</b>	0 1 0 1 0 1	<b>BBSize:</b>	10 10 10
<b>Resolution:</b>	30 30 30	<b>BBPosition:</b>	0 0 0
		<b>BBResolution:</b>	50 50 50

Figure 5 - Left Parametric Mode, Right Implicit Mode

Object Name can be changed to identify the current displayed mesh and used for detecting corresponding file names for saving and loading.

### Parameter Settings (Yellow)

Users are able to type the parameter domain and resolution of the parametric formulas or type the bounding box center, bounding box size and resolution of the implicit formulas depending on the input modes selected in Object Settings.

### Appearance Settings (Cyan)

Users are able to type and modify the meshes appearance by defining the Diffuse Color, Specular Color, Transparency and Shininess. Users are also able to modify the time cycle interval and time range to affect the morphing of time-dependent meshes.

## Execution Code (Light Blue)

### Code Textbox

Users are given a large textbox to type in their analytical formula following a programming style syntax, ending each statement with a semi colon. The textbox can expand by clicking the enlarged button to display a larger scrollable textbox for users to view and input longer formulas as shown in Fig 6.

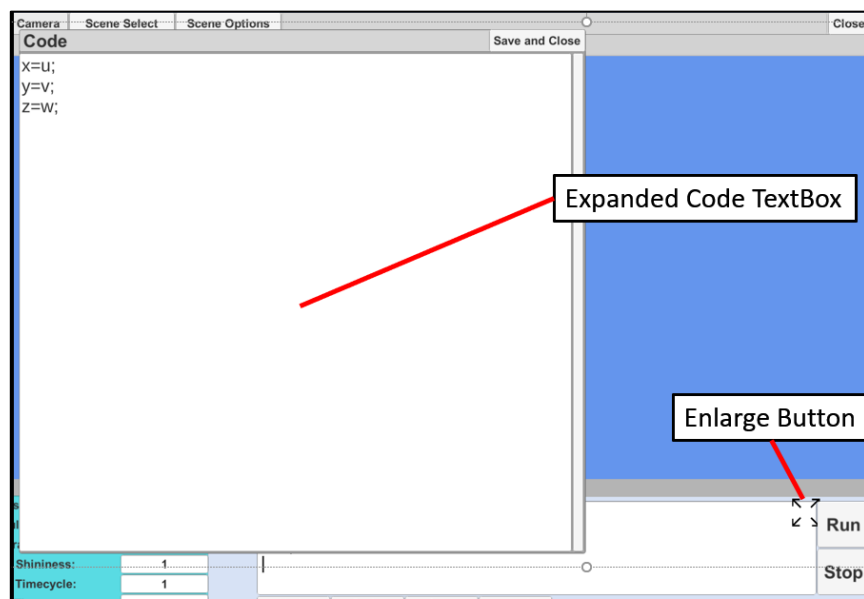


Figure 6 - Expanded Code Textbox

Users are able to type in a parametric formula to generate meshes by defining the cartesian coordinates  $x$ ,  $y$ ,  $z$  using parametric coordinates  $u$ ,  $v$ ,  $w$  in any order. For defining curves, one parametric coordinate ( $u$ ) is used. For defining surfaces, two parametric coordinates ( $u$ ,  $v$ ) are to be used. For defining solids, three parametric coordinates ( $u$ ,  $v$ ,  $w$ ) are to be used. Users are able to define the domain range of  $u$ ,  $v$ ,  $w$  and the sampling resolution within Parameter Settings (Yellow).

Users are also able to enter an implicit formula to generate meshes by defining the bounding box center, bounding box size and the sampling resolution within Parameter Settings (Yellow). The implicit formula is restricted to the form  $f(u,v,w,t) \geq 0$  of more than or equal to zero.

Users can also include the time variable “t” which increments with Unity’s time system in the parametric or implicit formula to create time-dependent meshes that will change its shape with time.

Common mathematical constants and functions (Appendix A) are also supported to be used with the formulas and these can be extended to include more.

### **Undo & Redo**

Undo and Redo buttons are included to give users the ability to undo any typing mistakes. Alternatively, the standard keys press for Undo and Redo on Windows (Ctrl+Z, Ctrl+Y) and MacOS (Cmd+Z, Cmd+Shift+Z) also performs the same function.

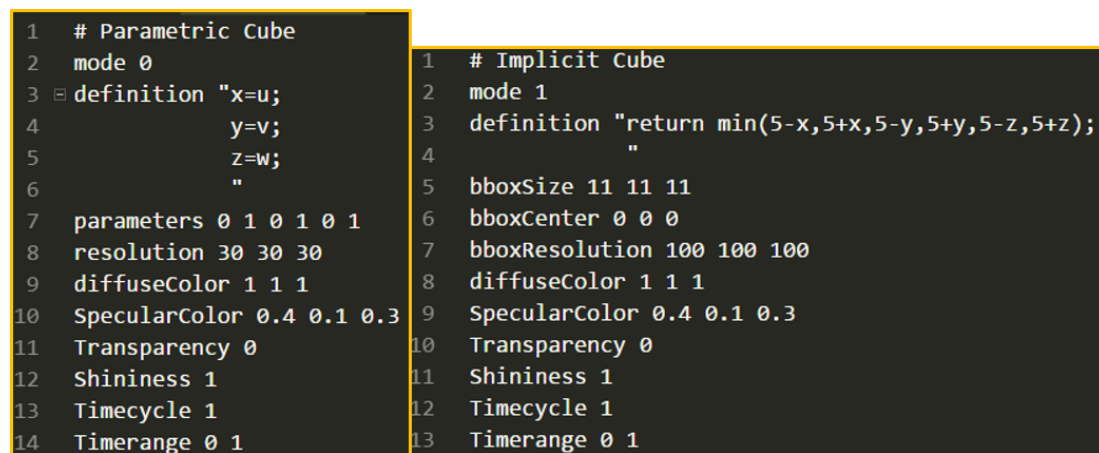
### **Save & Load**

Save and Load buttons are used to save all the UI Input setting into a file, modify the file using any text editor, and load the modified back into the program to generate and display their updated mesh.



The contents of the saved file (Fig. 7). are copied directly from the UI Input without any input validation. This is to allow users to also save their incomplete work without having first be runnable.

The contents of the saved file are being read and loaded line by line and skips commented line denoted by the '#' symbol at the start of the line. This allows users to rearrange the order of lines and add comments on the file.



```
1 # Parametric Cube
2 mode 0
3 definition "x=u;
4           y=v;
5           z=w;
6           "
7 parameters 0 1 0 1 0 1
8 resolution 30 30 30
9 diffuseColor 1 1 1
10 SpecularColor 0.4 0.1 0.3
11 Transparency 0
12 Shininess 1
13 Timecycle 1
14 Timerange 0 1

1 # Implicit Cube
2 mode 1
3 definition "return min(5-x,5+x,5-y,5+y,5-z,5+z);
4           "
5 bboxSize 11 11 11
6 bboxCenter 0 0 0
7 bboxResolution 100 100 100
8 diffuseColor 1 1 1
9 SpecularColor 0.4 0.1 0.3
10 Transparency 0
11 Shininess 1
12 Timecycle 1
13 Timerange 0 1
```

Figure 7 - Saved File Contents

Clicking Save or Load will create the familiar OpenFileDialog Box native to the running platform on Windows or MacOS for users to save or load the file. On Android, the buttons saves and loads file depending on the matching Object Name specified in Object Settings.

## Run & Stop

After finishing typing their formulas, the user can click the Run button to start mesh generation. The program uses a strict format (Appendix B) imposed on all UI Input

settings which will be validated and will display an error message (Fig 8) if there are any problems with the format, otherwise the program will compile the input formula, generate the mesh and display it on the UI Scene.

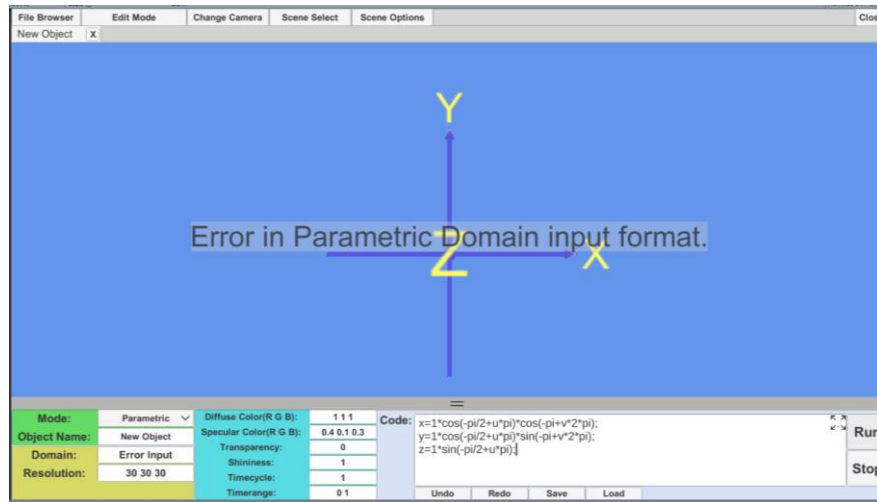


Figure 8 - Parametric Domain Input Error

The Stop button is available to allow users to stop rendering the mesh when the application load is too heavy.

## Mesh Generation - Runtime Compile

Mesh generation starts by obtaining the validated input formula entered by the user. It will remove any existing mesh and proceed to check the code and keep track for the usage of parameter “u”, “v”, “w” and “t” to aid in faster rendering. For Implicit mesh generation, only “t” is tracked.

The code is then processed by using .Net Regex.Replace to find for the usage of mathematical constants and functions and replacing them to their C# code equivalent (Appendix A). Additional use of Regex.Replace is used to process the code further to

allow the use of custom defined variables and the caret sign '^' for exponent. After processing, the code along with the additional parameter settings is compiled into a C# script which executes to generate the mesh.

For code and explanation of Regex.Replace usage, see Appendix C.

An alternate method was investigated to use a string-to-math parser that based on compiler design and lexical analysis, this method however had resulted in slower mesh generation compared to runtime compile.

### Mesh Generation – Parametric

The same algorithm used in generating a 3D cube is used to create curves (Fig. 9) surfaces (Fig. 10) and solids (Fig. 11) meshes defined using parametric function with slight modifications to include definition for parameter Domain and Resolution.

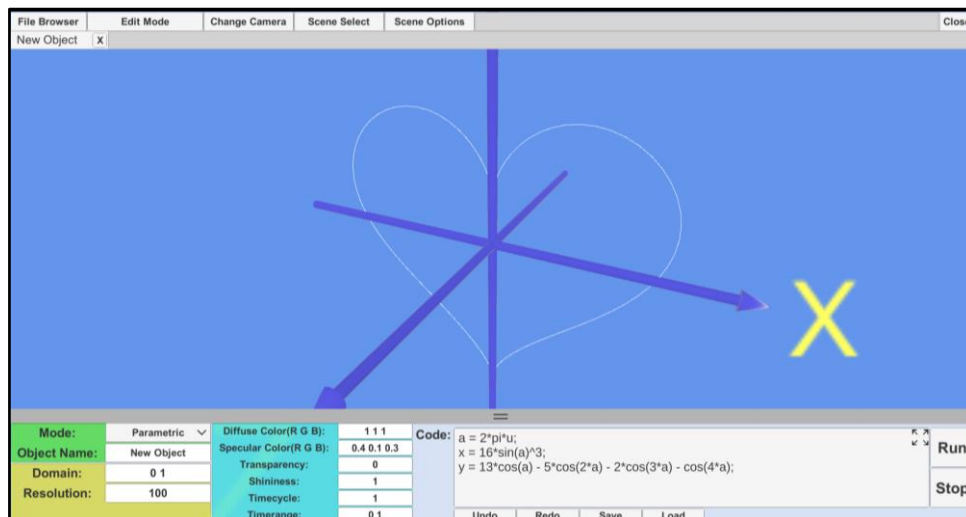


Figure 9 - Heart Curve in Unity

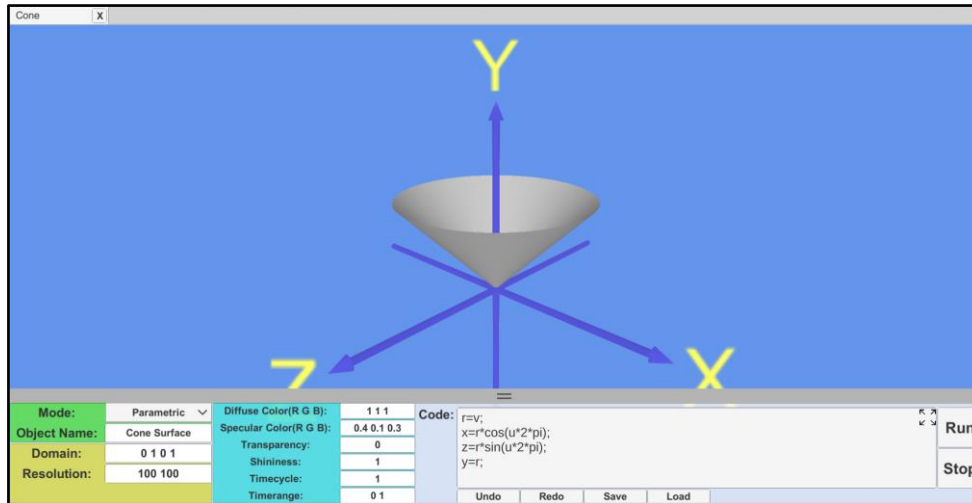


Figure 10 - Cone Surface in Unity

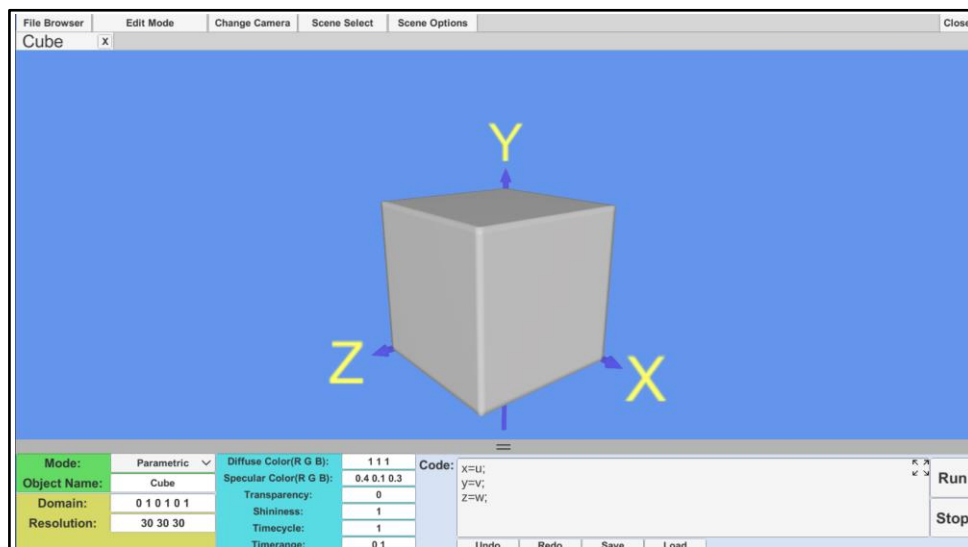


Figure 11 – Cube Solid (Parametric) in Unity

## Mesh Generation – Implicit

Marching Cubes algorithm by Paul Bourke is used to implement mesh generation (Fig. 12) defined by implicit functions. Minor modifications were made include definition for Marching Cube Bounding Size, Marching Cube Bounding Position, and Marching Cube Bounding Resolution.



Figure 12 - Implicit Mesh Generated Robot in Unity

## Mesh Rendering

Rendering is done using Unity's render pipeline that works on Programmable Shaders. The appearance of meshes is determined by the Shader code. Unity provides a Standard Material that uses its built-in Physically-Based Rendering (PBR) shader that supports specular highlights and reflections. This shader and properties definition are different from the Phong shader implementation used in FVRML. However, an educated guess can be made to pair the old appearance definition to the new unity material properties. Unity's Standard Material is then used to render the generated meshes. The inputs from appearance settings are taken and update the new material properties. This will give a better appearance while using the old appearance definitions.

Table 1 – Appearance Settings to Material Properties pairing

Appearance Settings	Updated Standard Material Properties
Diffuse Color: R, G, B	Albedo Color: R, G, B
Specular Color: R, G, B	(Hidden in Shader) Specular Color: R, G, B
Transparency	Albedo Color: A
Shininess	Smoothness

To better display parametric curve meshes, an additional Standard Material Properties, emission color, is used and set to brighten the appearance of curves.

### Implementation Issues

The current algorithm that used to generate parametric meshes, only works with certain order when defining the parametric formula. The Cartesian coordinates must match with the Parametric coordinates in the following order:  $x=u$ ,  $y=v$ ,  $z=w$ . If this order is changed, the triangle faces of generated mesh will reverse giving an inside-out appearance as seen in Fig. 13.

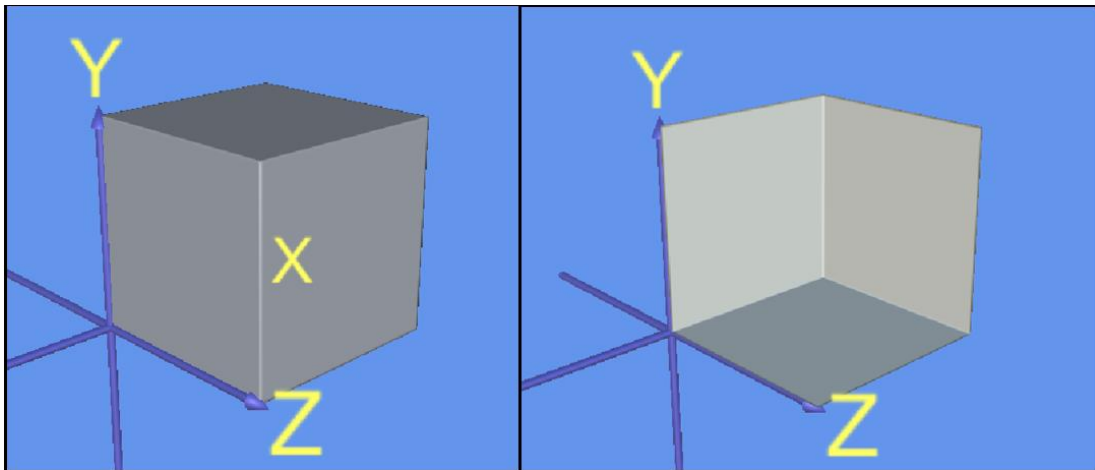


Figure 13 - Left: Normal, Right: Inside-Out

This issue is fixed by modifying the Shader to render both front and back faces

### Programable Colors

FVRML is able to define the meshes color appearance using the varying input parameter  $u,v,w$ . Unity 3D is able to achieve similar results (Fig. 14) by runtime compiling a new shader based of the appearance settings for each new mesh generated, the new compiled shader is then applied onto the mesh's material and the mesh will be colored based of the varying input parameters entered.

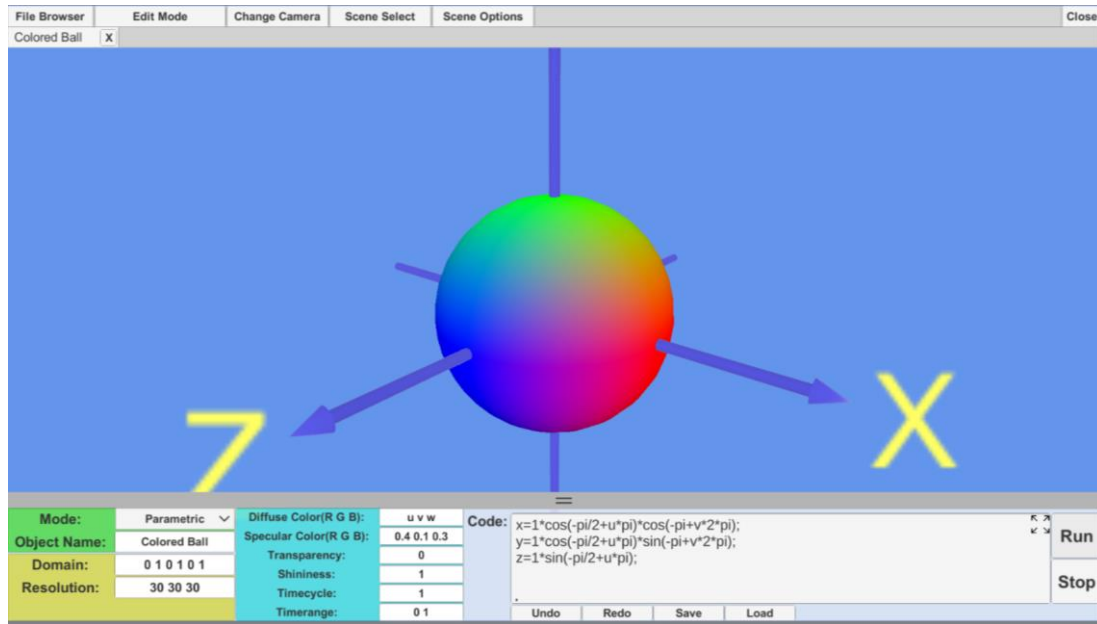


Figure 14 - Programable Colored Ball: Diffuse Color:  $u \ v \ w$

However, this only works within the Unity Editor and is unavailable to the end-user. Unity uses its internal program - UnityShaderCompiler.exe to automatically recompile any new changes to shader files during editing. We were unable to find ways to access this compiler directly by script and to package it to made available to the end-user. It is to be noted that Unity 3D previously allowed users to compile shaders during runtime but had since been removed due to improvements and complexity made in the newer rendering pipeline. Unity might make runtime compile shaders available in the future.

### Degenerative Normals

Certain generated objects have unexpected darken color appearance on the edge of the shape. These are degenerative normals (Fig. 15) that is formed when vertices have zeroed normal which propagates and affects other vertices' normals calculation, this problem also exists in FVRML. This is caused by a combination of the current technique used in generating meshes and the algorithm used in calculating meshes. A

short fix is to change the parametric formulas to prevent vertices having the same points and zero area faces.

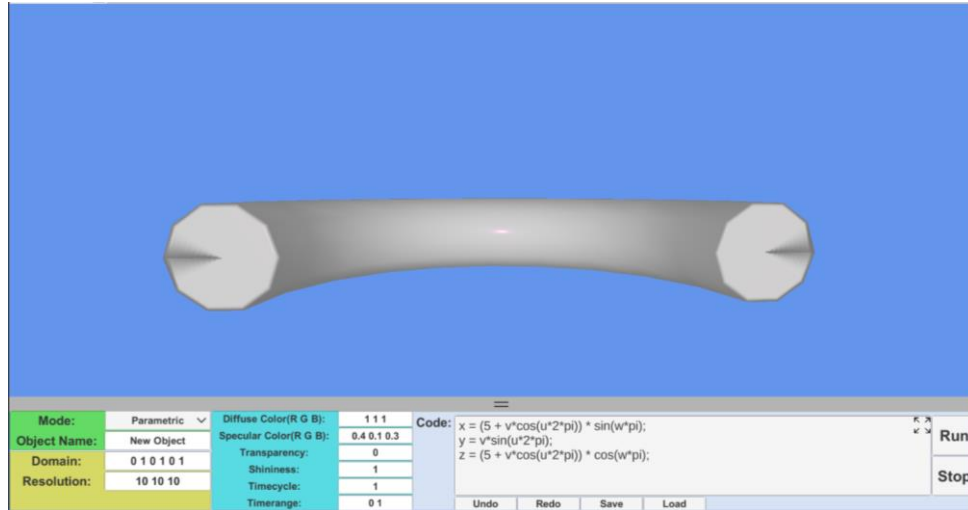


Figure 15 - 3D Half Torus with degenerative normal

## User study and Comparison with FVRML

A closed user study was conducted from 11 February to 17 March in which participants are provided with the demo standalone applications for Windows and MacOS to explore. Participants are then requested to fill out simple feedback form (Appendix E) to indicate whether the demo app is more convenient to use than FVRML and reasons why.

A total of 58 participants had been requested to try the new software, 90% of participants feedback indicated that they agree the new software is more convenient in addition to giving a lot of suggestions for improvement mainly on the UI. Most participants stated that Unity 3D provides a friendlier approach to editing and displaying 3D objects because it allows a full view of the object's shape and



appearance definition and formula while also given the flexibility to hide or show the UI elements which increase the user experience.

Table 2 – List of suggested features from participates to be considered

- |  |
|--|
| <ul style="list-style-type: none"><li>• Ability to change the coordinate axis model color</li><li>• Adjustable slider for color, transparency, shininess and others that automatically updates appearance.</li><li>• Customizable background color</li><li>• Separate All Multi-Input fields such as Domain and Resolution to individual fields for individual parameter.</li><li>• Error message for code compilation errors.</li></ul> |
|--|

User study have shown that despite being an early developed software, participates had expressed willingness to see improvement being done on the new software.

## Conclusion

We are able to display curves, surfaces and solids using analytical functions in parametric and implicit form using Unity 3D. Successfully explored and provide a flexibly UI system for users to define the functions and modify the object's shapes and appearance as well as the ability to save this information to a file and pass on to another user to load and view on their Windows, MacOS, Android or IOS device.

Plenty of resources was available on Unity 3D development as most of the study investigation work done on the User Interface. This suggest that Unity 3D is a good candidate to replace FVRML and should be investigate further.

## Future Work

In addition, preliminary test in Unity had shown potential future work on providing improvements interactivity to the software.

- Mesh generation of objects can be done on the GPU using compute Shader to increase performance speed.
- With Unity's render pipeline, the object appearance can be fully customized by user and loaded from various online shader libraries. Various styles of mesh rendering can be achieved, examples seen in Fig 16.

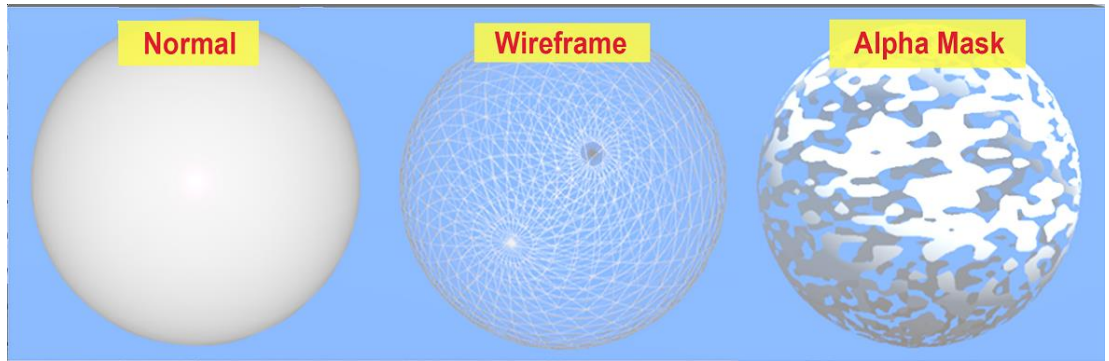


Figure 16 – Normal, Wireframe, Alpha Mask Rendering modes

- As a game engine, physics can apply onto the generated objects to have it affected by simulated gravity and wind. The interaction between these objects can be observed.

## References

- [1] Bitmanagement. (2012). BS Contact is a real-time and online capable viewer offering full interactivity. [Online] Available: <http://www.bitmanagement.com/products/interactive-3d-clients/bs-contact> [Accessed 26 Sep. 2018].
- [2] Q. Liu and A. Sourin, "Function-based representation of complex geometry and appearance," in 10<sup>th</sup> Int. Conf. 3D Web Technology, Web 3D 2005, Network, 2005, pp. 123–134.
- [3] Q. Liu and A. Sourin, "Function-defined shape metamorphoses in VRML," in *3rd Int. Conf. Computer Graphics and Interactive Techniques in Australasia and Southeast Asia 2005*, Dunedin, New Zealand, 2005, pp. 339-346.
- [4] Bolton, D. (2016). Microsoft is finally about to end support for internet explorer 8, 9 and 10. [Online] Available: <https://www.independent.co.uk/life-style/gadgets-and-tech/news/microsoft-ends-support-internet-explorer-version-8-9-10-a6800881.html> [Accessed 26 Sep. 2018].
- [5] Z. Liu, "Visual immersive geometry," Nanyang Technological University, DRNTU, Final Year Project (FYP) ntu.10356.70357, Apr. 2017.
- [6] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3D surface construction algorithm," presented at the Proceedings of the 14th annual conference on Computer graphics and interactive techniques, 1987.
- [7] Catlike Coding. (2019) Rounded Cube Building in 3D [Online] Available: <https://catlikecoding.com/unity/tutorials/rounded-cube/>

- [8] Paul Bourke (1994) Polygonising a scalar field [Online] Available:  
<http://paulbourke.net/geometry/polygonise/>

## Appendix

### Appendix A – Supported Math Operator and Functions

User Input	C# Code Equivalent
$\sin(x)$	Math.Sin(x)
$\cos(x)$	Math.Cos(x)
$\tan(x)$	Math.Tan(x)
$\arcsin(x)$	Math.Asin(x)
$\arccos(x)$	Math.Acos(x)
$\arctan(x)$	Math.ATan(x)
$\cosh(x)$	Math.Cosh(x)
$\sinh(x)$	Math.Sinh(x)
$\tanh(x)$	Math.Tanh(x)
$\operatorname{atan2}(x)$	Math.Atan2(x)
$\operatorname{abs}(x)$	Math.Abs(x)
$\sqrt{x}$	Math.Sqrt(x)
$\operatorname{pow}(x, y)$	Math.Pow(x,y)
$\operatorname{floor}(x)$	Math.Floor(x)
$\operatorname{ceil}(x)$	Math.Ceil(x)
$\log(x, y)$	Math.Log(x, y)
$\exp(x, y)$	Math.Exp(x, y)
$\min(a, b, +\text{params})$	Maths.Min(a, b, +params)
$\max(a, b, +\text{params})$	Maths.Min(a, b, +params)
$\pi$	Math.PI

## Appendix B – Input Format Validation

Inputs must follow the following format to be consider valid input

Valid Inputs
Parameter Domain: “# #” or “# # # #” or “# # # # # #” and # = real number
Parameter Resolution: “#” or “# #” or “# # #” and # = Integer and # != 0
BBSIZE: “#” or “# # #” and # = real number
BBPosition: “# # #” and # = real number
BBResolution: “#” or “# # #” and # = real number
DiffuseColor: “# # #” and # = real number, value auto clamps to 0-1
SpecularColor: “# # #” and # = real number, value auto clamps to 0-1
Transparency: “#” and # = real number, value auto clamps to 0-1
Shininess: “#” and # = real number, value auto clamps to 0-1
TimeCycle: “#” and # = real number
Timerange: “# #” and # = real number

## Appendix C – Regex.Replace

Special Replace to include user created variables

<pre>inputcode = Regex.Replace(inputcode,  "([ ^uvwxyzt0-9 ] [a-zA-Z_][a-zA-Z0-9_]+) \\=(.+)", "double \$1=\$5");</pre>	
<pre>[ ^uvwxyzt0-9 ]</pre>	To qualify as a user variable, if it is a single character, the character must not be one of the reserved key letters: u,v,w,x,y,z,t or a number.
<pre>([a-zA-Z_][a-zA-Z0-9_]+)</pre>	If it is multi-character, it can start with an alphabet or underscore followed by alphanumerics
<pre>\\=(.+)</pre>	Looks for an equal sign to check for variable assignment
<pre>double \$1=\$5</pre>	Adds ‘double’ in front of the user variable to allow for runtime compile

Special Replace to include the caret ‘^’ pow symbol.

Complex Regex is used to allow usage of ‘^’ to work with brackets

E.g.  $(1+1)^{(1+1)} \rightarrow \text{Math.Pow}(1+1,1+1)$

<pre>inputcode = Regex.Replace(inputcode,      //Left expression     "(((0 [1-9][0-9]*) \\.(0 [0-9]*[1-9]))? [a-zA-Z_][a-zA-Z0-9_]*)      ((Maths?\\.[a-zA-Z0-9]*)?     \\((?:[^(] (?&lt;counter&gt;\\( (?&lt;-counter&gt;\\) )))+(?(counter)(?!))\\)))\\s*)"      + "\\^" +      //Right Expression     "([\\s]*((0 [1-9][0-9]*) \\.(0 [0-9]*[1-9]))? [a-zA-Z_][a-zA-Z0-9_]*)      ((Maths?\\.[a-zA-Z0-9]*)?     \\((?:[^(] (?&lt;counter&gt;\\( (?&lt;-counter&gt;\\) )))+(?(counter)(?!))\\)))",      "Math.Pow(\$2,\$9)");</pre>	
“(0 [1-9][0-9]*) \\.(0 [0-9]*[1-9]))? [a-zA-Z_][a-zA-Z0-9_]*)	Looks for a whole or decimal number or a variable.
(Maths?\\.[a-zA-Z0-9]*)?	Looks for math functions usage – Math.sin()
\\((?:[^(] (?<counter>\\( (?<-counter>\\) )))+(?(counter)(?!))\\)))\\s*)	Looks for balanced parenthesis bracket
\\^	Check for caret

## Appendix D – User Study Feedback Form

No.	Question	Answer
1.	I am testing (please select one or both answers by placing "x"):	Windows version [ ] Mac version [ ]
2.	The new tool is more convenient than the current VRML-based software (please select one answer by placing "x"):	Agree [ ] Neutral [ ] Disagree [ ]
3.	I propose the following improvements to the user interface	
4.	I have found the following bugs (please write brief descriptions with reference to the version of the software):	