LABORATORY MANUAL

CZ2003: Computer Graphics and Visualization
SW Lab 1B

Visual Mathematics

SESSION 2016/2017
SEMESTER 2
COMPUTER SCIENCE COURSE

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
NANYANG TECHNOLOGICAL UNIVERSITY

VISUAL MATHEMATICS

1. OBJECTIVE

During five lab sessions, you will learn how to visualize curves, surfaces and solid shapes defined by mathematical formulas. Upon completion of these labs you will know:

- How to define shapes by implicit, explicit and parametric analytic functions, and
- How to make shape morphing from one parametrically defined surface to another.

2. EQUIPMENT

You will work on a personal computer running MS Windows 8 with the following software tools installed:

- *BS Contact VRML/X3D* plug-ins to IE and Firefox[1],
- *FVRML* plug-in[2]
- *VRMLpad (*you can also install a trial version on your home PC from[3])

3. INTRODUCTION

You will use educational visualization software—function-based extension of Virtual Reality Modeling Language (VRML) and Extensible 3D (X3D)[4]. To avoid learning VRML/X3D and concentrate on visualization principles—mathematical definitions, coordinate domains and sampling resolutions—you will use the provided source code templates to only fill them in with mathematical formulae and visualization parameters.

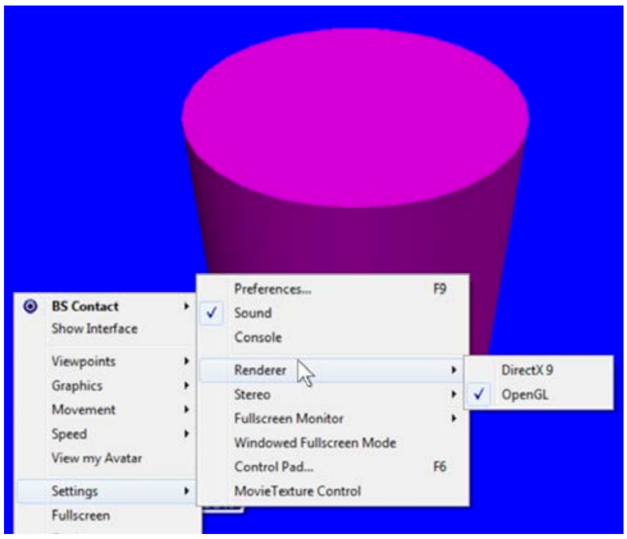
VRML and X3D are ISO standard file formats for describing interactive 3D objects and virtual worlds. They are designed to be used on the Internet, intranets, and local client systems. They are also intended to be universal interchange formats for integrated 3D graphics and multimedia. VRML and X3D are capable of representing static and animated dynamic 3D and multimedia objects with hyperlinks to other media such as text, sounds, movies, and images. VRML and X3D are following *declarative programming style*, i.e. it tells the computer *what to do*. It differs from the *imperative programming style*, like in C and in a popular graphics library OpenGL, which tell the computer *how to do* things.

Both VRML and X3D follow the same programming principles and VRML is one of the formats of X3D. Therefore, let's consider an example of how to create a simple VRML scene with one object: a shiny purple cylinder with a radius of 3, and a height of 6 on a blue background. The VRML code defining it is listed in Figure 1a and it can be also downloaded from the course-site. The code contains *File Header*, *Transform Node*, *Shape Node*, *Appearance Node*, *Geometry Node, Light source and Background Nodes*. Every VRML file starts with the header *#VRML V2.0 utf8*. The *utf8* specification refers to an ISO standard for text strings known as UTF-8 encoding. Geometry of the shape is defined in the *geometry* node. The *Transform* node is a grouping node that defines *Translation*, *Rotation* and *Scaling* transformations. In the example, it is the rotation about axis *X* by 0.7 radians. The *Shape* node is the basic container node for a geometry object. The *Appearance* node defines color, the smoothness and the shininess of the surface, etc. This is done with the *Material* and texturing nodes. Colors are defined as red, green and blue components in the range [0,1]. In the example, purple and shiny material is defined. Finally, a light source and a background color are defined.

The VRML code can be created and edited with any text editor, but the provided in the lab license of VRMLpad is more convenient. The VRML file has to be saved with extension *.wrl*. To visualize it, you need to use Internet Explorer or Mozilla Firefox internet browsers with *BS Contact* plugin installed. Once you click at the *.wrl* file, the VRML shape defined in Figure 1a will be displayed as it is shown in Figure 1b.

```
#VRML V2.0 utf8

# Transform node may define 3 affine transformations: scaling, rotation and translation.
# They will be applied in exactly this order. Transform nodes can be nested.
Transform {
  rotation  1 0 0 0.7
  translation 0 0 0
  scale 1 1 1
  children[

# Shape node includes geometry and appearance definitions (nodes).
        Shape {
# Purple color with 50% of shininess is defined
              appearance Appearance {
              material Material {
                    diffuseColor  0.5 0 0.5
                    shininess     0.5 } }
# Geometry is defined as a cylinder with radius 3 and height 6.
# The centre of the cylinder is at the origin.
# Top, bottom and side polygons can be removed if FALSE is defined.
              geometry Cylinder  {
                    radius 3         height 6
                    side TRUE  top TRUE  bottom TRUE
              }
                    }
        ]
  }
# Point light source with white color is defined at point with coordinates 250 400 150
# It can be visible within 1500 m distance.
PointLight {
  on    TRUE
  ambientIntensity  1
  color 1 1 1
    location 250 400 150
  radius 1500    }
# Blue background color is set for the scene.
Background { skyColor 0 0 1 }
```

(a)



(b)

Figure 1. A VRML code defining a cylinder and VRML scene visualization with BS Contact VRML/X3D viewer. The code is available in file cylinder.wrl

VRML and it current successor Extensible 3D (X3D) are very versatile programming tools but they only visualize shapes defined by polygons (triangles). Definitions of the polygons' vertices by their coordinates is a tedious task. Usually, third party software tools are used to generate these coordinates. Another approach to visualization is to define geometric shapes by mathematical formulas which allow the visualization software for computing coordinates of all the points belonging to the shapes. To be able to visualize geometric shapes defined by mathematical formulas, you will be using ***FVRML plug-in*** which is a function-based extension of VRML. FVRML allows for including in VRML practically any type of object's geometry, sophisticated graphics appearance and transformation by defining them analytically straight in the VRML code. Lecture Module 3 introduces how to use mathematics for defining geometry. Here is a brief summary.

For defining geometry, appearance and their transformations, three types of functions can be used concurrently. They are implicit, explicit and parametric functions.

Implicit functions are the functions defined as $f(x,y,z,t)=0$, where x, y, z are Cartesian coordinates and *t* is the time. You will only use them for defining surfaces since VRML is a 3D visualization system. The implicit functions equal to zero for the points located on the surface. Hence, a sphere can be defined by equation: $R^2 - x^2 - y^2 - z^2 = 0$.

Explicit functions define one value as a function of the others. Thus functions $g=f(x,y,z,t)\geq 0$, known in computer graphics as FReps, can be used for defining solid objects. In this case, the function equals to zero for the points located on the surface of the object, positive values indicate points inside the solid object, and negative values are for the points which are outside the object. To illustrate it, let us consider an example of function $g=\sqrt{x^2+y^2+z^2}$ which defines a distance from the origin to any point with Cartesian coordinates (x, y, z). If we use function $g = R - \sqrt{x^2+y^2+z^2} \geq 0$, it will define a solid origin-centered sphere with radius R. The equation of a solid sphere could be also written as $g = R^2 - x^2 - y^2 - z^2 \geq 0$. Addition of time *t* to the parameters of the function will allow us to make variable time-dependent shapes. For example, a sphere bouncing up and down by height a during time $t=[0,\ 1]$ can be defined as $g = R^2 - x^2 - (y - a\sin(t\pi))^2 - z^2 \geq 0.$

Parametric functions are explicit functions of some other coordinates u, v, w (which are called parameters) and time t. They can define Cartesian coordinates x,y,z of curves, surfaces, solid objects, and r,g,b values of the colors as:

$$x=f_1(u|,v|,w|,t);\ \ y=f_2(u|,v|,w|,t);\ \ z=f_3(u|,v|,w|,t)$$
$$r=\varphi_1(u|,v|,w|,t);\ \ g=\varphi_2(u|,v|,w|,t);\ \ b=\varphi_3(u|,v|,w|,t)$$

To define a curve, only one parameter u has to be used, to define a surface–2 parameters u and v are required, for solid objects–all three parameters u, v, w have to be used. When *t* is added, these objects will become time-dependent. For example, the bouncing up and down sphere, whose color dynamically changes from green to red, can be then defined as:

$$x = R\cos v\cos u$$
$$y = R\sin u + a\sin(t\pi)$$
$$z = R\sin v\cos u$$
$$r = \sin(t\pi)$$
$$g = 1 - \sin(t\pi)$$
$$b = 0$$
$$u = [0, 2\pi],\ \ v = [0, \pi],\ \ t = [0, 1].$$

Geometry and color can be defined by implicit, explicit or parametric functions in their own domains and then merged together into one shape. For example, we can define an origin-centred sphere with radius 0.7 by the following implicit function:

$$0.7^2 - x^2 - y^2 - z^2 = 0$$

Then, a parametrically-defined color is applied to it:

$$r=1 \quad g=fabs\,(\sin(u)) \quad b=0$$

The final shape created with these implicit and parametric functions and its FVRML code are shown in Figure 2. The code is also available for downloading from the course-site. The first part of this code, beginning with EXTERPROTO, defines a prototype of the function-based VRML plug-in—declarations of the variables used. Only a little part of this code is shown in the Figure to save space. The EXTERNPROTO code is the same for any FVRML code. Note also that only the left part of the implicit function has to be typed in the code since it is always assumed to be defined as greater or equal than 0.

```
#VRML V2.0 utf8
EXTERNPROTO FGeometry [
exposedField SFString definition
exposedField MFFloat parameters
……………………………………..
[ the rest of the EXTERNPROTO is skipped]

FShape {
  geometry FGeometry {
        definition "0.7^2-x^2-y^2-z^2"
        bboxCenter 0 0 0
        bboxSize 1.4 1.4 1.4
        resolution [75 75 75]
}
  appearance FAppearance { material FMaterial {
        diffuseColor "r=1; g=abs(sin(u*pi)); b=0;"  }    }
}
```

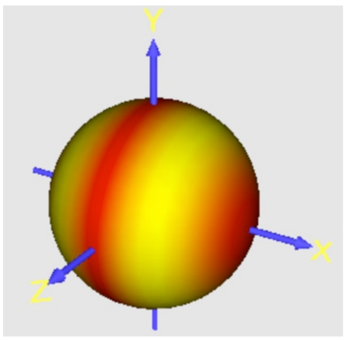


Figure 2. Modeling shape by consecutive definition of its geometry and color. The code is available in file sphere.wrl.

Defining complex shapes, usually assumes using multiple formulas and temporary variables. This requires a script-like mathematical language. FVRML emulates a subset of JavaScipt in which all variables, arrays and constants have only one type *float.* The following mathematical functions are implemented: *abs(x), fabs(x), sqrt(x), exp(x), log(x), sin(x), cos(x), tan(x), acos(x), asin(x), atan(x), ceil(x), floor(x), round(x), max(x,y), min(x,y), atan2(y,x), mod(x,y), cosh(x), sinh(x), tanh(x), log10(x)*. There are also flow control operators: *for-loops*, *while-loops*, *do-while-loops*, *break*, *continue*, and *if-else*. When writing function scripts, implicit functions $f(x,y,z)=0$ and explicit functions $f(x,y,z)\geq 0$ have to be named '*function frep*'. Parametric functions for shapes have to be named '*function parametric_x*', '*function parametric_y*', and '*function parametric_z*'. Variables, x,y,z are reserved for Cartesian coordinates, while variables u,v,w are parametric coordinates. Variable t is reserved for defining the time.

FVRML files have to be defined with *.wrl* extension, as the common VRML files. They are edited and visualized in the same way as VRML files. If only a black background color is displayed when you click at the file name, most likely you have made a syntax error when defining the object. Right click at the screen and select *Settings / Console*. It will show a console window where the error messages are sent.

4. EXPERIMENTS

There are 5 lab sessions comprising 5 experiments which will end with one final assessment after the 5th lab session. Refer to Section 5 for the assessment criteria.

4.1 Experiment 1: Visualization using polygons

Since this experiment will be done before the respective modules 3 and 4 are lectured, you are expected to do a self-study based on the introduction section of this manual as well as the text-book Chapter 3, 14.

This assignment will be evaluated. Do the following exercises:

1. *Display a simple polygon mesh as it is illustrated in Fig. 3 (download* ***polygons.wrl****)*
2. *Explore different Graphics Modes of the VRML browser (Wireframe, Vertices, Flat). Make sure OpenGL is selected in Settings/Renderer when you right-click at the VRML browser window (See Fig. 1b).*
3. *Examine how the color of the shape defined in* ***diffuseColor*** *field can be changed. Note that the color values must be real numbers between 0 and 1. See what happens if the color values are less than 0 or greater than 1.*
4. *Change the displayed polygon mesh (a pyramid) to anything else by adding new vertices and polygons. Make a six-sided polygon (hexagon) and a cube.*
5. *Notice how the order of vertices changes the visible side of polygons.*
6. *Create a folder with name Lab1 and copy there all the FVRML files you have experimented with.*
7. *Write a brief report explaining what each file defines and also copy it to Lab1 folder.*

```
#VRML V2.0 utf8
#polygon mesh example: a pyramid

Shape {
      appearance Appearance{ material Material {
                       diffuseColor    1 0 0  #red=1, green=0, blue=0
                       specularColor   1 1 1    #red=1, green=1, blue=1
                       transparency 0   # try values between 0 and 1
                       shininess 1      # shiny surface, try values between 0 and 1
                               }  }
      geometry IndexedFaceSet {
              coord Coordinate {   point [
                               # bottom vertices:
                               -1.0  -1.0   1.0,  #vertex 0
                               1.0   -1.0   1.0,  #vertex 1
                               1.0   -1.0  -1.0,  #vertex 2
                               -1.0  -1.0  -1.0,  #vertex 3
                               # top vertex:
                               0.0    1.0   0.0   #vertex 4       ] }
              coordIndex [
                       #bottom square
                        0,  3,  2,  1,  -1,
                       #side1
                        0,  1,  4, -1,
                       #side2
                        1,  2,  4, -1,
                       #side3
                        2,  3,  4, -1,
                       #side4
                        3,  0,  4, -1 ]
      }  }
```

Figure 3. Simple polygon object defined in VRML

4.2 Experiment 2: Parametric Curves

This assignment illustrates Module 3 and it serves a purpose to teach you how to visualize curves defined by parametric functions. Ideally, this experiment has to follow the lecture on parametric curves.

This assignment will be evaluated. Do the following exercises:

1. *Download file **curve.wrl** from the course-site (Fig. 4). Use it as a template for the following exercises.*
2. *Define parametrically in different files*
    - *straight line segment,*
    - *circle and its arc,*
    - *ellipse and its arc,*
    - *2D spiral,*
    - *3D helix.*
3. *Convert the explicitly defined curve y=sin(x) to parametric representation x(u), y(u) and define it in FVRML file.*
4. *Explore what happens when you change the curves resolutions to as little as 2 and see how the shape of the curves changes.*
5. *Change the curves parameter domain to see how they elongate or shorten.*
6. *Create a folder with name Lab2 and copy there all the FVRML files you have experimented with.*
7. *Write a brief report explaining what each file defines and also copy it to Lab2 folder.*

```
#VRML V2.0 utf8
EXTERNPROTO FGeometry [
 exposedField SFString definition
 exposedField MFFloat parameters
 ……………………………………..
    [ the rest of the EXTERNPROTO is skipped]
FShape {
# This definition is needed for drawing curves
polygonizer   "analytical_curve"

geometry FGeometry {
# The parametric formulae defining the curve.
# Change them to other formulae to see how geometry changes within the parameter domain
# and based on the sampling resolution defined below
definition "  x=1*(cos(2*pi*u))^3;
              y=1*(sin(2*pi*u))^3;
              z=0;"

# Domain for the parameter u.
# Explore how the curve changes when you change the domain values.
parameters [0 1]
# Sampling resolution along the curve. It defines how many times the parameter domain is
# sampled to calculate the curve function.
# Explore how the shape and the rendering speed change when you reduce or increase
# the resolution.
resolution [100]
 }
appearance FAppearance {
material FMaterial {
# Fixed red color is defined for the curve.
diffuseColor "r=1; g=0; b=0;"
   }    }
}
```

Figure 4. FVRML template of parametric curve

4.3 Experiment 3: Parametric Surfaces and Solids

This assignment mostly illustrates Module 3 and it serves a purpose to teach you how to visualize surfaces and solids defined by parametric functions. Ideally, this experiment has to follow the lecture on parametric surfaces and solids.

This assignment will be evaluated. Do the following exercises:

1. *Download file **surface.wrl** (Fig. 5) and **solid.wrl** (Fig. 6) from the course-site. Use them as templates for the following exercises.*
2. *Define parametrically in separate files*
    - *3D plane,*
    - *3D triangle,*
    - *bilinear surface,*
    - *sphere,*
    - *ellipsoid,*
    - *cone.*
3. *~~Explore how the shapes change when their sampling resolution is changed.~~*
4. *Define parametrically in separate files*
    - *solid box,*
    - *solid sphere,*
    - *solid cylinder,*
    - *solid cone.*
5. *Consider conversion of any closed surface into a solid object by introducing the third parameter. For example, convert a cylindrical surface into a solid cylinder.*
6. *Study how to make surfaces and solids using the concept of translational and rotational sweeping.*
7. *Use curve y=sin(x) for making a solid by applying rotational and translational sweepings together.*
8. *Create a folder with name Lab3 and copy there all the FVRML files you have experimented with.*
9. *Write a brief report explaining what each file defines and also copy it to Lab3 folder.*

```
FShape {
geometry FGeometry {

# The parametric formulae defining the surface.
# Change them to some other formulae to see how the surface geometry changes
# based on the parameters domain and the sampling resolution defined below
definition " x=u;
             y=v;
             z=0;"

# The parameters domain. Explore what happens when you make it
# smaller or bigger
parameters [0 1 0 1]

# Sampling resolution in parameters u and v.
# This is how the parameters domain is sampled to calculate the
# geometry function.
# Explore how the shape and the rendering speed change when you
# reduce or increase the resolution.
resolution [75 75]

 }

appearance FAppearance {
material FMaterial {
# Fixed green color is defined for the surface
diffuseColor "r=0; g=1; b=0;"
   }    }
}
```

Figure 5. FVRML template of parametric surface

```
FShape {
geometry FGeometry {

# The parametric formulae defining the solid.
# Change them to some other formulae to see how the solid geometry changes
# based on the parameters domain and the sampling resolution defined below
definition " x=u;
             y=v;
             z=w;"

# The parameters domain. Explore what happens when you make it
# smaller or bigger
parameters [0 1 0 1 0 1]

# Sampling resolution in parameters u and v.
# This is how the parameters domain is sampled to calculate the
# geometry function.
# Explore how the shape and the rendering speed change when you
# reduce or increase the resolution.
# For solid objects the resolution has to be the same in all three dimensions.
resolution [75 75 75]

 }

appearance FAppearance {
material FMaterial {
# Fixed green color is defined for the surface
diffuseColor "r=0; g=1; b=0;"
   }    }
}
```

Figure 6. FVRML template of parametric solid

4.4 Experiment 4: Implicit Solids

Assignment:

This assignment illustrates Modules 3 and 4 and it serves a purpose to teach you how to visualize solids defined by implicit functions. Ideally, this experiment has to follow the lecture on Constructive Solid Geometry.

This assignment will be evaluated. Do the following exercises:

1. *Download file **CSGsolid.wrl** from the course-site (Fig. 7) and display the shape. Use it as a template for the following exercises.*
2. *Define a complex CSG solid shape using set-theoretic operations in **min/max** form on at least one plane halfspace, ellipsoid, cylinder, and cone. Using symbols of these operations (| and &) is not allowed. Note that min/max functions can take only two arguments.*
3. *Adjust the tight bounding box (nearly touching the shape) and the optimum resolution for your shape to render it within 5 seconds only.*
4. *Define in **FMaterial** field a variable diffuse color for the whole shape by writing functions $r(u,v,w)$, $g(u,v,w)$, $b(u,v,w)$ where $u=x$, $v=y$, and $w=z$. Make sure the color values are correct (within [0,1]) on the visible surfaces of the shape and the shape rendering is still interactive.*
5. *Create a folder with name Lab4 and copy there all the FVRML files you have experimented with.*
6. *Write a brief report explaining what each file defines and also copy it to Lab4 folder.*

```
FShape {
geometry FGeometry {

# Function script defining the CSG solid.
# Change to some other formulae to see how the solid geometry changes
# based on the parameters domain and the sampling resolution defined below
definition " function frep(x,y,z,t){
             shape1=0.7^6-x^6-y^6-z^6;
             shape2=0.25^2-x^2-y^2;
             final=min(shape1, -shape2);
             return final;}"

# Adjust the tight bounding box and an optimal resolution
bboxCenter 0 0 0
bboxSize 2 2 2
resolution [100 100 100]

 }

appearance FAppearance {
material FMaterial {
# Variable color is defined for the CGS solid
diffuseColor "r=1; g=(v+1)/2; b=0;"
   }    }
}
```

Figure 7. FVRML template of CSG solid

4.5 Experiment 5: Morphing

Assignment:

This assignment illustrates Modules 3 and 5 and it serves a purpose to teach you how, given two parametric formulas defining some surfaces, you can define an animated transformation (morphing) between these surfaces. Ideally, this experiment has to follow the lecture on Motions.

This assignment will be evaluated. Do the following exercises:

1. *Download file **morphing.wrl** from the course-site (Fig. 8) and display the animated surface. Use the FVRML code as a template for the following exercises.*
2. *With reference to Table 1, you have to calculate your formula numbers according to the following algorithm:*

```
Formula_number_1 = Your_number_in_the_attendance_list;
While Formula_number_1 > 26 { Formula_number_1 = Formula_number_1 – 26};
Formula_number_2 = Formula_number_1+Numeric_part_of_your_group_name;
While Formula_number_2 > 26 { Formula_number_2 = Formula_number_2 - 26;}
```

3. *For Your_number_in_the_attendance_list refer to the attendance list where you mark your attendance in the lab. The Numeric_part_of_your_group_name is a decimal number of your group (e.g., S1: 1, S2: 1, BCG2: 2, etc).*

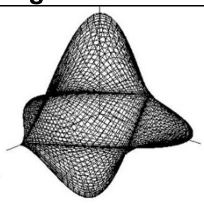
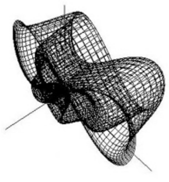
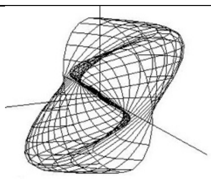
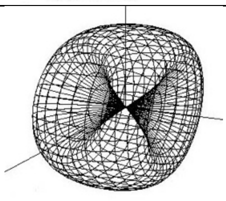
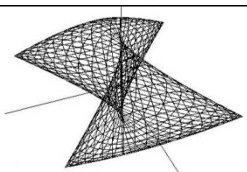
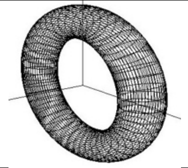
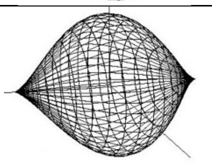
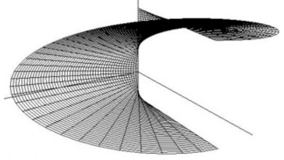
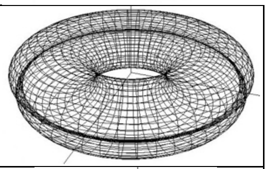
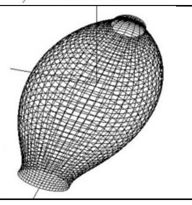
10. *Display the surface defined by Formula_number_1.*

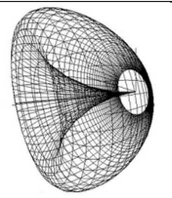
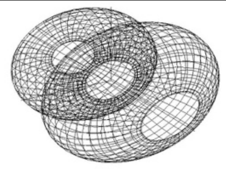
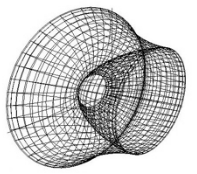
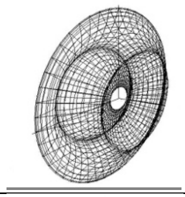
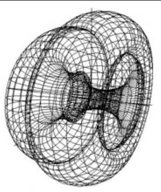
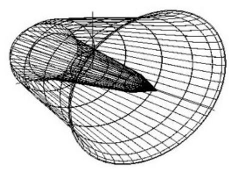
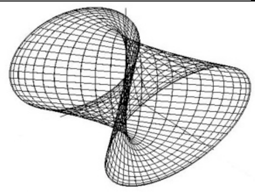
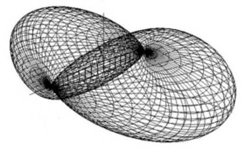
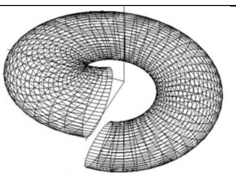
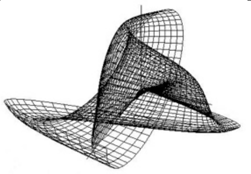
4. *Display the surface defined by Formula_number_2.*
5. *Modify the formulae so that they use parameters u and v in the same ranges, e.g., u=[0,1] and v=[0,1] for both surfaces.*
6. *Define one animated shape (with enabled animation) with a linear morphing function, as in Fig. 8.*
7. *Adjust the resolution parameters of the FShape node to achieve smooth appearance and jitter-free animation. The resolution parameters define the number of polygons which will be created when the shape is rendered. The higher the resolution you choose, the more polygons will be created which improves the shape quality but may result in slowing down the rendering speed. Therefore, the reasonable balance of the appearance and animation quality must be achieved by selecting the resolution parameters. The scene must be rendered (appear on the screen) within 2 sec.*
8. *Create a folder with name Lab5 and copy there all the FVRML files you have experimented with.*
9. *Write a brief report explaining what each file defines and also copy it to Lab5 folder. Write there also your group and attendance numbers as well as the surface numbers which you worked with.*

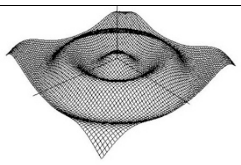
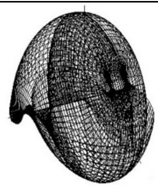
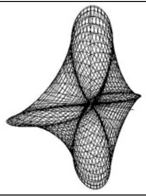
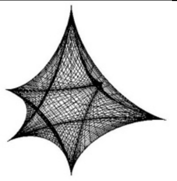
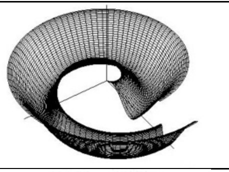
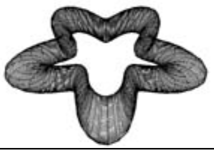
```
FShape {
# Enabling cycled animation
loop TRUE
# Mapping the interval of the internal time t=[0,1] to the actual time in sec.
cycleInterval 7

geometry FGeometry {
resolution [30 30]
parameters [-1 1 -1 1]
# Definition of the animated linear transformation (morphing)
# of one surface defined by       x1(u,v), y1(u,v), z1(u,v)
# to another surface defined by x2(u,v), y2(u,v), z2(u,v)
definition "
function parametric_x(u,v,w,t)
{ x1=cos(u*pi/2)*cos(v*pi);
  x2=(cos(u*pi/2))^3*(cos(v*pi))^3;
# linear morphing function for x coordinate
  return x1+(x2-x1)*t;        }

function parametric_y(u,v,w,t)
{ y1=cos(u*pi/2)*sin(v*pi);
  y2=(cos(u*pi/2))*(sin(v*pi))^3;
# linear morphing function for y coordinate
  return y1+(y2-y1)*t;       }

function parametric_z(u,v,w,t)
{ z1=sin(u*pi/2);
  z2=0.6*(sin(u*pi/2))^5;
# linear morphing function for z coordinate
  return z1+(z2-z1)*t;        }"
 }

appearance FAppearance {
material FMaterial {
diffuseColor "r=1; b=0; g=0;"
}    }
}
```

Figure 8. FVRML template of parametric morphing

Table 1

#	formulas	parameters	Image
1	$x = 1.6(\cos(\varphi))^3$ $y = 1.6(\cos(\theta)\sin(\varphi))^3$ $z = 1.6\sin(\theta)\sin(\varphi)$	0≤θ≤2π, 0≤φ≤π	
2	$x = 1.5a\cos(\theta)$ $y = 1.5a\sin(\theta)\cos(\theta)$ $z = 1.5a(\sin(2a\pi))^5$	0≤θ≤2π, 0≤a≤1	
3	$x = b\cos(\alpha)$ $y = b\sin(\alpha)$ $z = b\sin(2b\pi)\sin\alpha$	0≤α≤2π, 0≤b≤1	
4	$x = \cos(\varphi)\sin(\varphi)$ $y = \cos(a\pi)\sin(\varphi)$ $z = \sin(a\pi)\sin(\varphi)$	0≤a≤2, 0≤φ≤π	
5	$x = \cos(\varphi)$ $y = \cos(2\theta)\sin(\varphi)$ $z = \sin(2\theta)\cos(\varphi)$	0≤θ≤π, 0≤φ≤π	
6	$x = (1+0.25\cos(\theta))\cos(b\pi)$ $y = (1+0.25\cos(\theta))\sin(b\pi)$ $z = 0.25\sin(\theta)$	0≤θ≤2π, 0≤b≤2	
7	$x = \cos(0.5\theta)(\sin(\varphi))^3$ $y = \sin(0.5\theta)(\sin(\varphi))^3$ $z = \cos(\varphi)$	0≤θ≤4π, 0≤φ≤π	
8	$x = 0.5b\pi\cos(2\theta)$ $y = 0.5\theta - 0.5$ $z = 0.5b\pi\sin(2\theta)$	0≤θ≤π, 0≤b≤1	
9	$x = 0.5\cos(4\theta)(-3+\cos(\varphi)(1+\cos(\varphi)))$ $y = 0.5\sin(\varphi)(1+\cos(\varphi))$ $z = 0.5\sin(4\theta)(-3+\cos(\varphi)(1+\cos(\varphi)))$	0≤θ≤π/2, 0≤φ≤2π	
10	$x = 0.5((\varphi - 0.5\sin(\varphi)) - 3)$ $y = 0.5\cos(4a\pi)(1-0.5\cos(\varphi))$ $z = 0.5\sin(4a\pi)(1-0.5\cos(\varphi))$	0≤α≤0.5, 0≤φ≤2π	

11	$x = 2.5\varphi/(1+\varphi^3)$ $y = 2.5\cos(\theta)\varphi^2/(1+\varphi^3)$ $z = 2.5\sin(\theta)\varphi^2/(1+\varphi^3)$	$0 \le \theta \le 2\pi,\ 0 \le \varphi \le 2\pi$	
12	$x = 0.25(\varphi - \sin(2\varphi) - 2\pi)$ $y = 0.5\cos(0.5\theta)(2-\cos(2\varphi))$ $z = 0.5\sin(0.5\theta)(2-\cos(2\varphi))$	$0 \le \theta \le 4\pi,\ 0 \le \varphi \le 2\pi$	
13	$x = 0.5(\cos(\varphi) + 0.5\cos(2\varphi))$ $y = 0.5\cos(6\theta)(2+\sin(\varphi) - 0.5\sin(2\varphi))$ $z = 0.5\sin(6\theta)(2+\sin(\varphi) - 0.5\sin(2\varphi))$	$0 \le \theta \le \pi/3,\ 0 \le \varphi \le 2\pi$	
14	$x = 0.1(3\cos(\varphi/3) + 0.8\cos(\varphi))$ $y = 0.2\cos(\theta)(5+3\sin(\varphi/3) - 0.8\sin(\varphi))$ $z = 0.2\sin(\theta)(5+3\sin(\varphi/3) - 0.8\sin(\varphi))$	$0 \le \theta \le 2\pi,\ 0 \le \varphi \le 6\pi$	
15	$x = 0.15(4\cos(\varphi/4) - \cos(\varphi))$ $y = 0.15\cos(\theta)(6+4\sin(\varphi/4) - \sin(\varphi))$ $z = 0.15\sin(\theta)(6+4\sin(\varphi/4) - \sin(\varphi))$	$0 \le \theta \le 2\pi,\ 0 \le \varphi \le 8\pi$	
16	$x = \cos(2\pi\sin(\varphi))$ $y = \cos(4\pi\theta)\sin(\varphi)$ $z = \sin(4\pi\theta)\sin(\varphi)$	$0 \le \theta \le 0.5,\ 0 \le \varphi \le \pi$	
17	$x = \sin(\theta/6)$ $y = \cos(\varphi)$ $z = \sin(\theta/6)\sin(\varphi)$	$0 \le \theta \le 12\pi,\ 0 \le \varphi \le 2\pi$	
18	$x = (\cos(0.25\theta) + 1)\cos(\varphi)$ $y = \sin(0.25\theta)\cos(\varphi)$ $z = \sin(\varphi)$	$0 \le \theta \le 8\pi,\ 0 \le \varphi \le 2\pi$	
19	$x = 0.5(\cos(\varphi) + 2)\cos(8\theta)$ $y = 0.5(\sin(\varphi) + 16\theta/\pi - 2)$ $z = -0.5(\cos(\varphi) + 2)\sin(8\theta)$	$0 \le \theta \le \pi/4,\ 0 \le \varphi \le 2\pi$	
20	$x = 2\theta\sin(\varphi)$ $y = 0.2\varphi\sin(10\theta)$ $z = 0.2\varphi\cos(10\theta)$	$0 \le \theta \le \pi/5,\ 0 \le \varphi \le 2\pi$	

21	$x = 1.5(2a-1)$ $y = 0.15\cos\left(5\sqrt{(4a-2)^2+(4b-2)^2}\right)$ $z = 1.5(2b-1)$	0≤a≤1, 0≤b≤1	
22	$x = 1.5\cos(\theta)\sin(\varphi)$ $y = 1.5\sin(\theta)\sin(\varphi)$ $z = 0.15\sqrt{\theta^2+\varphi^2}\cos(\varphi)$	0≤θ≤2π, 0≤φ≤2π	
23	$x = 2(\cos 8\theta)^3(\sin\varphi)^5$ $y = 2(\sin 8\theta)^3(\sin\varphi)^5$ $z = 2(\sin\varphi)^5\cos(\varphi)$	0≤θ≤π/4, 0≤φ≤π	
24	$x = 2(\cos\theta)^3(\sin 4\varphi)^3$ $y = 2(\sin\theta)^3(\sin 4\varphi)^3$ $z = 2(\cos 4\varphi)^3$	0≤θ≤2π, 0≤φ≤π/4	
25	$x = 0.2\sqrt{16\theta^2+4\varphi^2}\cos 2\varphi$ $y = 0.4\theta\cos 4\theta$ $z = -0.2\sqrt{16\theta^2+4\varphi^2}\sin 2\varphi$	0≤θ≤π/2, 0≤φ≤π	
26	$x = (1\text{-}0.3\sin(5b\pi))\cos(b\pi)(4+\cos(a\pi))$ $y = (1\text{-}0.3\sin(5b\pi))\sin(b\pi)(4+\cos(a\pi))$ $z = \sin(a\pi)$	−1≤a≤1, −1≤b≤1	

5. ASSESSMENT

The assessment will be based on the following criteria:

Labs 1-3: Polygons, Curves, Surfaces and Solids:

a	The submitted source code files and reports show an evidence of your extensive work towards learning polygons, parametric curves, parameterization of explicit curves, as well as definition of surfaces and solids by parametric functions	Up to 40

Lab 4: Implicit Fantasies:

a	An object is defined by implicit functions with at least one union, intersection and difference operations defined by min/max functions in the function definition. The geometry of the shape is created using at least one ellipsoid, cylinder, cone, and plane halfspace. The scene is rendered within 5 sec.	Up to 15
b	A variable color of the implicitly defined object is defined by color functions $r(u,v,w)$, $g(u,v,w)$, $b(u,v,w)$. The colors values are within [0, 1] interval on the visible surfaces of the object.	Up to 15

Lab 5: Parametric Metamorphoses:

a	Individual shapes involved in morphing are correct. The scene is rendered (appears on the screen) within 2 sec.	Up to 15
b	Correct and smooth morphing between shapes is implemented.	Up to 15

Total maximum marks: 100

All five assignments have to be submitted together electronically by the end of Week 13 (next Monday morning 9:00am).

To submit your assignments files for evaluation, do the following:

1. Create a folder and name it after yourself **exactly as your name is written in the group list**, e.g., ALEXEI SOURIN. Copy into this folder your lab folders Lab1, Lab2, Lab3, Lab4 and Lab5.

2. Zip the folder. The zipped file **must have the same name as your folder**, i.e. your name, e.g., ALEXEI SOURIN.zip.

3. Go to the course-site. Click at **Assignments** button click at **Visual Mathematics**. In **Attach File** click at **Browse My Computer** and select your zipped file**.** Click at **Submit** button**.** If your submission will not be received by the deadline, the lab supervisors and/or course coordinator will send you a warning email, therefore read your emails every day on Week 14.

6. REFERENCES

1 BS contact VRML/X3D *http://www.bitmanagement.com*
2. Function-based Extension of VRML, *http://www.ntu.edu.sg/home/assourin/FVRML.htm*
3. VRMLpad, *http://www.parallelgraphics.com/products/vrmlpad/download*
4. VRML and X3D specifications *http://www.web3d.org/standards/all/*