LABORATORY MANUAL

CZ2003: Computer Graphics and Visualization
SW Lab or Personal Computers

Making Images with Mathematics

Lab Experiments 1 - 5

SESSION 2020/2021
SEMESTER 2
COMPUTER SCIENCE COURSE

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
NANYANG TECHNOLOGICAL UNIVERSITY

MAKING IMAGES WITH MATHEMATICS

1. OBJECTIVE

In this coursework you will learn how to visualize curves, surfaces and solid shapes defined by simple mathematical formulas. Each of 5 experiments takes 2 hours. Upon completion of these experiments you will know:

- How to define shapes by implicit, explicit and parametric analytic functions
- How to make shape morphing transformations and motions, and
- How to color shapes using mathematical functions.

2. EQUIPMENT

In the lab, the following software tools are installed on Windows PC:

- *BS Contact VRML/X3D* viewer[1] for displaying VRML shapes,
- *FVRML* plug-in[2] extending VRML with new features,
- *VrmlPad* editor and viewer.

All these tools can be also installed on your personal Windows computers from the links provided in the course-site. In addition, you can also install *Shape Explorer* visualization software which can run on both Windows and Mac. **You therefore can work on the assignments using your own computer rather than in the lab.**

3. INTRODUCTION

You will use two visualization software tools: FVRML function-based extension of VRML and Shape Explorer. The minimum requirement of the coursework is to concentrate on visualization principles—mathematical definitions, coordinate domains and sampling resolutions—and therefore to use the provided source code templates to only fill them in with mathematical formulae and visualization parameters. However, interested students are encouraged to use VRML and FVRML for further learning of the programming principles of computer graphics and visualization.

3.1 VRML, X3D and FVRML

Virtual Reality Modeling Language (VRML) and its current successor Extensible 3D (X3D)[3]. are ISO* standard file formats and programing languages 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* which tells the computer *what to do*. It differs from the *imperative programming style*, like in C and in a popular graphics library OpenGL, which tells the computer *how to do* things.

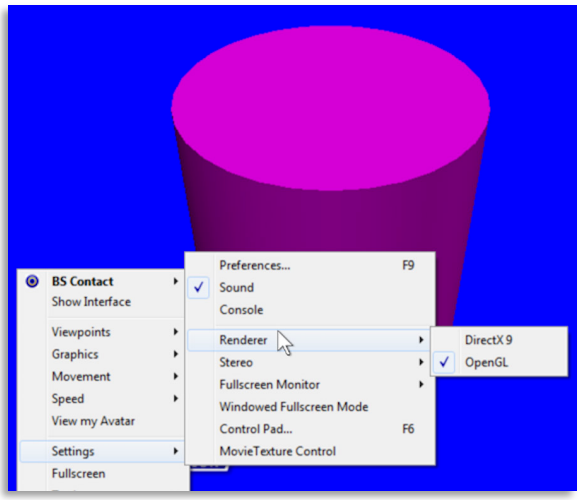
We will use VRML. The rationale for this is that both VRML and X3D follow the same programming principles and VRML coding is one of two formats used in X3D. VRML is easier to read than the other format which is XML. Also, VrmlPad is a convenient visualization tool licensed to our School.

* The International Organization for Standardization (ISO) is an international standard-setting body composed of representatives from various national standards organizations.

Let's consider how to create a simple VRML scene with one object: a shiny purple cylinder with a radius 3, and a height 6 on a blue background. The VRML code defining it is given 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 domain [0,1]. In the example, purple and shiny material is defined. Finally, a light source and a background color are defined.

```
#VRML V2.0 utf8
# Transform node may define 3 transformations: scaling, rotation and translation. They will
# be applied in exactly this order regardless the order in which they are written in the node
# Transform nodes can be nested to add more transformations and to change their order.
Transform {
        rotation  1 0 0 0.7
        translation         0 0 0
        scale 1 1 1
        children[

# Shape node includes geometry and appearance 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.

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 install *BS Contact* viewer. Once you click at the *.wrl* file, the VRML shape defined in Figure 1a will be displayed as it is shown in Figure 1b. Alternatively, you can use VRMLPad both for editing and visualization of VRML files.

VRML and its current successor Extensible 3D (X3D) are very versatile programming tools but they only visualize a few predefined shapes (sphere, cylinder, cone, box) and so-called polygon meshes—the shapes defined by polygons. Definitions of the polygons' vertices by their coordinates is a tedious task. Usually, third party software tools are used to generate these coordinates.

In this course, you learn how to define geometric shapes by mathematical formulas which are used to compute 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*** 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 writing mathematical formulae directly in the VRML code. Lecture Module 3 introduces how to use mathematics for defining geometry. Below is a brief summary.

For defining geometry, appearance and their transformations, three types of functions can be used concurrently in FVRML. 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 are equal to zero for the points located on the surface. Hence, a sphere can be defined by equation: $R^2 - x^2 - y^2 - z^2 = 0$. In the FVRML code you will only provide the left part of the equality. By changing this equality into an inequality
$g = f(x,y,z,t) \geq 0$, known in computer graphics as *FRep*, we define not only a surface but the space bounded by this surface, or a half-space. In this case, the function equals to zero for the points located on the surface of the object, positive values of the function indicate points inside the solid object, and negative values are for the points which are outside the object. To illustrate this, 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 for making variable time-dependent shapes. For example, a sphere bouncing up and down by height a during time $t = [0,1]$ can be defined as $g = R^2 - x^2 - (y - a\ sin(t\pi))^2 - z^2 \geq 0$. Implicit functions can efficiently represent Set-theoretic (Boolean) operations. In FVRML code only the left part of the inequality (FRep) has to be written.

Explicit functions define one coordinate or a value as a function of the others. It is seldom used for drawing since the explicit representations often create axis dependency, i.e. there may be no only one representation available for any orientation of the shape or multivalued functions will be used in the representation which is not desirable case for visualization algorithms. Therefore, for defining geometry of shapes in FVRML the explicit functions are only used as individual parametric functions.

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:

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

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:

$$
\begin{aligned}
x &= R\cos v\ \cos u \\
y &= R\sin u + a\sin t\pi \\
z &= R\sin v\cos u \\
r &= \sin t\pi \\
g &= 1 - \sin t\pi \\
b &= 0 \\
u &= [0, 2\pi], \qquad v = [0, \pi], \qquad t = [0, 1].
\end{aligned}
$$

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-centered 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;" }   }
}
```

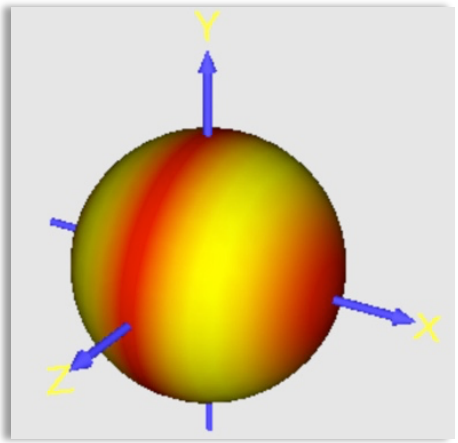


Figure 2. Modeling shape by consecutive definition of its geometry and color. The code is available in file sphere.wrl.

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: $abs(x)$, $fabs(x)$, $sqrt(x)$, $\exp(x)$, $\log(x)$, $\sin(x)$, $\cos(x)$,

$tan(x), acos(x), asin(x), atan(x), ceil(x), floor(x), atan2(y,x), mod(x,y), round(x), max(x,y), min(x,y), cosh(x), sinh(x), tanh(x), log10(x)$. 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.

All these FVRML can only be run on computers with Windows OS. Mac users have to use either Boot Camp or Virtual Machines (Parallels Desktop, VMWare, VirtualBox) with Windows.

3.2 Shape Explorer

Shape Explorer (Figure 3) is an interactive software tool designed to use the same function-based definitions for shapes: implicit and parametric formulas, domains and sampling resolutions. It is designed to run both on Windows and Mac in support of FVRML with a goal to eventually replace it. However, it is still work in progress. Shape Explorer is just one interactive window where you can type the function definitions and other parameters. The shape definitions can be saved in a proprietary format and loaded later to the software to continue working with it. Migration between Shape Explorer and FVRML can be done by copy-pasting the formulae, domains, resolutions and colors. Shape Explorer supports only one shape visualization at the time. It does not provide any transformations and other advanced features available in VRML. Its purpose is to work as a quick all-in-one multi-platform visualization tool.

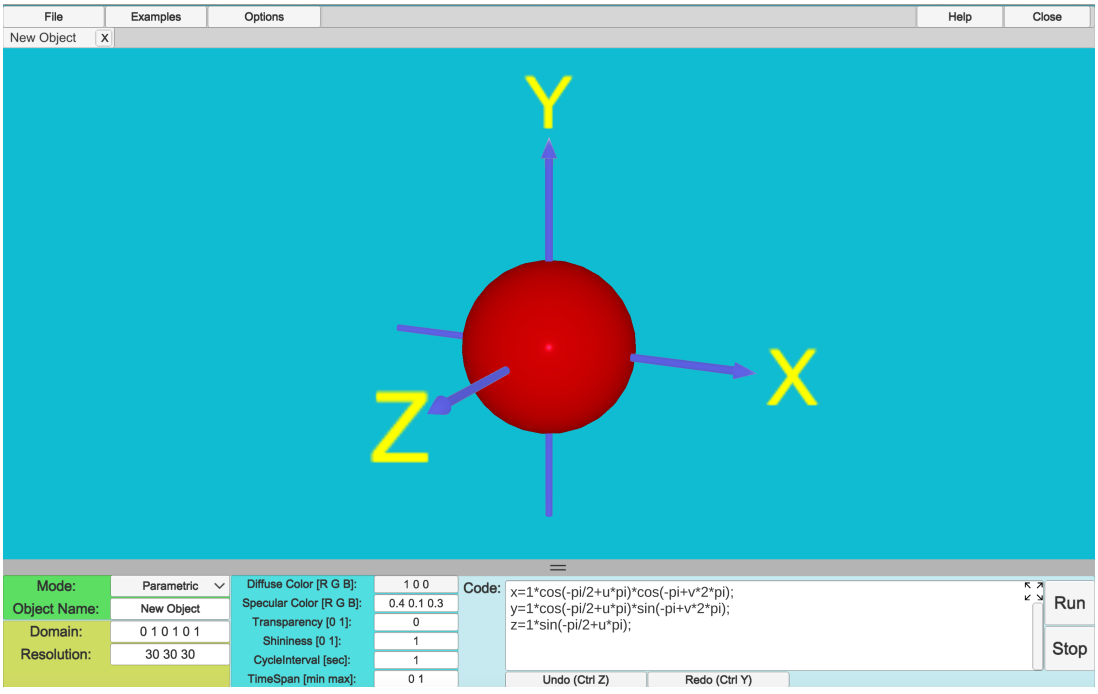


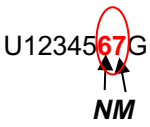
Figure 3. Shape Explorer.

4. EXPERIMENTS

There are 5 lab sessions comprising **5 experiments**:

1.	Parametric Curves	12 marks
2.	Parametric Surfaces	12 marks
3.	Parametric Solids	12 marks
4.	Implicit Surfaces and Solids	12 marks
5.	Transformations and Motions	12 marks
	Total:	**60 marks**

All the experiments are personalized, i.e. each student will have different data to work with. The personalization is based on using two last digits of your matriculation number:



which can be integer numbers from 0 to 9 where 0 will stand for 10. Therefore, the two numbers from 1 to 10 will define your personal variant of the assignments. These numbers will be further referred to as ***N*** for penultimate digit and ***M*** for the last digit.

This is an individual assignment. Any group work is not allowed. In case of plagiarism, all the involved parties will be failed without investigating who copied from whom.

In the remaining part of the manual, you will find the assignment instructions. Each of the five lab assignments will be evaluated and awarded up to 12 marks. Partial marks are indicated in the assignment instructions.

After completion of each of five lab assignment, you have to write a report in which you:

1. Copy screenshots of the displayed shapes;
2. Write their function definitions, domains, resolutions, other parameters;
3. Write the names of the respective FVRML or Shape Explorer files (further referred to as ***files***);
4. Write brief descriptions of your experiments with the shapes and observations made.

The content and quality of the report contributes to the assignment marks.

WITHIN ONE WEEK (7*24 HOURS) AFTER THE END OF EACH OF FIVE SCHEDULED LAB SESSION you have to do the following:

1. Create a folder and name it **exactly as your name is written on your matriculation card and add as a suffix the two last matric number NM**, e.g., JAMES BOND 67.
2. Copy to this folder the scan/photo of your matriculation card with clearly readable name, photo, and at least three last characters of the matriculation number.
3. Copy to this folder all the relevant files and PDF files of the report.
4. Zip your assignment folder. The zipped file **must have the same name as your folder**, e.g., JAMES BOND 67.zip.
5. Submit the zipped file through the respective digital drop box in the course site.
6. Check your email box regularly. The lab instructors or subject coordinator will email you if something is wrong.

4.1 Experiment 1: Parametric Curves

This assignment illustrates Module 3, and it serves a purpose to teach you how to visualize curves defined by parametric functions. To work on this assignment, you have to watch the following TEL lectures:

Module 1: Lecture 2 (Part 3) - Introduction to Computer Graphics and Foundation Mathematics {Rene Descartes and coordinate systems}
Module 3: Lecture 1 (Part 2/3) - Geometric Shapes: 2D Curves {straight-lines}
Module 3: Lecture 1 (Part 3/3) - Geometric Shapes: 2D Curves {straight-lines}
Module 3: Lecture 2 (Part 1/3) - Geometric Shapes: 2D Curves {circle}
Module 3: Lecture 2 (Part 2/3) - Geometric Shapes: 2D Curves {circle and beyond}
Module 3: Lecture 2 (Part 3/3) - Geometric Shapes: 2D Curves {ellipse and summary}
Module 3: Lecture 3 - Geometric Shapes: 3D Curves

Assignment instructions:
Create folder **Lab1**. Download into it from the course-site the files **ParametricCurve.wrl** (Fig. 4) and **CoordinateAxes.wrl**. Use **ParametricCurve.wrl** as a template for the following exercises. For each of the curves, you have to select a minimally sufficient sampling resolution providing for smooth curve visualization so that any further reduction of it will visually reveal the polyline interpolation of the curve.

1. Define parametrically in 4 separate files using functions $x(u), y(u), u \in [0,1]$ and display:

   a. Straight line segment spanning from the point with coordinates (-***N***,-***M***) to the point with coordinates (***M***,***N***).

   b. A circular arc with radius ***N***, centered at point with coordinates (***N***,***M***) with the angles $[\frac{\pi}{\boldsymbol{N}}, 2\pi]$.

   c. Origin-centered 2D spiral curve which starts at the origin, makes ***N***+***M*** revolutions clockwise and reaches eventually the radius 2****M****.*

   d. 3D cylindrical helix with radius ***N*** which is aligned with axis *Z*, makes ***M*** counterclockwise revolutions about axis *Z* while spanning from $z_1 = -\boldsymbol{N}$ to $z_1 = \boldsymbol{M}$.

   (4 marks)

2. With reference to Table 1, convert the explicitly defined curve number ***M*** to parametric representations $x(u), y(u), u \in [0,1]$ and display it. Note that sketches of the curves in Table 1 are done not to the actual scale since the values of ***N*** and ***M*** are different in each variant.

   (4 marks)

3. With reference to Figure 5, a curve is defined in polar coordinates by:
   $$r = \boldsymbol{N} - (\boldsymbol{M} + 5)\cos\alpha \quad \alpha \in [0,2\pi]$$
   Define the curve parametrically as $x(u), y(u), u \in [0,1]$ and display it.

   (4 marks)

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

# External VRML object "Coordinate Axes" is included in the scene.
# The size of the axes can be changed by the scale transformation
Transform {
            scale 1.2 1.2 1.2 children [
                Inline {url "CoordinateAxes.wrl"}  ]}

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;"
  }   }
}
```

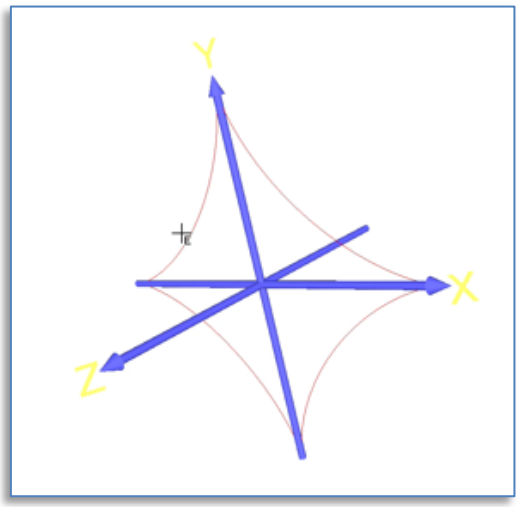
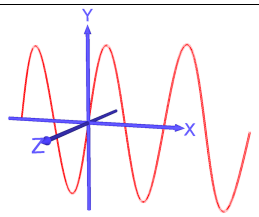
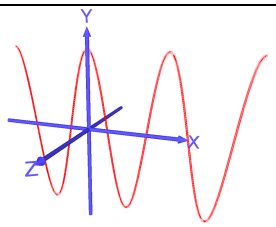
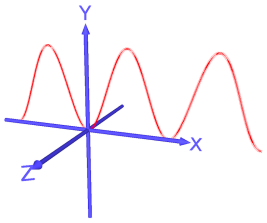
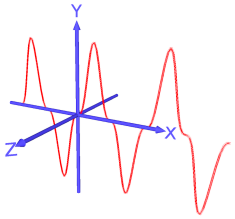
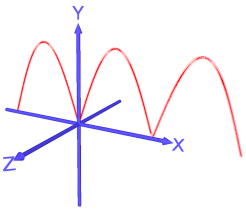
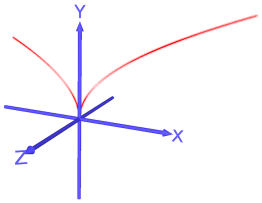
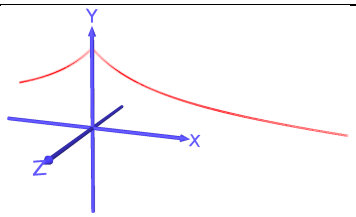
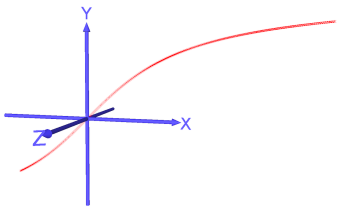
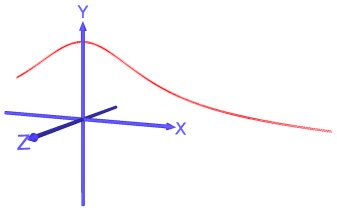
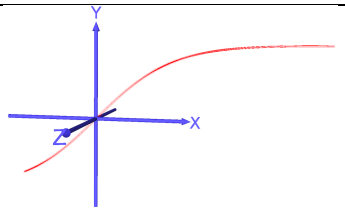


Figure 4. FVRML template of parametric curve. The code is in ParametricCurve.wrl.

Table 1. Curves defined explicitly.

M	Explicit formula	Sketch (not to the scale and with 3 full oscillations)
1	$y = \sin x$ The curve has to make ***N*** full oscillations within $x \in [-N, 2N]$	
2	$y = \cos x$ The curve has to make ***N*** full oscillations within $x \in [-N, N]$	
3	$y = (\sin x)^2$ The curve has to make ***N*** full oscillations within $x \in [-2N, N]$	
4	$y = (\sin x)^3$ The curve has to make ***N*** full oscillations within $x \in [-N, 3N]$	
5	$y = \lvert\sin x\rvert$ The curve has to make ***N*** full oscillations within $x \in [-1.5N, 2N]$	
6	$y = \sqrt{\lvert x\rvert}$ $\quad x \in [-N, 1.5N]$	
7	$y = \dfrac{N}{\sqrt{\lvert x\rvert + 1}}$ $\quad x \in [-1.5N, 2.5N]$	

8	$y = atan\, x \quad x \in [-0.7N, 2N]$	
9	$y = \frac{N}{x^2+1} \quad x \in [-N, 1.8N]$	
10	$y = \tanh x \quad x \in [-1.3N, 2N]$	

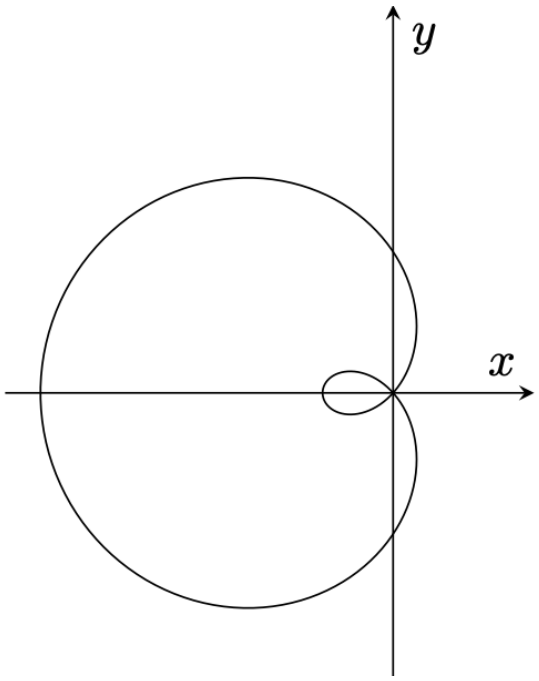


Figure 5. A polar curve called Limaçon $r = N - (M + 5)\cos\alpha \quad \alpha \in [0, 2\pi]$. The actual shape of the curve is determined by the values of *N* and *M*.

4.2 Experiment 2: Parametric Surfaces

This assignment illustrates Module 3, and it serves a purpose to teach you how to visualize surfaces defined by parametric functions. To work on this assignment, you have to watch the following TEL lectures:

Module 3: Lecture 4 (Part 1/4) - Geometric Shapes: Surfaces {classification, polygon mesh}
Module 3: Lecture 4 (Part 2/4) - Geometric Shapes: Surfaces {plane}
Module 3: Lecture 4 (Part 3/4) - Geometric Shapes: Surfaces {plane parametrically and bilinear surface}
Module 3: Lecture 4 (Part 4/4) - Geometric Shapes: Surfaces {bilinear surfaces and summary}
Module 3: Lecture 5 (Part 1/2) - Geometric Shapes: Quadric Surfaces and Sweeping {sphere}
Module 3: Lecture 5 (Part 2/2) - Geometric Shapes: Quadric Surfaces and Sweeping {other quadrics and sweeping}

Assignment instructions:
Create folder **Lab2**. Download into it from the course-site the files **ParametricSurface.wrl** (Fig. 6) and **CoordinateAxes.wrl**. Use **ParametricSurface.wrl** as a template for the following exercises. For each of the surfaces, you have to select a minimally sufficient sampling resolution providing for smooth surface visualization so that any further reduction of it will visually reveal the polygonal interpolation of the surface. Also, the size of the polygons has to be as even as possible, i.e. you must avoid displaying elongated polygons.

1. In 4 separate files, define parametrically using functions $x(u,v), y(u,v), z(u,v)$, $u, v \in [0,1]$ and display:

   a. A plane passing through the points with coordinates (***N***, ***M***, 0), (0, ***M***, ***N***), (***N***, 0, ***M***).

   b. A triangular polygon with the vertices at the points with coordinates (***N***, ***M***, 0), (0, ***M***, ***N***), (***N***, 0, ***M***).

   c. An origin-centered ellipsoid with the semi-axes ***N***, ***M***, (***N***+***M***)/2

   d. A cylindrical surface with radius *N* which is aligned with axis Z, and spans from $z_1 = -\boldsymbol{N}$ to $z_1 = \boldsymbol{M}$.

   (4 marks)

2. Define parametrically using functions $x(u,v), y(u,v), z(u,v), u, v \in [0,1]$ a surface obtained by translational sweeping of the curve number ***M*** (Table 1) along axis *Z* so that it will span from $z_1 = -\boldsymbol{N}$ to $z_1 = \boldsymbol{M}$.

   (4 marks)

3. Define parametrically using functions $(u,v), y(u,v), z(u,v), u, v \in [0,1]$ a surface created by rotational sweeping of the curve defined in Fig. 5. The curve has to be placed in coordinate plane *ZY*, translated by (0, 0, -***N***) and then subjected to rotational sweeping about axis Y clockwise by angle $\frac{\pi}{N}$ with the offset angle $+\frac{3\pi}{2M}$.

   (4 marks)

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

# External VRML object "Coordinate Axes" is included in the scene.
# The size of the axes can be changed by the scale transformation
Transform {
            scale 1.2 1.2 1.2 children [
                Inline {url "CoordinateAxes.wrl"} ]}


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;"
  }  }
}
```

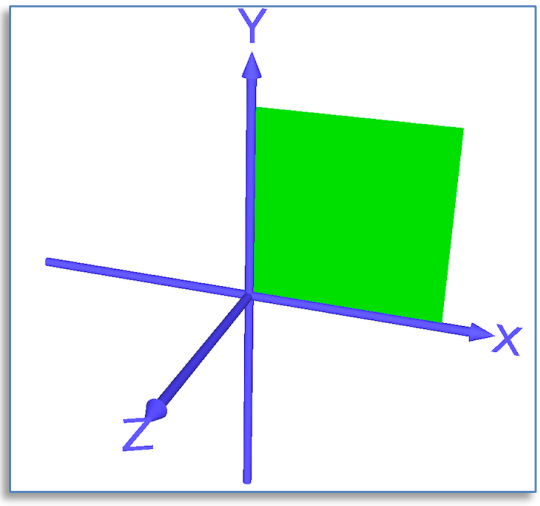


Figure 6. FVRML template of parametric surface. The code is in ParametricSurface.wrl

4.3 Experiment 3: Parametric Solids

This assignment illustrates Module 3, and it serves a purpose to teach you how to visualize surfaces defined by parametric functions. To work on this assignment, you have to watch the following TEL lectures:

Module 3: Lecture 6 (Part 1/2) - Geometric Shapes: Solid Objects {parametric solids}
Module 3: Lecture 6 (Part 2/2) - Geometric Shapes: Solid Objects {solids by sweeping}

Assignment instructions:
Create folder **Lab3**. Download into it from the course-site the files **ParametricSolid.wrl** (Fig. 7) and **CoordinateAxes.wrl**. Use **ParametricSolid.wrl** as a template for the following exercises. For each of the solids, you have to select a minimally sufficient sampling resolution providing for smooth solid surface visualization so that any further reduction of it will visually reveal the polygonal interpolation of the solid surface. Also, the size of the polygons has to be as even as possible, i.e. you must avoid displaying elongated polygons.

1. Define parametrically using functions $x(u,v,w),\ y(u,v,w),\ z(u,v,w),$ $u,v,w \in [0,1]$ in 4 separate files and display:

   a. A solid box with the sides parallel to the coordinate planes and the coordinates of two opposite vertices (***N***, 0, ***M***), (***N***+***M***, ***M,*** 2(***N+M)***).

   b. A solid three-sided pyramid with the vertices of the base with coordinates (0,0,0), (***N***,0,0), (0,0,***M***), and the apex at (0,***N+M***,0).

   c. A lower half of the origin-centered solid sphere with radius ***N***.

   d. An upper half of the torus which axis is the vertical axis Y. The radius of the torus tube is $\frac{\boldsymbol{N}}{5}$**.** The distance from axis Y to the center of the torus tube is ***N***.

   (4 marks)

2. Define parametrically using functions $x(u,v,w),\ y(u,v,w),\ z(u,v,w),\ u,v,w \in [0,1]$ a solid object created by translational sweeping of the surface obtained in experiment 2 (exercise 2) along Axis Y so that **its lowest point** moves from $y = -\boldsymbol{N}$ to $y = \boldsymbol{N} + \boldsymbol{M}$.

   (4 marks)

3. Define parametrically using functions $x(u,v,w),\ y(u,v,w),\ z(u,v,w),$ $u,v,w \in [0,1]$ a solid object created by filling in the surface defined in experiment 2 exercise 3).

   (4 marks)

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

# External VRML object "Coordinate Axes" is included in the scene.
# The size of the axes can be changed by the scale transformation
Transform {
            scale 1.2 1.2 1.2 children [
          Inline {url "CoordinateAxes.wrl"} ]}

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;"
  }  }
       }
```

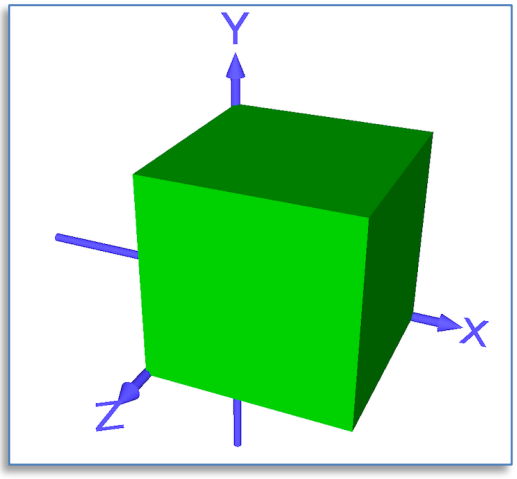


Figure 7. FVRML template of parametric solid. The code is in ParametricSolid.wrl

4.4 Experiment 4: Implicit Surfaces and Solids

This assignment illustrates Module 3, and it serves a purpose to teach you how to visualize surfaces and solid objects defined by implicit functions and using CSG operations. To work on this assignment, you have to watch the following TEL lectures:

Module 3: Lecture 7 (Part 1/3) - Geometric Shapes: Constructive Solid Geometry {from implicit to inequality}
Module 3: Lecture 7 (Part 2/3) - Geometric Shapes: Constructive Solid Geometry {CSG definitions}
Module 3: Lecture 7 (Part 3/3) - Geometric Shapes: Constructive Solid Geometry {examples}
Module 3: Lecture 8 - Geometric Shapes: Blobby Shapes
Week 11 - Visual Appearance: Illumination

Assignment instructions:
Create folder **Lab4**. Download into it from the course-site the files **ImplicitSurface.wrl, CSGsolid.wrl** (Figs. 8, 9) and file **CoordinateAxes.wrl**. Use **ImplicitSurface.wrl** and **CSGsolid.wrl** as templates for the following exercises. For each of the solids, you have to adjust the tight bounding box (nearly touching the shape), and select a minimally sufficient sampling resolution providing for smooth surface visualization so that any further reduction of it will visually reveal the polygonal interpolation of the surface. In addition, each shape has to be visualised within 5 seconds.

1. In 4 separate files, define by implicit functions $f(x, y, z) = 0$ and by setting a proper bounding box:

   a. A plane passing through the points with coordinates (***N***, ***M***, 0), (0, ***M***, ***N***), (***N***, 0, ***M***).

   b. A lower half of the surface of the origin-centered sphere with radius ***M***.

   c. A cylindrical surface with radius ***M*** which is aligned with axis Z, and spans from $z_1 = -\boldsymbol{N}$ to $z_1 = \boldsymbol{M}$.

   d. A two-side conical surface with radius ***M*** at distance 1 from its apex. The cone is aligned with axis Z, and spans from $z_1 = -1$ to $z_1 = 1$ with the cone apex located at the origin.

   (4 marks)

2. With reference to Table 2, build one complex shape using set-theoretic operations following the design sketch number ***M***. It has to be one function script created with MIN/MAX functions and functions $f(x, y, z) \geq 0$ of the participating shapes. Note that in FVRML each min/max function can take only two arguments and therefore nested functions have to be used.

   (4 marks)

3. **This exercise can only be done using FVRML.**
   Color the shape defined in exercise 2 with a function-defined red color $r = f(u, v, w)$, $g = 0, b = 0$ where $u = x,\ v = y,$ and $w = z$. As function f use function number ***M*** from Table 1 but scale it so that the values will be located within [0,1] on the visible front surface of the shape.

   (4 marks)

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

Background {skyColor 1 1 1}

# External VRML object "Coordinate Axes" is included in the scene.
# The size of the axes can be changed by the scale transformation
Transform {
        scale 1.2 1.2 1.2 children [
                Inline {url "CoordinateAxes.wrl"}  ]}


FShape {
geometry FGeometry {

# Function script defining the surface.
# Change to some other formulae to see how the surface geometry changes
# based on the parameters domain and the sampling resolution defined below
        definition "x+y+z"

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

#The resolution must be the same in all three dimensions
resolution [100 100 100]

 }

appearance FAppearance {
material FMaterial {
# Variable linear change of green color
# from 0 to 1 is defined for the surface defined
# within the bounding box x, y, z = [-1 1]
diffuseColor "r=1; g=(v+1)/2; b=0;"
  }   }
}
```

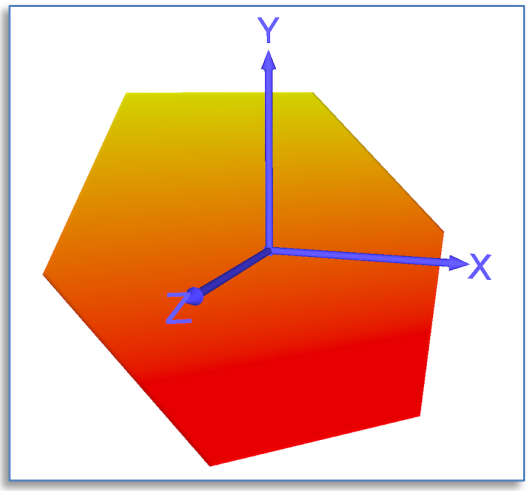


Figure 8. FVRML template of implicit surface. The code is in ImplicitSurface.wrl

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

# External VRML object "Coordinate Axes" is included in the scene.
# The size of the axes can be changed by the scale transformation
Transform {
            scale 1.2 1.2 1.2 children [
          Inline {url "CoordinateAxes.wrl"}  ]}

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){
                            superellipsoid=0.7^6-x^6-y^6-z^6;
                            cylinder=0.25^2-x^2-y^2;
                            final=min(superellipsoid, -cylinder);
                            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 linear change of color is defined for the CGS solid
diffuseColor "r=1; g=(v+1)/2; b=0;"
  }   }
}
```

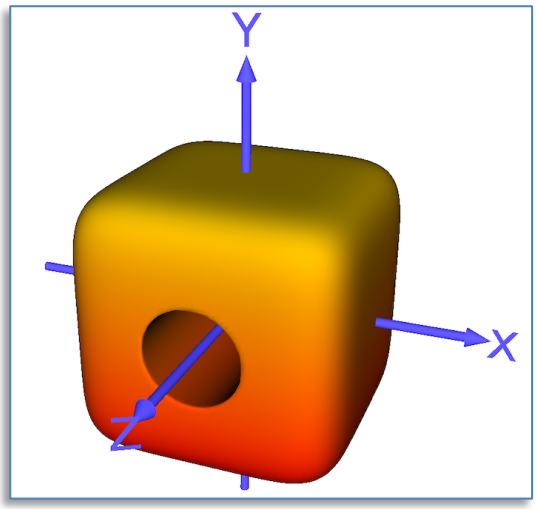
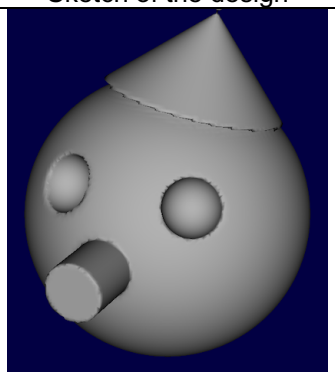
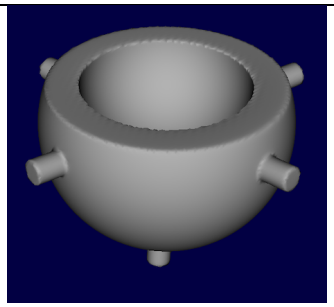
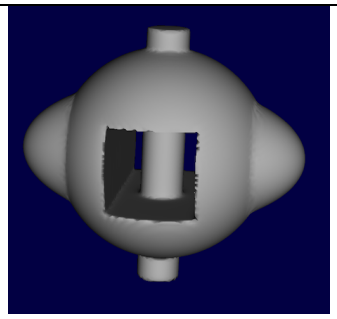
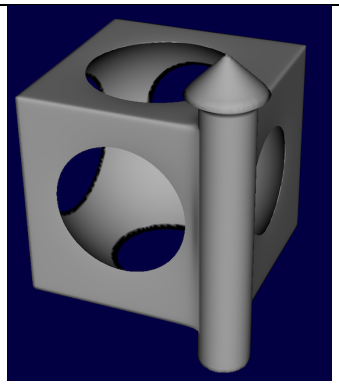
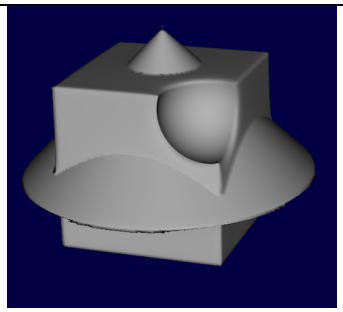
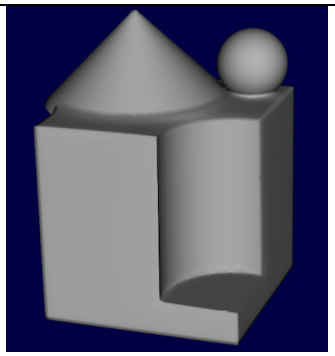
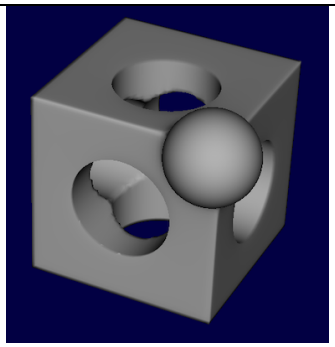
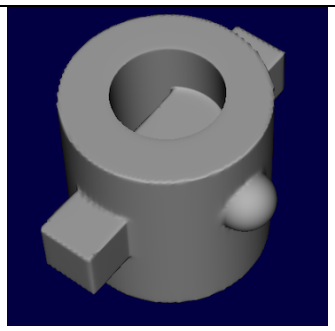
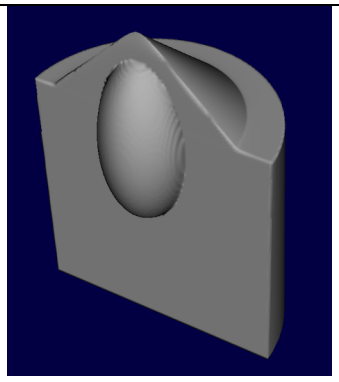
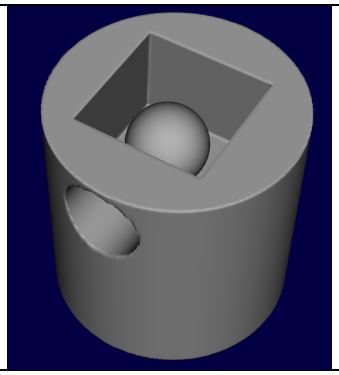


Figure 9. FVRML template of CSG solid. The code is in CSGsolid.wrl

Table 2. CSG designs to follow.

M	Boolean function description	Sketch of the design
1	$(((Sphere_1 \cup Cylinder) \setminus Sphere_2) \setminus Sphere_2) \cup Cone$	
2	$(((Sphere_1 \cup Cylinder_1 \cup Cylinder_2 \cup Cylinder_3) \setminus PlaneHalfSpace) \setminus Sphere_2$	
3	$((Sphere \cup Ellipsoid) \setminus Box)) \cup Cylinder$	
4	$(Box \setminus Sphere) \cup Cylinder \cup Cone$	
5	$(Box \cup Cone) \setminus Sphere$	

6	$(Box \setminus Cylinder) \cup Cone \cup Sphere$	
7	$(Box \setminus Cylinder_1 \setminus Cylinder_2 \setminus Cylinder_3) \cup Sphere$	
8	$(Cylinder_1 \setminus Cylinder_2) \cup Sphere \cup Box$	
9	$((Cylinder \cup Cone) \setminus PlaneHalfSpace) \cup Ellipsoid$	
10	$((Cylinder_1 \setminus Box_1) \setminus Cylinder_2) \cup Sphere$	

4.5 Experiment 5: Transformations and Motions

This assignment illustrates Module 4 and 5, and it serves a purpose to teach you how to create moving objects, and a morphing between two surfaces defined parametrically. To work on this assignment, you have to watch the following TEL lectures:

Week 8 - 2D Transformations
Week 9 - 3D Transformations
Week 10 - Motions and Morphing

Assignment instructions:

Create folder **Lab5**. Download into it from the course-site the files **Animation.wrl** (Fig. 10), **Morphing.wrl** (Fig. 11) and **CoordinateAxes.wrl** and refer to them while working on the following exercises. For each of the shape, you have to select a minimally sufficient sampling resolution providing for smooth surface visualization so that any further reduction of it will reveal the polygonal interpolation of the surface.

1. a. Derive a transformation matrix performing rotation by $\frac{\pi}{2}$ about an axis parallel to axis *Y* and passing through the point with coordinates (***M***+5, 0, 0).

   b. Apply this matrix to the parametric definitions of the curve obtained in experiment 1 (exercise 3). Derive the transformed definitions $x(u), y(u), z(u),\ u \in [0,1]$ of the rotated curve and display it.

   (4 marks)

2. Modify the parametric definitions to $x(u,t), y(u,t), z(u,t), u,t \in [0,1]$ so that the rotation of the curve will be displayed as a 5 seconds rotation motion with some deceleration.

   (4 marks)

3. a. With reference to Table 3, convert to $(u,v), y(u,v), z(u,v)$, $u,v \in [0,1]$ definitions of surfaces ***M*** and (***N+M***) and display them.

   b. Define parametrically using $(u,v,t),\ y(u,v,t),\ z(u,v,t),\ u,v,t \in [0,1]$ a swing (back and forth) morphing transformation between surfaces ***M*** and (***N+M***). The morphing animation has to take 5 seconds and it has be done with a uniform speed.

   (4 marks)

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

# External VRML object "Coordinate Axes" is included in the scene.
# The size of the axes can be changed by the scale transformation
Transform {
            scale 1.2 1.2 1.2 children [
                Inline {url "CoordinateAxes.wrl"}  ]}

        Background {skyColor 0.9 0.9 0.9 }

        FShape {
# Enabling animation
        loop TRUE
# Setting the time interval of 3 sec
        cycleInterval 3
        appearance FAppearance {
        material FMaterial {
        diffuseColor "r=1; g=0; b=0;"
          }
        }
        geometry FGeometry {
        resolution [6 2]
        parameters [0 1 0.0001 1]
# Using time variable t to animate definitions
        definition "
                x=0.1*v*cos(2*pi*u)+1*cos(2*pi*t);
                y=0.1*v*sin(2*pi*u)+1*sin(2*pi*t);
                z=0;"
 }   }
```

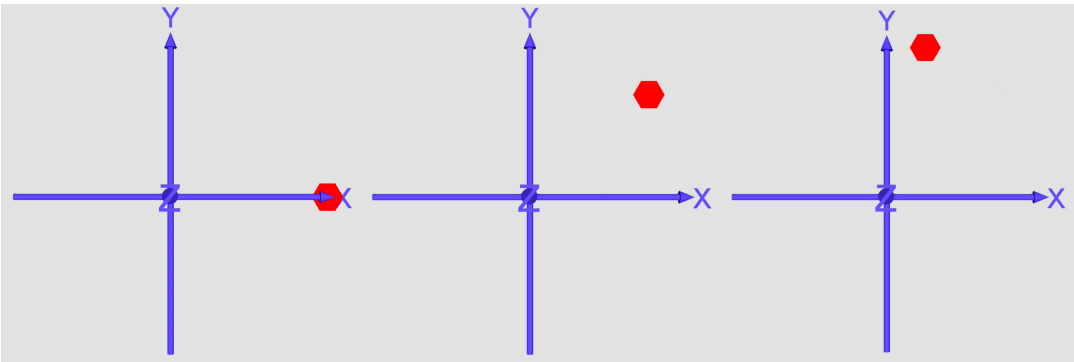
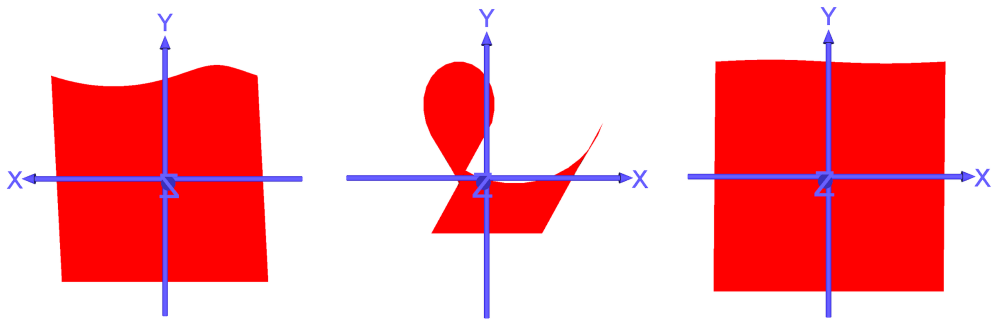


Figure 10. FVRML example of animation. The code is in Animation.wrl

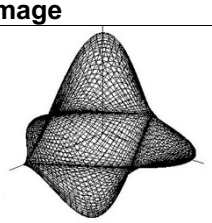
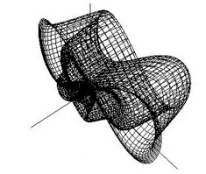
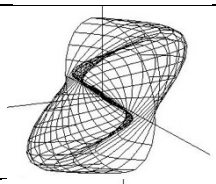
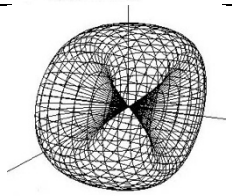
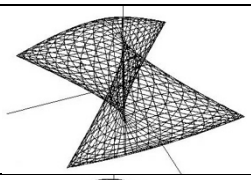
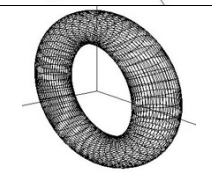
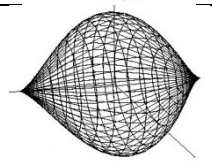
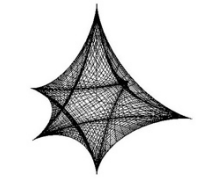
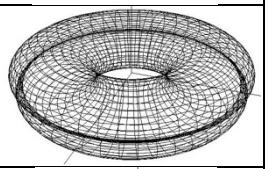
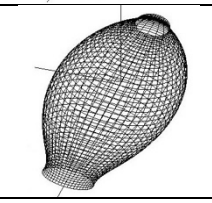
```
#VRML V2.0 utf8
EXTERNPROTO FGeometry [
        exposedField SFString definition
        exposedField MFFloat parameters
        ……..
[ the rest of the EXTERNPROTO is skipped]

# External VRML object "Coordinate Axes" is included in the scene.
# The size of the axes can be changed by the scale transformation
Transform {
            scale 1.2 1.2 1.2 children [
                Inline {url "CoordinateAxes.wrl"}  ]}
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 [0 1 0 1]
# Definition of the animated linear transformation (morphing)
# of a square polygon defined by x1(u,v), y1(u,v), z1(u,v)
# to a circular disk defined by x2(u,v), y2(u,v), z2(u,v)
definition "
function parametric_x(u,v,w,t)
# linear morphing function for x-coordinate
        { x1=-1+2*u;   x2=v*cos(u*2*pi);    return x1+(x2-x1)*t; }

function parametric_y(u,v,w,t)
# linear morphing function for y-coordinate
        { y1=-1+2*v;   y2=v*sin(u*2*pi);    return y1+(y2-y1)*t; }

function parametric_z(u,v,w,t)
# linear morphing function for z-coordinate
{return 0;}"
 }

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

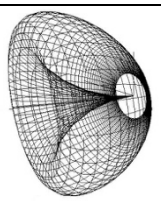
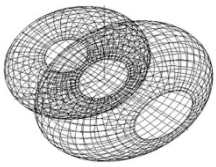
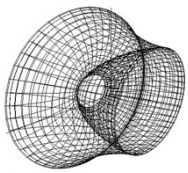
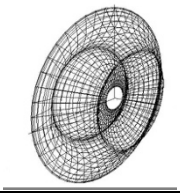
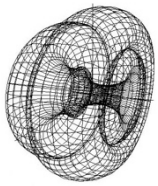
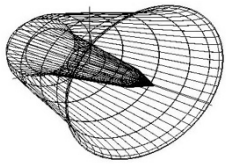
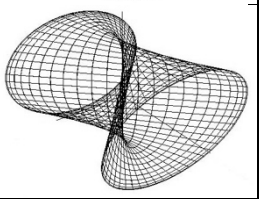
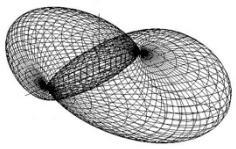
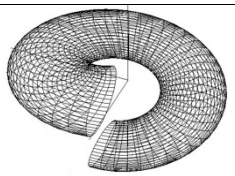
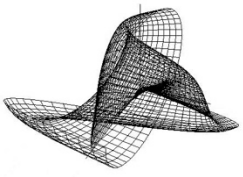



```
}
```

Figure 11. FVRML example of parametric morphing. The code is in Morphing.wrl

Table 3 Parametric surfaces

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

11	$x = 2.5\varphi/\left(1+\varphi^3\right)$ $y = 2.5\cos(\theta)\varphi^2/\left(1+\varphi^3\right)$ $z = 2.5\sin(\theta)\varphi^2/\left(1+\varphi^3\right)$	$0\le\theta\le 2\pi,\ \ 0\le\varphi\le 2\pi$	
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	$x = 2\theta\sin(\varphi)$ $y = 0.2\varphi\sin(10\theta)$ $z = 0.2\varphi\cos(10\theta)$	$0\le\theta\le \pi/5,\ \ 0\le\varphi\le 2\pi$	

5. REFERENCES

1 BS contact VRML/X3D viewer *http://www.bitmanagement.com* ***Use the link in the course-site***

2. Function-based Extension of VRML, *http://www.ntu.edu.sg/home/assourin/FVRML.htm* ***Use the link in the course-site***

3. VRML and X3D specifications *http://www.web3d.org/standards/all/*