

|  |
| --- |
| Lab 4 Report:  Implicit Solids |
| CZ2003 – Computer Graphics & Visualization  Wilson Thurman Teng U1820540H Lab Group: SSR2 |

Contents

[**Lab 4** 3](#_Toc24391218)

[**4.1** **Implicit Solid** 3](#_Toc24391219)

[**4.2** **Bounding Box/Resolution for Efficient Rendering under 5 seconds** 4](#_Toc24391220)

[**4.3** **Function-defined Diffuse Color** 5](#_Toc24391221)

[**Reparameterization** 5](#_Toc24391222)

[**Domain Pairing between the 2 Shapes** 5](#_Toc24391223)

[**Further Experimentation** 5](#_Toc24391224)

[**Screenshots** 6](#_Toc24391225)

# **Lab 4**

## **Implicit Solid**

## **Bounding Box/Resolution for Efficient Rendering under 5 seconds**

## **Function-defined Diffuse Color**

|  |
| --- |
| **Reparameterization** |
| To achieve morphing between shape 7 & 8, reparameterization is applied to the parametric equations of both shapes.  In this experiment, all the domains *(*θ,𝜑 *for Shape 7 & b*,θ *for Shape 8)* have been **reparametrized to a common base domain [0,1]**.  Next, by utilizing a **linear interpolation model with time parameter *t***, morphing animations can be produced. |
| **Domain Pairing between the 2 Shapes** |
| I have experimented with 2 versions of the morphing.  In **morphing7to8\_Version1.wrl**, [θ from S7 ⇔ *b* from S8] and [𝜑 from S7 ⇔ θ from S8].  In **morphing7to8\_Version2.wrl**, [θ from S7 ⇔ θ from S8] and [𝜑 from S7 ⇔ *b* from S8].  After experimenting with both versions, **morphing7to8\_Version1.wrl** may be a better morphing animation as compared to **morphing7to8\_Version2.wrl**.  This is so as **morphing7to8\_Version1.wrl** animation looks like Shape 7 is unfolding to form Shape 8 which is easy to understand and visualize. In contrast, **morphing7to8\_Version2.wrl** animation looks like it is folding into itself which is harder to visualize. |
| **Further Experimentation** |
| function back\_n\_forth(t)  { return 1-fabs(1-2\*t);                        }    To allow for back and forth animation between the 2 shapes, I have implemented the above function, **back\_n\_forth(*t*)**, which takes in parameter *t*. When *t* is cycled through domain [0,1]. It will ouput a value from 0 to 1 and since VRML cycles through the domain, we essentially have a periodic triangular function which allows for back and forth animation perpetually. |

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Screenshots** | | |
|  | **Shape 7** | **Morphing** | **Shape 8** |
| **Top-down View** |  | | |
| **Side View** |  | | |