Design and Development of a Visualizer for 3D Face with Emotion

Divya Nagar (ug201213005)
Sonika Agrawal (ug201210035)
Shreshtha Garg (ug201213035)

I. Abstract

Texture mapping is a graphic design process in which a two-dimensional surface, called a texture map, is wrapped around a three-dimensional object. Thus, the 3-D object acquires a surface texture similar to that of the 2-D surface. Texture mapping enhances the visual realism of 3D models by adding fine details. Recently, time and effort have been devoted to automatic texture mapping. It is possible to describe the texture mapping process in terms of a functional optimization problem. Several methods of this type have been proposed to minimize deformations. We have focused on human face texture mapping via multiple projection methods and came to conclusion for the method to be adopted for better results.

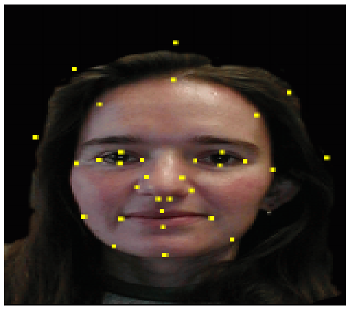
II. Introduction

There are different approaches available to map textures on 3D models and wireframes as well. Initially we worked with 3D models which were created in OpenGL itself and then tried to map a 2D image as texture on it, later we applied the same technique on wireframes. After successful mapping of texture on a 3D wireframe we started adding details to it for which we tried to refine the mesh by creating new faces and vertices. At the end of this beautification features can be added to give more realistic look to the 3D wireframe model.

III. Overview and Problem Statement

The problem is to simulate a face of a person using the frontal 2D image and 3D model of a face mesh (wireframe model). Since the depth information is not available we will actually approximate the 3D model of the 2D image, using the texture mapping in OpenGL. To do texture mapping we will find an appropriate function, for mapping the pixels of 2D image on the vertices of 3D model of the face mesh. Three points each on the wireframe and the image are used for mapping.

The program has to take input as: ".wrl" file for 3D wireframe of face and 2D images ".bmp" of faces (frontal poses) as shown in figure 1(b).


(a) 2D image of face

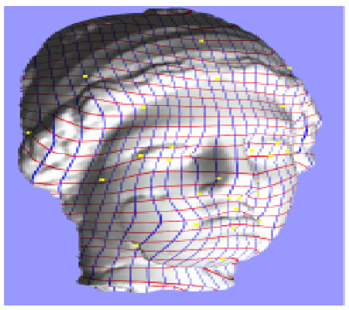

(b) 3D model of a fesh mesh

Fig. 1: Input of the Program

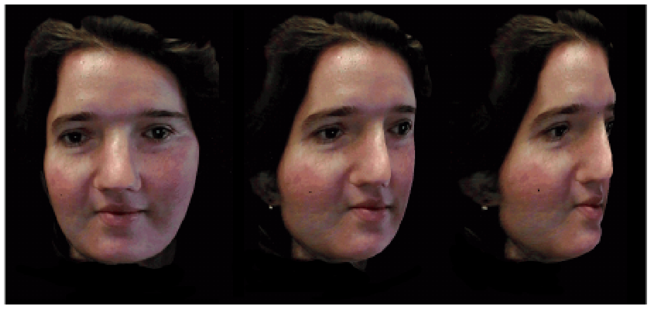


Fig. 2: A mapped model

IV. Motivation and Scope

Before texture mapping could be done in hardware, many graphics applications used only shading techniques to model solid objects. But due to the way human perception works, techniques such as flat shading needs huge numbers of polygons to achieve any decent level of realism, whilst methods such as Gouraud shading need excessive colour calculations to increase detail. It is far simpler to have a texture, that is mathematically mapped onto a surface. Compared to flat shading something like planetary terrain, using texture mapping can reduce the number of polygons used by a factor of as much as 200:1. The advantage of texture mapping is that it adds much detail to a scene while requiring a modest increase in rendering time.Texture mapping becomes a lot more useful and powerful when a texture can be mapped onto increasingly complex surfaces automatically, such as cubes, spheres, cylinders or arbitrary 3D surfaces. The results can be very realistic indeed. Texture mapping is the key to modern console systems. These methods are used extensively in most 3D games.

V. Contributions of this work

Our works will help people who wish to work with Vrml files in OpenGL. We developed a parser in C++ which can parse the Vrml file and can be used in any code, this kind of parser are not available and OpenGL also does not support Vrml files. We followed proper module architecture and wrote object oriented code which can be reused by other developers and easy to manage. This project can help many people and can make things easy for them to handle OpenGL, QT and Vrml together.

VI. Deliverables

A fully developed software that is capable of the following:

1. Reading a .wrl file as its first input.
2. Accepting a .bmp file as its second input.
3. Creating a 3D mesh that can be both moved and rotated in x, y and z directions.
4. Allowing the mapping of the image to the mesh.
5. Developing the final 3D model.
6. Providing rotation and movement features.

VII. Literature survey

We read the basics of computer graphics, illuminations methods, diffusion methods, specular methods, different coordinate systems for texture mapping, conversion of Cartesian coordinates into pixel coordinates and mapping of pixel positions to texture space. Apart from basic texture mapping we read papers related to interactive texture mapping, polygon meshes, refinement and optimization of meshes to add details.

A. The MatchMaker Algorithm[1]

The input to Matchmaker consists of a planar mesh M, a set of constrained mesh vertices VC, and a matching set PC of the user-defined 2D positions of those vertices. The main use of the planar mesh is to generate the optimal virtual boundary to minimize the anticipated parameterization distortion. The stages of the algorithm are listed below.

1. Virtual Boundary: First, the mesh is embedded in a bounding rectangle and the region between the mesh boundary and the rectangle is triangulated. This generates a new mesh M* which consists of the triangles of M and the new triangles. We fix the vertices of the virtual boundary in their current locations, adding them to VC and PC, respectively.

2. Matching: The matching procedure finds a triangulation T(PC) of the fixed points PC and a matching triangulation TM*(VC) of the mesh M*. The matching adds Steiner vertices to the mesh if it fails to compute the matching triangulations without them.

3. Embedding: Given the matching triangulations, each mesh triangle in TM*(VF) is mapped to the corresponding triangle in T(PF). After the mapping we obtain a provably valid embedding of the original mesh which satisfies the constraints.

4. Smoothing: The resulting embedding is smoothed, keeping the fixed vertices in place, in order to reflect the geometry of the 3D model as much as possible.

5. Post-Processing: To reduce the number of redundant Steiner vertices, those that can be removed without violating the validity of the mapping are removed from the mesh The result of the algorithm is a valid embedding that satisfies the constraints while closely preserving the metrics of the unconstrained parameterization.

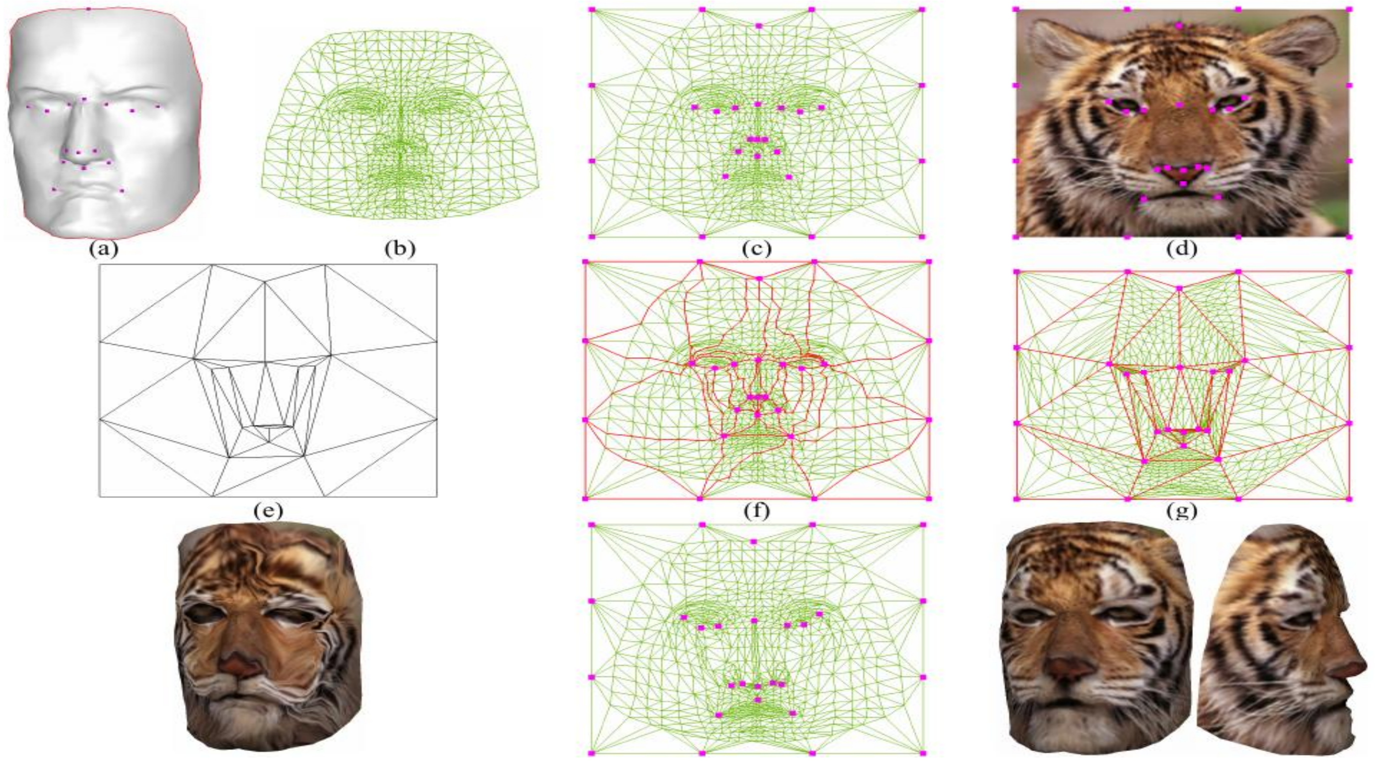


Fig. 3: MatchMaker Algorithm: Mapping a tiger face onto a human. (a) 3D model with VC vertices. (b) Unconstrained parameterization. (c) Parameterization with virtual boundary. (d) Texture with PC points. (e) Triangulation of PC points. (f) Matching triangulation TM*(VC) of M*. No Steiner vertices are required. (g) Triangle embedding. (h) Textured model after embedding. (i) Mesh after constrained smoothing.(j) Resulting textured model.

B. Parametrization

Parametrization implies mapping texture to surface and mesh parametrization is one of the most important part of texture mapping. There are various ways to do mesh parametrization.Here,we have used projection transformations to transform the 3D mesh to 2D planar mesh. There exits two projection methods:
1) parallel projection
2) perspective projection

1) Parallel Projection
In the parallel projection, vertexs coordinates are transferred to the 2D view plane along one parallel line.Two methods for parallel projection are orthographic projection i.e. projection along one line that are perpendicular to the view plane or projection plane, and oblique projection i.e projection along one line that has one obliqued angle to the projection plane.

For orthographic projection, we have the equation:

$$\begin{aligned} x_p &= x \\ y_p &= y \end{aligned} \tag{1}$$

In this equation,we project coordinate position(x; y; z) to the view plane$(x_p; y_p)$

2) Perspective Projection
In the perspective projection,vertexs positions are transformed to projection coordinates along lines that converge to the

center of projection behind the projection plane. Projection path of a spatial position (x; y; z) to a general projection point at $(x_{prp}; y_{prp}; z_{prp})$. The coordinate position $(x_p; y_p; z_{vp})$ is the intersection between the projection line and the view plane, where z_{vp} is some selected position for the view plane on the z_{view} axis.

For perspective projection, we have the equation:

$$\begin{aligned} x' &= x - (x - x_{prp})u \\ y' &= y - (y - y_{prp})u \\ z' &= z - (z - z_{prp})u \end{aligned} \tag{2}$$

where coordinate position $(x'; y'; z')$ represents a point along the projection line.

On the view plane, $z' = z_{vp}$ and we can solve the z' equation for parameter u at this position along the projection line:

$$u = \frac{z_{vp} - z}{z_{prp} - z} \tag{3}$$

Substituting this value of u into the equations for x' and y' , we obtain the general perspective-transformation equations:

$$\begin{aligned} x_p &= x(\frac{z_{prp} - z_{vp}}{z_{prp} - z}) + x_{prp}(\frac{z_{vp} - z}{z_{prp} - z}) \\ y_p &= y(\frac{z_{prp} - z_{vp}}{z_{prp} - z}) + y_{prp}(\frac{z_{vp} - z}{z_{prp} - z}) \end{aligned} \tag{4}$$

C. *UV mapping*

UV mapping is the 3D modeling process of projecting a 2D image to a 3D model's surface. This process projects a texture map onto a 3D object. The letters "U" and "V" denote the axes of the 2D texture because "X", "Y" and "Z" are already used to denote the axes of the 3D object in model space. UV texturing permits polygons that make up a 3D object to be painted with color from an image. The image is called a UV texture map, but it's just an ordinary image. The UV mapping process involves assigning pixels in the image to surface mappings on the polygon, usually done by "programmatically" copying a triangle shaped piece of the image map and pasting it onto a triangle on the object. UV is the alternative to XY; it only maps into a texture space rather than into the geometric space of the object. But the rendering computation uses the UV texture coordinates to determine how to paint the three-dimensional surface. UV coordinates are applied per face, not per vertex. This means a shared vertex can have different UV coordinates in each of its triangles, so adjacent triangles can be cut apart and positioned on different areas of the texture map. The UV Mapping process at its simplest requires three steps: unwrapping the mesh, creating the texture, and applying the texture.

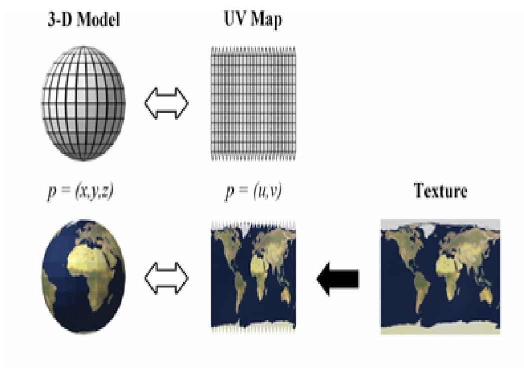


Fig. 4: Example of UV Mapping

VIII. Brief description of Work Done

Texture mapping is one of the most powerful tool of computer graphics and have vast application when it comes to animations and virtual reality. Texture mapping can be done in mainly two ways either texture scanning or image order scanning. A disadvantage of texture scanning is that a selected texture patch usually does not match up with the pixel boundaries so we prefer image order scanning to map the textures.

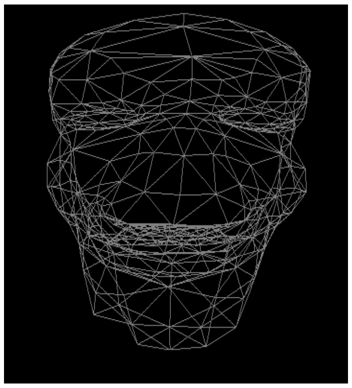
Step wise description of the work done is given as follows:

A. Extraction of Index and Vertex

Vrml file with wrl extension is given as input. A WRL file is a plain ASCII text files that includes data specifying 3-D details such as vertices, edges for a 3-D polygon, surface color, image-mapped textures, light and reflection mapping, and transparency. In our work, , as we load the wrl file in the software, the parser on parsing extracts the data of every vertex coordinates and faces indices which are thus obtained to reconstruct 3D face wireframe.The acquisition of the data is implemented using c++ language. We used Boost to develop the parser. Boost is a set of libraries for the C++ programming language that provide support for tasks and structures such as linear algebra, pseudorandom number generation, multithreading, image processing, regular expressions, and unit testing. It contains over eighty individual libraries. We only worked with libraries which include regular expressions.

B. Mesh Parametrization

Next step is to produce a valid parametrization of given mesh. We use perspective and parallel projection as parametrization technique. Below given figure shows results of perspective and parallel projection on a face mesh. As can be seen from the below images, we observe that parallel projection given better results for face's parametrization.


(a) using perspective projection

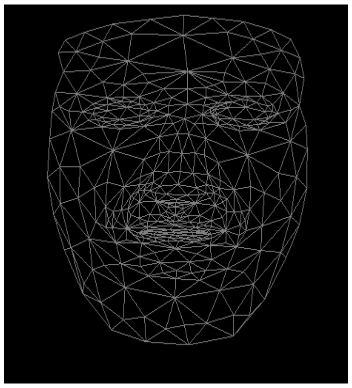

(b) using parallel projection

Fig. 5: Parameterization of a face mesh

C. Texture Mapping

After finishing the mesh processing, we need to produce the image texture which will be mapped onto the mesh. The last step is displaying the 3D face model with texture on screen. In OpenGL, glEnable(GL TEXTURE 2D) is mainly used to enable the texture. glBindTexture(GLenum target, GLuint texture) is used to bind a named texture to a texturing target. After that, we need to begin draw triangles. In this section, we need add each vertex with correspondent texture value. The texture function is glTexCoord2f(x,y). For calculating corresponding texture value for each vertex, we use UV mapping. UV mapping has been described in details in the previous section.

D. GUI Creation

GUI creation is being done using QT creator in which we used XML to define objects, CSS to give beautification to objects and C++ to provide functionality. There are three main windows with three different contexts. At one time only one context works in a window. Once window starts, OpenGL gets initialize and then for other features like rotate/move we call paint function which runs with a timer, times starts paint in every 6 milliseconds.

1) UML Diagrams: 1.1) Use Case Diagram

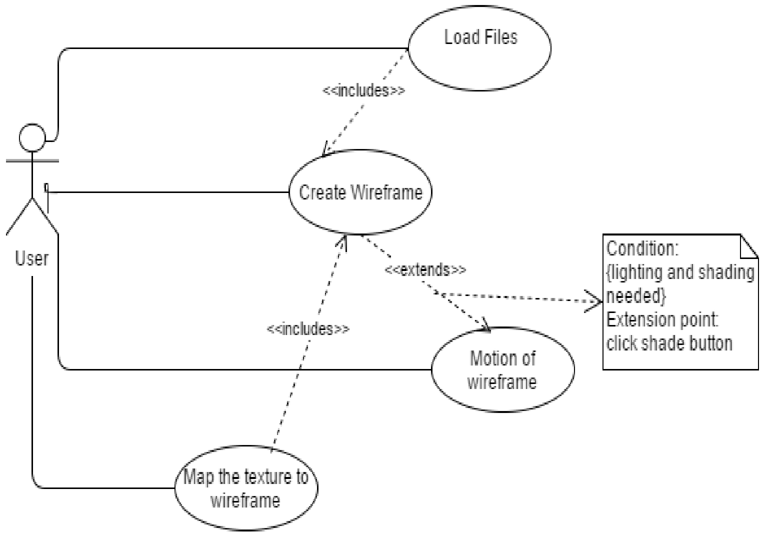


Fig. 6: Use Case Diagram

1.2) Activity Diagram

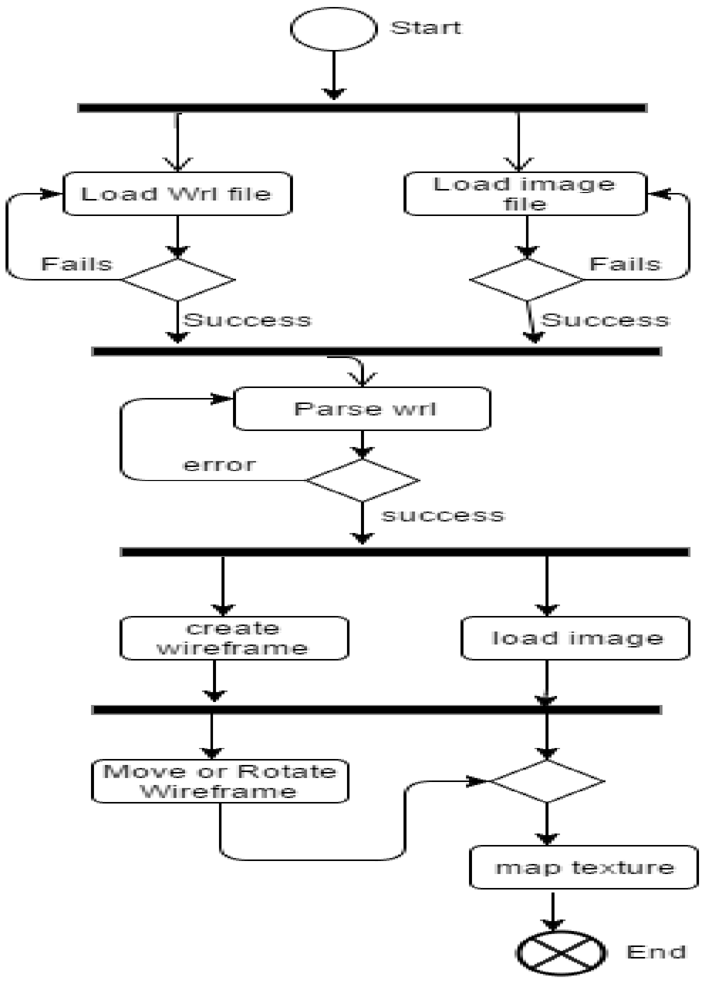


Fig. 7: Activity Diagram

2) Test Cases: Below tables describe possible testcases of the software.

2.1) Test 1: Wrl File Selection

Precondition	Software is installed properly with all dependencies
Trigger	User clicks on ”Open Wrl” button
Post condition (test fails)	System failure Window will not appear or will show all type of files
Post condition (test passes)	Window will come and user will be able to select Wrl files

TABLE I: Wrl File Selection

2.2) Test 2: Image File Selection

Precondition	Software is installed properly with all dependencies
Trigger	Click on ”Open Image” button
Post condition (test fails)	System failure Window will not appear or will show all type of files
Post condition (test passes)	Window will come and user will only be able to select bmp files

TABLE II: Image File Selection

2.3) Test 3: Empty wrl path

Precondition	user clicks on ”Open Wrl” button and selection window appear
Trigger	Select no file
Post condition (test fails)	Empty Wrl file error will be shown
Post condition (test passes)	No error will be shown and if image path given ”OK” button will be enabled

TABLE III: Empty Wrl File

2.4) Test 4: Empty image path

Precondition	user clicks on ”Open image” button and selection window appear
Trigger	Select no file
Post condition (test fails)	Empty image file error will be shown
Post condition (test passes)	No error will be shown and if wrl path given ”OK” button will be enabled

TABLE IV: Empty image File

2.5) Test 5: Data Parsing

Precondition	Paths of Wrl file is given
Trigger	Click on OK button on the first window
Post condition (test fails)	Parsing error will be shown
Post condition (test passes)	Continue the creation of OenGL object

TABLE V: Data Parsing

2.6) Test 6: Texture Loading

Precondition	Paths of Image is given
Trigger	Click on OK button on the first window
Post condition (test fails)	Texture loading error will be shown
Post condition (test passes)	Continue to load texture show the image

TABLE VI: Texture Loading

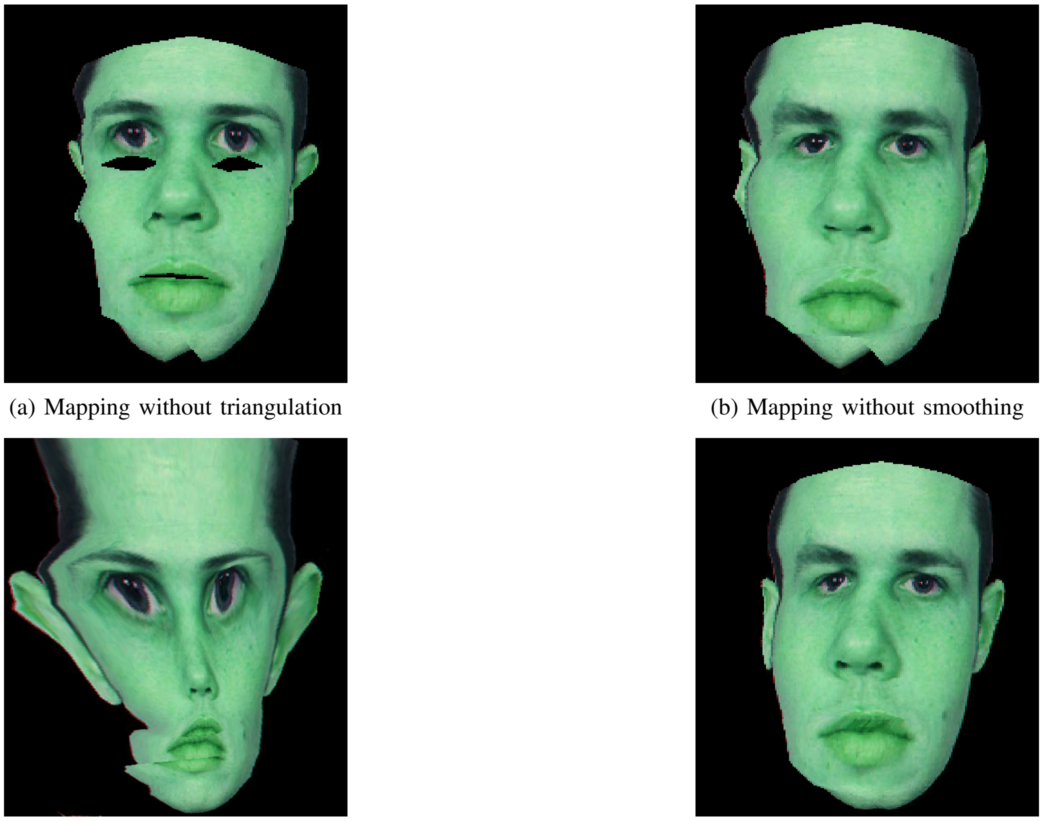
2.7) Test 7: Mapping

Precondition	Image and Wrl loaded properly
Trigger	Click on ”Map” Button
Post condition (test fails)	System failure window will not open
Post condition (test passes)	Mapping starts new window appear

TABLE VII: Mapping

IX. Experimental results/ Proofs

1. We performed both perspective and parallel transform on 3D mesh and did the mapping. Parallel transform is giving better results to us as it can be seen in figure-5.
2. Face texture mapping is much more difficult as compare to mapping on symmetrical meshes like cube, sphere as we need to deal with asymmetry.
3. If there is any place in the mesh where triangles are not present then it changes mapping drastically due not not getting any corresponding uv maps for mapping.
4. Smoothing of mesh gives better results as compare to rough mesh.


(a) Mapping without triangulation

(b) Mapping without smoothing

(c) Mapping with perspective projection

(d) Final mapping

X. Discussion

We have applied two methods of projection on same image and wrl file. As mentioned in the result section, parallel projection gives better result as compared to perspective projection for our project.
In our project, instead of following one to one feature mapping, we have tried mapping the whole texture at once. This process has both advantage and disadvantage. The advantage being, that there are no discontinuities in the final 3d face model but the disadvantage being that the dissimilarity between image and wrl brings distortion. Since there is not feature wise mapping, the actual mapping might not fits with the required mapping. Hence for better results we had to take approximate square dimensional images.

XI. Demonstration

This software is designed and developed for both Windows and Linux. We developed Graphical User Interface using QT Creator-3.6.1 and QT-5.0. Algorithm implementation is being done in C++ OpenGL and we also developed a small parser using Boost library which parses WRL file to get the data of vertices and faces indices.

A. Selecting Vrml file and Image

In the starting of the software a window will appear in which user will be able to give the path of Vrml file and Image file. The extension of Vrml file and image file will be .wrl and .bmp.

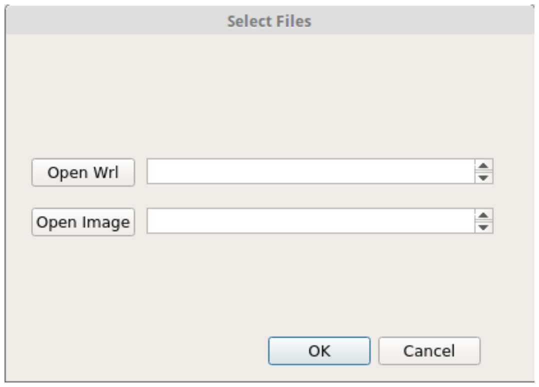


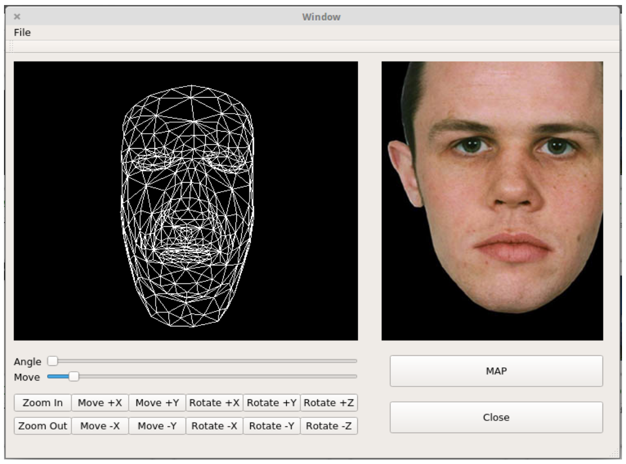
Fig. 9: First window

Button	Functionality
OK	Starts loading the Vrml file, image and redirects to second window
Cancel	Close the software

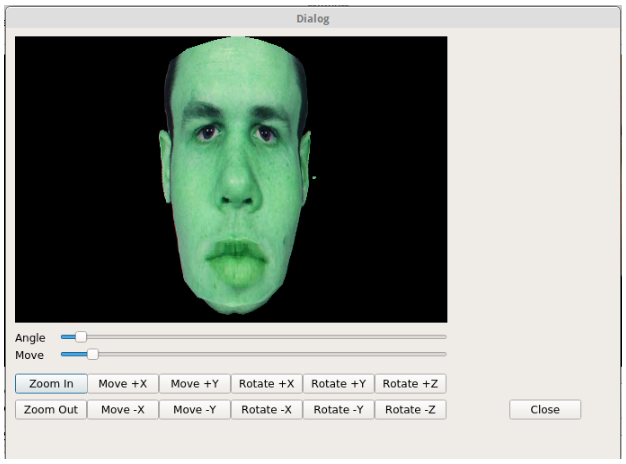
TABLE VIII: Buttons with their functionalities for first window

B. *Loading Vrml file and Image*

After getting Vrml file and image path from the user, software runs the parser on Vrml file, parses the data of vertices, faces indices and gives them to OpenGL functions to create OpenGL object. Second window shows the created OpenGL object and loaded image in separate widgets. In this window 14 buttons are given. Functionality of each button is given in the table below.



(a) Second window



(b) Third Window

Fig. 10: Software screenshots

Button	Functionality	Button	Functionality
Zoom In	Move mesh to negative Z direction	Rotate +Z	Rotate mesh to positive Z direction
Zoom Out	Move mesh to positive Z direction	Rotate +Z	Rotate mesh to negative Z direction
Move +X	Move mesh to positive X direction	Rotate +X	Rotate mesh to positive X direction
Move -X	Move mesh to negative X direction	Rotate -X	Rotate mesh to negative X direction
Move +Y	Move mesh to positive Y direction	Rotate +Y	Rotate mesh to positive Y direction
Move -Y	Move mesh to negative Y direction	Rotate -Y	Rotate mesh to negative Y direction
MAP	Starts mapping and shows a new window with mapped object	Close	Close the current window and opens the first window

TABLE IX: Buttons with their functionalities for second window

C. Mapping image to mesh

In the second window after loading the Vrml file and image when user clicks on ”MAP” button, sof tware starts mapping mapping the image on mesh and a new window appears which displays the mapped mesh. Functionalities of the button which are shown below to the widget are same as previous window which are described in TABLE-II.

XII. Individual Contribution

Name	Work
Divya Nagar	Parser development and UI creation helped by Shreshtha Garg
Shreshtha Garg	Literature survey, Integration of parser, UI and OpenGL algorithms, Testing
Sonika Agrawal	Implementation of OpenGL transforms and mapping algorithms helped by Divya Nagar

XIII. Future scope of work

The method proposed in this report is a effective system for adding textures onto 3D face model. But there is a lot of scope for future work. Some of the sections where major work can be done are as following :

A. Parametrization

Parametrization is one of the key concept for mapping a texture on 3D mesh. Better parametrization can change the results drastically. We have tried out parallel and perspective projections as methods for parametrization of the given mesh. Though we have achieved good results using these techniques, these are general techniques and do not consider the fact that the input mesh will only consist of faces. New techniques, specified for parametrizing face meshes better, can be found out and applied.

B. Feature Detection

Feature detection is new and interesting field of computer vision. Here, we can apply detection tool for finding face landmarks such as eyes, nose and mouth. This will help in finding one to one correspondence between the textute and mesh’s features and will provide less distortion in the final result.

C. Beautification

Mapping the texture is a important area of study because it let us make graphics more realistic. More features such as smoothing the skin, changing the skin shading, applying makeup etc can be applied that can create even more realistic and enhanced scenes.

D. 3D textures

Texture mapping is not only constrained to 2D textures. 2D textures provide less information then needed, hence while mapping 2d image on 3d mesh, one has to make assumptions. One of the next step can be applying 3D textures, as 3D textures contains more information and hence can provide better results.

XIV. References

1.Kraevoy, V., Sheffer, A., Gotsman, C.: Matchmaker: Constructing Constrained Texture Maps. In: Proceedings of ACM SIGGRAPH 2003, ACM Transactions on Graphics, vol. 22(3), pp. 326333 (2003)
2.LVY, B. Constrained Texture Mapping for Polygonal Meshes. In Proceedings of ACM SIGGRAPH 2001, Computer Graphics Proceedings, Annual Conference Proceedings, 417-424.
3. Edition ,Morgan Kaufman Publishers,2008. Donald Hearn, M. Pauline Baker. Computer Graphics C Version ,Second Edition.Prentice Hall,2008
4. Jerom Maillot , Hussein Yahia, Anne Verroust, Interactive texture mapping, Thomus Digital Image-INRIA-Rocquencourt.
5. SANDER, P. V., SNYDER, J., GORTLER, S., AND HOPPE, H. 2001. Texture Mapping Progressive Meshes. In Proceedings of ACM SIGGRAPH 2001, Computer Graphics Proceedings, Annual Conference Proceedings, 409-416.
6. Introduction to Modern OpenGL Programming, University of New Mexico and Dave Shreiner ARM Inc,University of Texas at Austin CS384G - Computer Graphics Fall 2010 Don Fussell
7. Alan Ezust, Paul Ezust, An Introduction to Design Patterns in C++ with Qt 4,
8. Changsheng Xiang ,A NEW WAY FOR MAPPING TEXTURE ONTO 3D FACE MODEL, UNIVERSITY OF DAYTON, Dayton, Ohio December, 2015
9. Paul S. Heckbert, Fundamentals of texture mapping and and image wraping, University of California Berkeley,CA-94720, June 17,1989
10. OpenGL texture mappnig tutorial, available at: http://www.opengl-tutorial.org/