CZ3003

COMPUTER GRAPHICS and VISUALIZATION

LAB 3 REPORT

Parametric Surfaces and Solids

LAB GROUP : **SSP6**

Sam Jian Shen (U1821296L)

School of Computer Science and Engineering

Nanyang Technological University

Contents

Surface 3
- 3D Plane 3
- 3D Triangle 4
- Bilinear surface 5
- Sphere 7
- Ellipsoid 8
- Cone 9

Solid 10
- Box 10
- Sphere 11
- Cylinder 12
- Cone 13

Rotational Sweeping 14
- To Surface 14
- To Solid 15

Translational Sweeping 16
- To Surface 16
- To Solid 17

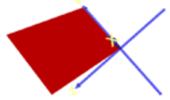
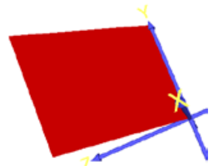
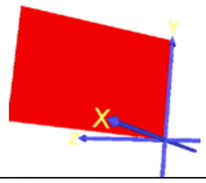
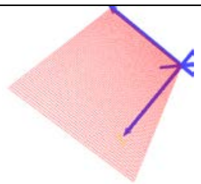
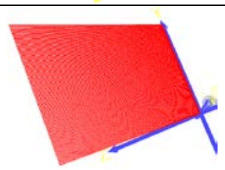
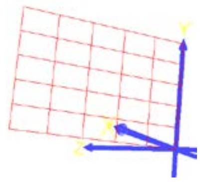
Sine Equation 18

Surface

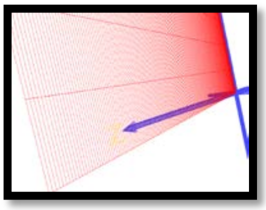
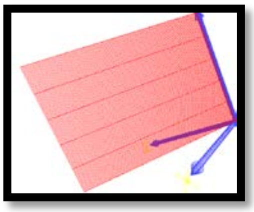
- In general solid utilize 2 parameters to craft out a surface

3D Plane

- Theory
  - To find a point(P) of the plane can be formed with 1 point(P_o) and 2 vectors ($V_{1\,\&\,2}$) of a given magnitude(u)
    - $P = P_o + vV_1 + uV_2$ where $V_1 = P_1 – P_0$, $V_2 = P_2 – P_1$
    - Since a point consist of 3 coordinates (x,y,z)
  - The equations can be derived as:
    - $Ax + By + Cz + D = 0$
- In order to model a 3D plane we need to provide given values of points
- Let $P_o = (0,0,0)$
- $P_1 = (0,1,0)$ $P_2 = (1,1,1)$
- $V_1 = (0,1,0)$ $V_2 = (1,0,1)$
- Sub into this eqn $P = P_o + vV_1 + uV_2$
  - $X = u$
  - $Y = v$
  - $Z = u$

View Mode	Normal (75,75)	High Resolution (200,200)	Low Resolution (5,5)	Remarks
Smooth				It has no difference in appearance.
Wireframe	 Source: 3D Plane.wrl	 Source: 3D Plane_HighRes.wrl	 Source: 3D Plane_LowRes. wrl	The number of rendering vertices point is more significant as the resolution goes higher. The number of “rectangle” it can produce depends on resolution values. For example, low resolution is 5x5 = 25.

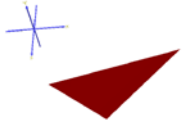
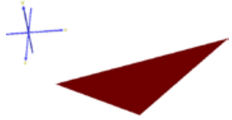
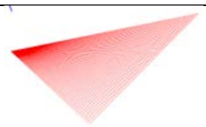
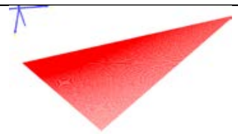
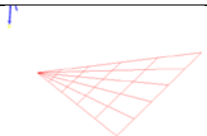
*Note that the resolution is referred to as the parameter u and v. It is to define how many separators between the domain range. For example, I can customize the resolution such that it is 5 x 200 for my plane shown below. The purpose of resolution may not be obvious on the plane but it is to ensure smooth/rough surfaces, we will be able to evidently prove this statement at the later experiment.

Source: 3D Plane_SpecialRes.wrl

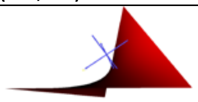
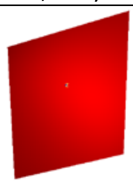
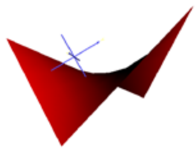
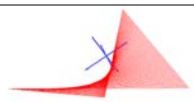
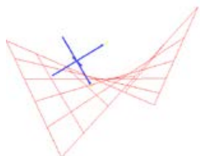
3D Triangle

- Theory
    - For a 3D triangle, it requires a bilinear equation to deal with.
    - It is base on 4 points to derived such equation
        - For triangle 2 of the points can let it be the same
        - It is important to choose the right point when sub into the equation (which we will discuss later at the experiment)
    - The equation as follows, where u and v is the parameter
        - First derived 2 different vectors in the same direction (the order matter)
            - $P' = P_1 + u(P_2 - P_1)$
            - $P'' = P_3 + u(P_4 - P_3)$
                - In this case let $P_4 = P_3$ // (Triangle rule)
        - The final equation would be:
            - $P = P' + v(P'' - P')$
- In order to model a 3D triangle we need to provide given values of points
- Let $P_1 = (3,3,3)$
- $P_2 = (2,3,4)$
- $P_3 = P_4 = (1,2,3)$
- Sub into $P = P' + v(P'' - P')$ eqn
    - x=3-u-2*v+u*v;
    - y=3-v;
    - z=3+u-u*v;

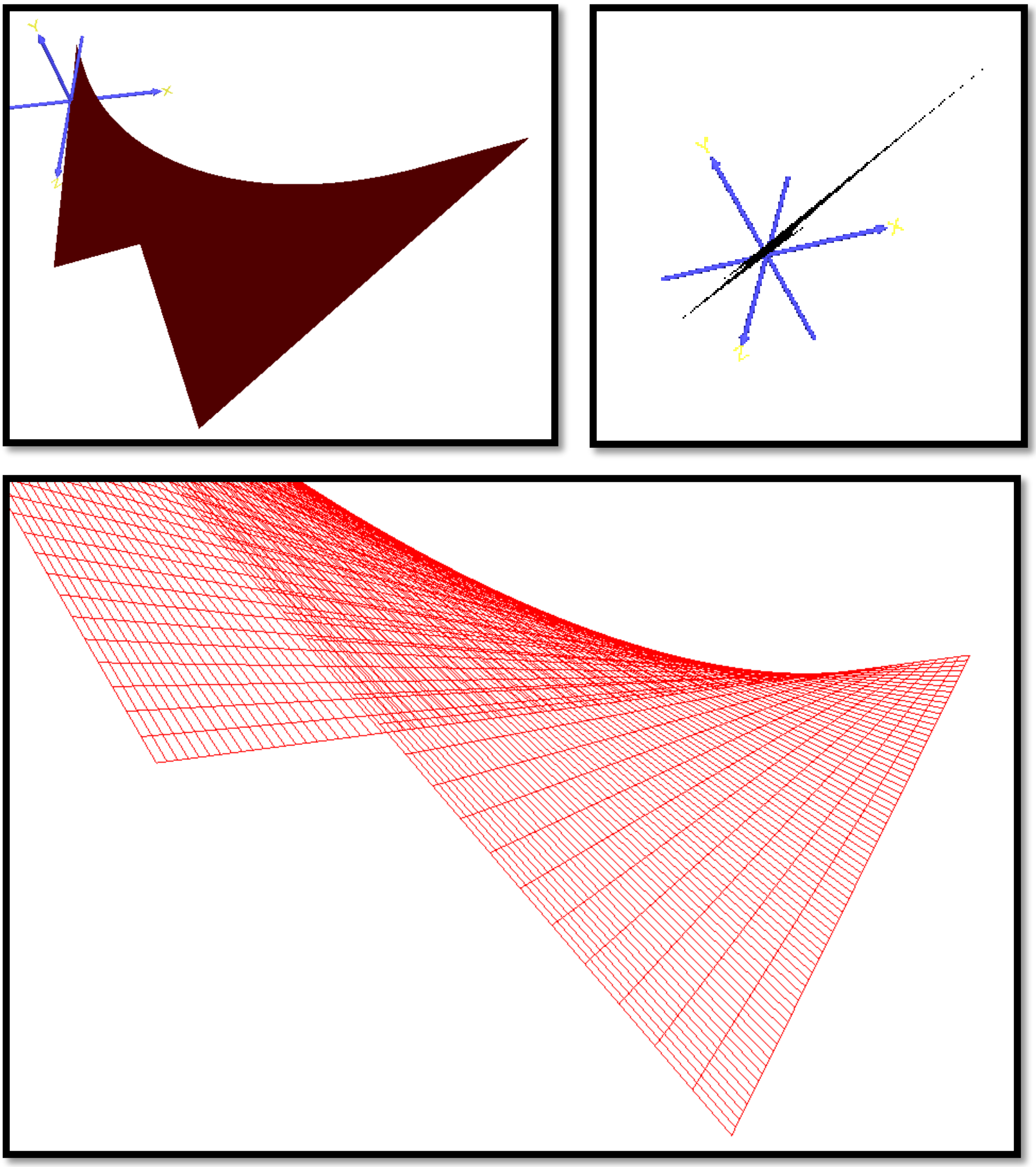
View Mode	Normal (75,75)	High Resolution (200,200)	Low Resolution (5,5)	Remarks
Smooth				It has no difference in appearance.
Wireframe	 Source: 3D Triangle.wrl	 Source: 3D Triangle _HighRes.wrl	 Source: 3D Triangle _LowRes.wrl	The number of rendering vertices point is more significant as the resolution goes higher. The number of “rectangle” it can produce depends on resolution values. For example, low resolution is 5x5 = 25.

Bilinear surface

- Theory
  - It is base on 4 points to derived such equation
  - The equation as follows, where u and v is the parameter
    - First derived 2 different vectors in the same direction (the order matter)
      - $P' = P_1 + u(P_2 - P_1)$
      - $P'' = P_3 + u(P_4 - P_3)$
    - The final equation would be (with final vector) :
      - $P = P' + v(P'' - P')$
- In order to model a 3D triangle we need to provide given values of points
- Let $P_1 = (-2,-2,2)$
- $P_2 = (-2,3,0)$
- $P_3 = (2,-2,-2)$
- $P_4 = (2,2,2)$
- Sub into $P = P' + v(P'' - P')$ eqn
  - x=-2+4*v;
  - y=-2+5*u-u*v;
  - z=2-2*u-4*v+8*u*v;

View Mode	Normal (75,75)	High Resolution (200,200)	Low Resolution (5,5)	Remarks
Smooth			  	It has differences in terms of the smoothness of the surface. This can be easily seen when the resolution is low we can see the turning point at the edge. It is more obvious when set graphic to "flat"
Wireframe	 Source: Bilinear Surface.wrl	 Source: Bilinear Surface_HighRes.wrl	 Source: Bilinear Surface_LowRes.wrl	The number of rendering vertices point is more significant as the resolution goes higher. The number of "rectangle" it can produce depends on resolution values. For example, low resolution is 5x5 = 25.

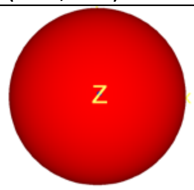
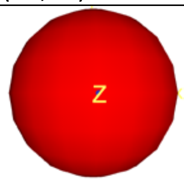
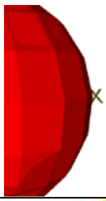
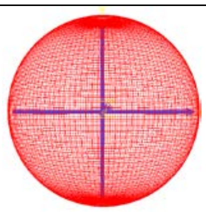
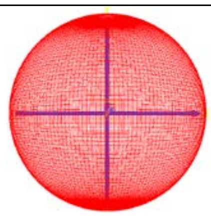
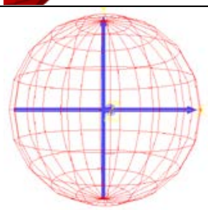
*Note that the choosing the 4 points must follow certain rules to ensure bilinear surface. Firstly, the initial 2 vectors chosen must be in follow the same directions. It should not cross one another. If cross one another the result would look like this as shown below.



Source: Bilinear Surface (Wrong usage).wrl

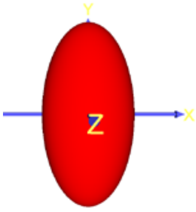
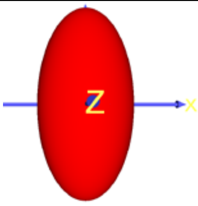
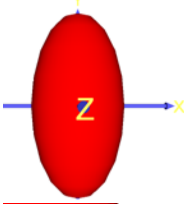
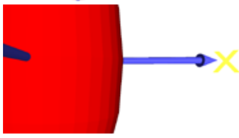
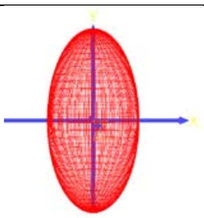
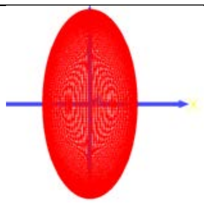
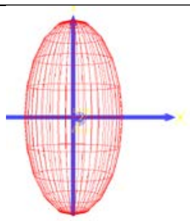
Sphere

- To achieve we use a circle equation and using the rotational sweeping method, it rotates 180 degrees to form the surface.
  - Rotational sweeping
    - Where f(n) is circle function , the rest is rotational in circle form
    - X = f(u)'*cos(pi*v);
    - Y = f(u)'';
    - Z = f(u)'* sin(pi*v); //pi 180degree rotate
- Parametric Equation
  - x=1*cos(2*pi*u)*cos(pi*v);
  - y=(1*sin(2*pi*u));
  - z=(1*cos(2*pi*u))*sin(pi*v);

View Mode	Normal (75,75)	High Resolution (200,200)	Low Resolution (20,20)	Remarks
Smooth			 	It has differences in terms of the smoothness of the surface. This can be easily seen when the resolution is low we can see the turning point at the edge. It is more obvious when set graphic to "flat"
Wireframe	 Source: Sphere.wrl	 Source: Sphere_HighRes.wrl	 Source: Sphere_LowRes.wrl	The number of rendering vertices point is more significant as the resolution goes higher. The number of "rectangle" it can produce depends on resolution values. For example, low resolution is 20x20 = 400.

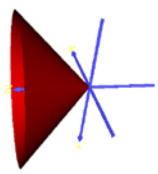
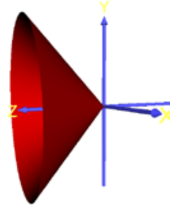
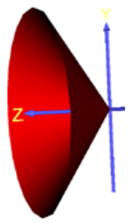
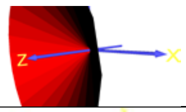
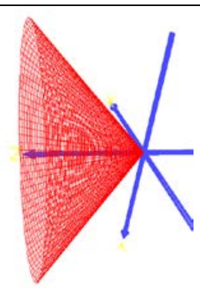
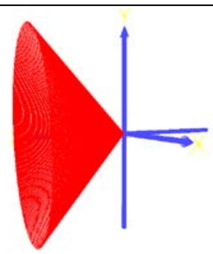
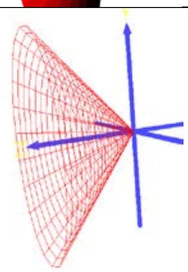
Ellipsoid

- To achieve we use an ellipsoid equation and using the rotational sweeping method just like a sphere, it rotates 180 degrees to form the surface.
    - Rotational sweeping
        - Where f(n) is circle function , the rest is rotational in circle form
        - Where a,b,c is a real number
        - X = a*f(u)’*cos(pi*v);
        - Y = b*f(u)’’;
        - Z = c*f(u)’* sin(pi*v); //pi 180degree rotate
- Parametric Equation (Let a = 0.5, b = 1, c = 0.3)
    - x=0.5*cos(2*pi*u)*cos(pi*v);
    - y=(1*sin(2*pi*u));
    - z=(0.3*cos(2*pi*u))*sin(pi*v);

View Mode	Normal (75,75)	High Resolution (200,200)	Low Resolution (20,20)	Remarks
Smooth			 	It has differences in terms of the smoothness of the surface. This can be easily seen when the resolution is low we can see the turning point at the edge. It is more obvious when set graphic to “flat”
Wireframe	 Source: Ellipsoid.wrl	 Source: Ellipsoid_HighRes.wrl	 Source: Ellipsoid_LowRes.wrl	The number of rendering vertices point is more significant as the resolution goes higher. The number of “rectangle” it can produce depends on resolution values. For example, low resolution is 20x20 = 400.

Cone

- To achieve we use a cone surface, there are 2 ways either rotational or translational method. In this experiencement we will be using translational method
    - Translational sweeping
        - Where f(n) is circle function , the rest is rotational in circle form
        - Where a,b,c is a real number
        - X = v*f(v)’ //v;changes to the circle radius it move from origin to 1
        - Y = v*f(v)’’;
        - Z = v //v;move the Z-axis from origin to 1
- Parametric Equation
    - x=v*cos(2*pi*u); //In this case is 2*pi because full circle rotation form parameter u
    - y=v*sin(2*pi*u);
    - z=v;"

View Mode	Normal (75,75)	High Resolution (200,200)	Low Resolution (20,20)	Remarks
Smooth			 	It has differences in terms of the smoothness of the surface. This can be easily seen when the resolution is low we can see the turning point at the edge. It is more obvious when set graphic to “flat”
Wireframe	 Source: Cone.wrl	 Source: Cone_HighRes.wrl	 Source: Cone_LowRes.wrl	The number of rendering vertices point is more significant as the resolution goes higher. The number of “rectangle” it can produce depends on resolution values. For example, low resolution is 20x20 = 400.

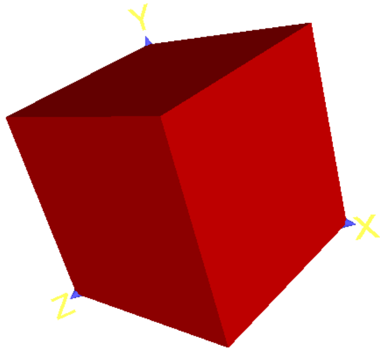
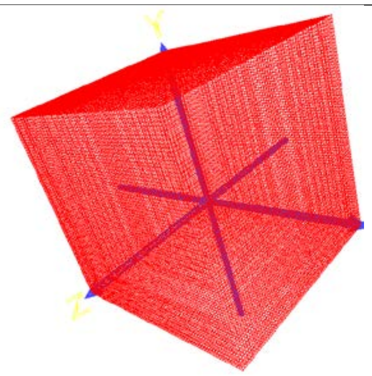
For surface, the resolution determines the smoothness of the surface. Having high-resolution result in more computational power to render all the vertices and vectors which results in lower performance. Low resolution will result in a rough surface.

Solid

- In general solid utilize 3 parameters to craft out a solid

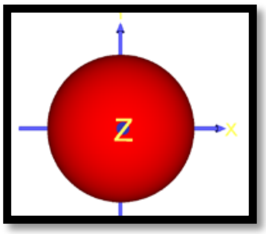
Box

- It is derived from 3 different vectors (straight-line equation) with
- All the point that matches accordingly u,v,w parameter will be shown (render only the surface area) as internal are not visible by the observer.
    - As long it satisfies "close" in a solid object.
- Parametric Equation
    - x=u;
    - y=v;
    - z=w;

View Mode	VRML (Normal)	Remarks
Smooth		A cube with length,breadth and height of value 1
Wireframe	 Source: Solid Box.wrl	The number of "rectangle" base on the resolution. In this case is 75x75x75 = 421875

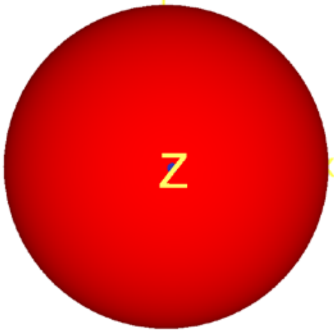
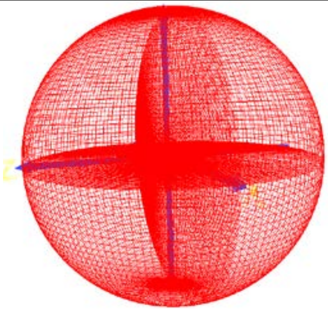
Sphere

- It is derived from 3 different vectors (straight-line equation) with
- All the point that matches accordingly u,v,w parameter will be shown (render only the surface area) as internal are not visible by the observer.
    - As long it satisfies “close” in a solid object.
- In surface the sphere can be considered a solid iff it is full sphere, not partial sphere.
- Changing the parameter w reduce the radius of the sphere
    - In this case reduce by 0.3



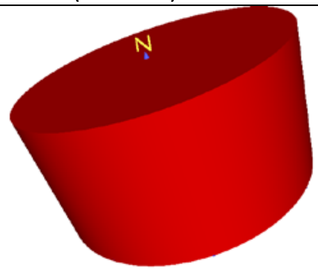
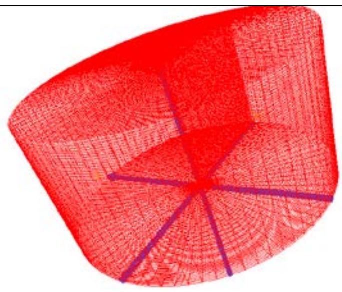
Souce: Sphere_Partial.wrl

- Parametric Equation
    - x=w*cos(2*pi*v)*1*cos(2*pi*u);
    - y=w*sin(2*pi*v);
    - z=w*cos(2*pi*v)*1*sin(2*pi*u);
    - Note* : the only different compare to surface is additional radius as 3rd parameter.

View Mode	VRML (Normal)	Remarks
Smooth		A solid sphere with radius of value 1.
Wireframe	 Source: Solid Sphere.wrl	The number of “rectangle” base on the resolution. In this case is 75x75x75 = 421875

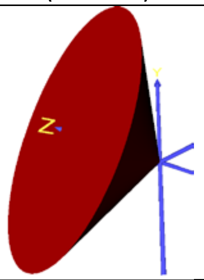
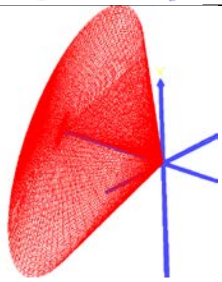
Cylinder

- Basically it is derived from the surface cylinder, the only differences are the radius of the circle varies with the 3rd parametric value. Such that the top and bottom of the cylinder can be formed into a solid cylinder.
- Parametric Equation
    - x=w*cos(2*pi*u);
    - y=w*sin(2*pi*u);
    - z=v;

View Mode	VRML (Normal)	Remarks
Smooth		A solid cylinder which refer to top and bottom cover from a surface cylinder
Wireframe	 Source: Solid Cylinder.wrl	The number of "rectangle" base on the resolution. In this case is 75x75x75 = 421875

Cone

- Basically it is derived from the surface cone, the only differences are the radius of the circle varies with the 3rd parametric value. Such that the bottom of the cone can be formed into a solid cone.
- The parameter v follows the rule on the surface cone as mentioned before.
- Parametric Equation
  - x=v*w*cos(2*pi*u);
  - y=v*w*sin(2*pi*u);
  - z=v;

View Mode	VRML (Normal)	Remarks
Smooth		A solid cone refers to the bottom of the cone from a surface cylinder.
Wireframe	 Source: Solid Cone.wrl	The number of “rectangle” base on the resolution. In this case is 75x75x75 = 421875

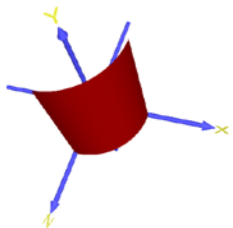
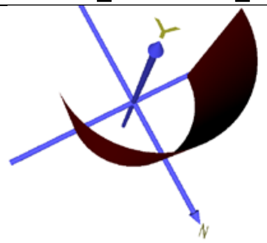
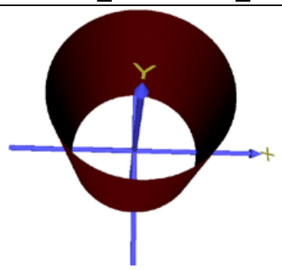
Rotational Sweeping

- To illustrate I am using the cylinder as an example.
- The rotational sweeping make used of rotation to form surface or solids.

To Surface

- Eqn
  - x=0.5*cos(2*pi*v);
  - y=u;
  - z=0.5*sin(2*pi*v);
- Steps
  - It starts off with rendering a straight-line base on 1st domain
  - The line will move in a circular motion with respect to x and y-axis base on 2nd domain
  - Once a circular forms the cylinder surface is formed

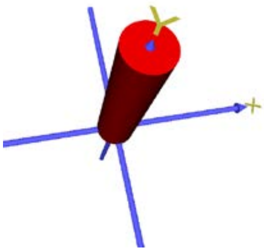
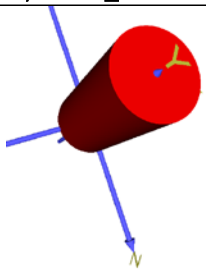
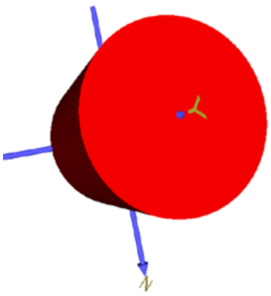
The check how it was rendered, changes towards the domain range for v

	VMRL	Domain	Remarks
Quarter	 Source: Cylinder Surface_Rotational_QuarterRender.wrl	U = [0,1] V = [0,0.25]	The partial render ¼ of cylinder surface
Half	 Source: Cylinder Surface_Rotational_HalfRender.wrl	U = [0,1] V = [0,0.5]	The partial render 1/2 of cylinder surface
Full	 Source: Cylinder Surface_Rotational.wrl	U = [0,1] V = [0,1]	A full cylinder surface

To Solid

- Eqn
  - x=0.5*cos(2*pi*v)*w;
  - y=u;
  - z=0.5*sin(2*pi*v)*w;
- Steps
  - It starts off with rendering a straight-line base on 1st domain
  - The line will move in a circular motion with respect to x and y-axis base on 2nd domain
  - Once a circular form the cylinder surface is formed
  - Afterward it renders the 3rd domain with respect to the circle equation's radius (which result in changing of the volume of the solid object)

The check how it was rendered, changes towards the domain range for w

	VMRL	Domain	Remarks
Quarter	 Source: Solid Cylinder_Rotational_QuarterRender.wrl	U = [0,1] V = [0,1] W= [0,0.25]	The partial render ¼ of cylinder solid volumn
Half	 Source: Solid Cylinder_Rotational_HalfRender.wrl	U = [0,1] V = [0,1] W= [0,0.5]	The partial render 1/2 of cylinder solid volumn
Full	 Source: Solid Cylinder_Rotational.wrl	U = [0,1] V = [0,1] W= [0,1]	A full cylinder solid

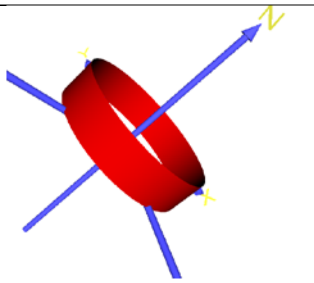
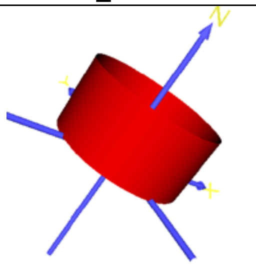
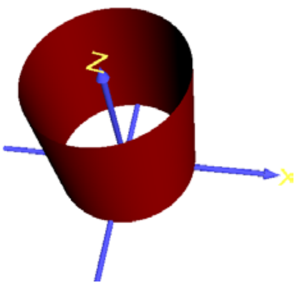
Translational Sweeping

- To illustrate I am using a cylinder as an example.
- Translational it used 2 points unique movement (be it straight-line or curvy) it will form a solid/surface

To Surface

- Eqn
    - x=0.5*cos(2*pi*u);
    - y=0.5*sin(2*pi*u);
    - z=v;
- Steps
    - It starts off with rendering a circle line base on 1st domain with respect o x and y-axis.
    - The circle will move in a straight line in this case with 2nd domain in term of the z coordinate
    - Once a distance reach end of 2nd domain, the cylinder surface is formed

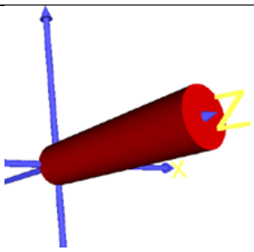
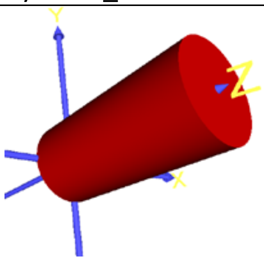
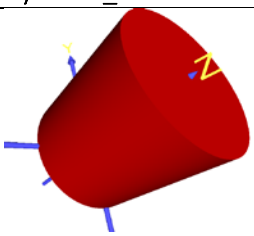
The check how it was rendered, changes towards the domain range for w

	VMRL	Domain	Remarks
Quarter	 Source: Cylinder Surface_Translational_QuarterRender.wrl	U = [0,1] V = [0,1] W= [0,0.25]	The partial render ¼ of cylinder surface
Half	 Source: Cylinder Surface_Translational_HalfRender.wrl	U = [0,1] V = [0,1] W= [0,0.5]	The partial render 1/2 of cylinder surface
Full	 Source: Cylinder Surface_Translational.wrl	U = [0,1] V = [0,1] W= [0,1]	A full cylinder surface

To Solid

- Eqn
  - x=0.5*cos(2*pi*u)*w;
  - y=0.5*sin(2*pi*u)*w;
  - z=v;
- Steps
  - It starts off with rendering a circle line base on 1st domain with respect o x and y-axis.
  - The circle will move in a straight line in this case with 2nd domain in term of the z coordinate
  - Once a distance reach end of 2nd domain, the cylinder surface is formed
  - Afterward it renders the 3rd domain with respect to the circle equation's radius (which result in changing of the volume of the solid object)

The check how it was rendered, changes towards the domain range for w

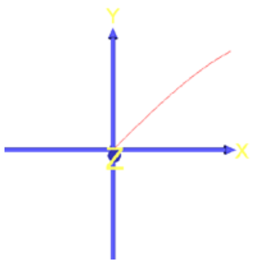
	VMRL	Domain	Remarks
Quarter	 Source: Solid Cylinder_Translational_QuarterRender.wrl	U = [0,1] V = [0,1] W= [0,0.25]	The partial render ¼ of cylinder solid volumn
Half	 Source: Solid Cylinder_Translational_HalfRender.wrl	U = [0,1] V = [0,1] W= [0,0.5]	The partial render 1/2 of cylinder solid volumn
Full	 Source: Solid Cylinder_Translational_Translational.wrl	U = [0,1] V = [0,1] W= [0,1]	A full cylinder solid

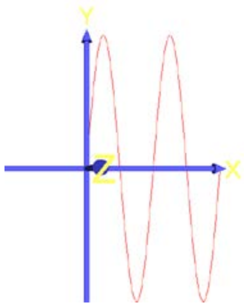
Base on observation rendering solid is the same between rotational and translational sweeping.

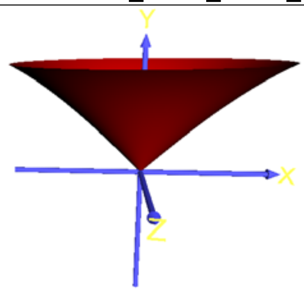
Sine Equation

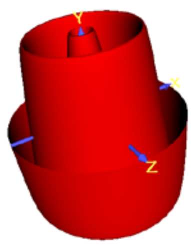
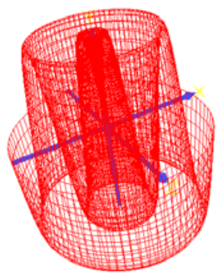
- For the sake of illustration, I also use 2 periods/cycle of the sine wave (4*pi*u)

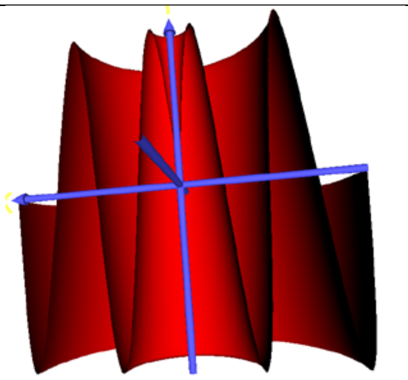
Sweeping Method	Steps	Para Eqn	VRML	Remarks

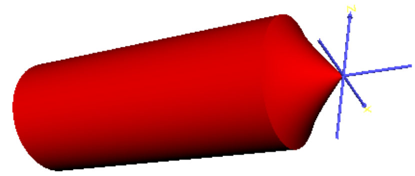
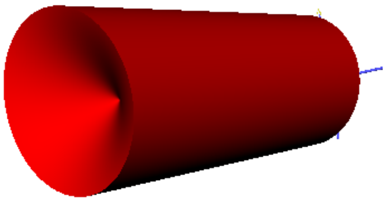
None	Curve	x=u;
		y=sin(u);
		z=0;
			
			Source: sine_wave_curve.wrl
				Less than a period sine wave

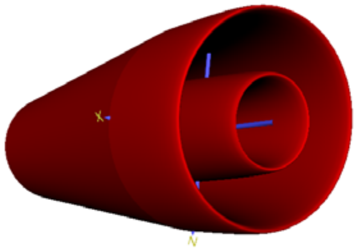
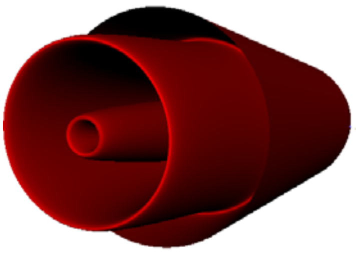
		x=u;
		y=sin(4*pi*u);
		z=0;
			
			Source: sine_wave_curve_extend.wrl
				2 period sine wave from the origin with respect to x and y axis.

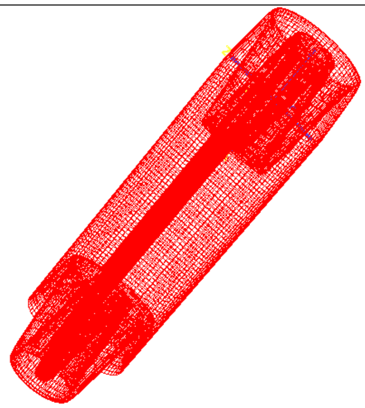
Rotational	Surface	x=u*(cos(2*pi*v));
		y=sin(u);
		z=u*(sin(2*pi*v));
			
			Source: sine_wave_curve_surface.wrl
				Less than a period sine wave surface with 360 degree rotational sweeping

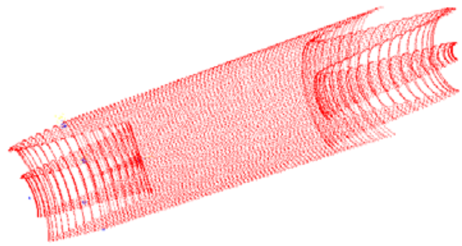
		x=u*(cos(2*pi*v));
		y=sin(4*pi*u);
		z=u*(sin(2*pi*v));"
			
			
			Source: sine_wave_surface_extend.wrl
				2-period sine wave surface from the origin with respect to x and y-axis with 360 degrees rotational sweeping

		x=u*(cos(pi*v));
		y=sin(4*pi*u);
		z=u*(sin(pi*v));"
			
			Source: sine_wave_surface_partial_render.wrl
				Cross-section

Translational	Solid	x=u*(cos(2*pi*v));
		y=sin(u)+5*w;
		z=u*(sin(2*pi*v));"
			
			
			Source: sine_wave_solid.wrl
				The solid is extend, 5 times value with translation sweeping method with y = f(u) + 5w

		x=u*(cos(2*pi*v));
		y=sin(4*pi*u)+5*w;
		z=u*(sin(2*pi*v));
			
			
				In this case the using 2 periods to turn into solid, notice the sine wave basically shift downward while forming the solid sine wave.

			 Source: sine_wave_solid_extend.wrl	

		x=u*(cos(pi*v));
		y=sin(4*pi*u)+5*w;
		z=u*(sin(pi*v));
			
			
			Source:
			sine_wave_curve_partial_render.wrl
				Cross section

*Note, all domain value remain unchanged in sine wave experiment.