SOFTWARE DESIGN DOCUMENT

For Cloudy Mesh

12/3/2012

Prepared by

Lucky People:

Umut Ağıl

Emre Avan

Gökhan Uras

Burak Üçok

Change History

VERSION NUMBER	***DATE***	***NUMBER OF FIGURE, TABLE OR PARAGRAPH***	***ADDED/ MODIFIED/ DELETED***	***TITLE OR BRIEF DESCRIPTION***
Version 1.0	11.11.2012			Original version
Version 1.1	17.01.2013	All the document	Modified	The document layout has been changed to make it more readable.
Version 1.1	17.01.2013	Section 3.2	Modified	Reference is given according to SRS
Version 1.1	17.01.2013	Section 3.1	Modified	Data flow diagram description has been added.
Version 1.1	17.01.2013	Section 3.2.1	Modified	Function description has been modified
Version 1.1	17.01.2013	Section 3.2.2	Modified	Function description has been modified
Version 1.1	17.01.2013	Section 3.2.3	Modified	Function description has been modified
Version 1.1	17.01.2013	Section 3.2.4	Modified	Function description has been modified
Version 1.1	17.01.2013	Section 3.1.4	Modified	Server module has been modified

Version 1.1	17.01.2013	Section 3.2.28	Added	Description of uploadMesh() function has been added
Version 1.1	17.01.2013	Section 3.2.29	Added	Description of convert() function has been added
Version 1.1	17.01.2013	Section 3.2.30	Added	Description of neighborSelection() function has been added
Version 1.1	17.01.2013	Section 3.2.31	Added	Description of findNeighbor() function has been added
Version 1.1	17.01.2013	Section 3.2.31	Added	Description of sendCloudFunction() function has been added
Version 1.1	17.01.2013	Section 5.28	Added	Pseudo-code of uploadMesh() function of browser module has been added.
Version 1.1	17.01.2013	Section 5.29	Added	Pseudo-code of uploadMesh() function of server module has been added.
Version 1.1	17.01.2013	Section 5.30	Added	Pseudo-code of convert() function has been added.
Version 1.1	17.01.2013	Section 5.31	Added	Pseudo-code of findNeighbor() function has been added.
Version 1.1	17.01.2013	Section 6	Added	Pseudo-code of findNeighbor() function has been added.

Table of Contents

1. Introduction

1.1. Purpose

This SDD document ensures detailed description of system architecture and software design of a Cloudy Mesh. Features, modules and functions of the software are explained in the document. Software developer can begin implementing through this document. This document also contains required UML diagrams and GUI information.

1.2. Scope

This document is prepared considering SRS of Cloudy Mesh which is published before. This document covers software design stage of Cloudy Mesh and contains crucial information relevant to software implementation. System structure of the software is also described in this document. Updated version of this document will be published if any update happens related to the software implementation.

1.3. Overview

This document is composed of eight sections. The first section is an introduction to document. The second section gives general overview of the system. Third part explains the architecture of the system. Fourth part includes the data design and description. Fifth part is about the details of the component design. Sixth part shows and explains the user interfaces. Eighth part contains cross-reference to SRS document. The last part is appendix.

1.4. Reference Material

- IEEE Std 1016-1998 - *IEEE Recommended Practice for Software Design*

1.5. Definitions and Acronyms

Mesh: Compilation of vertices, edges and faces of object.

Vertex: Corner of a polygon.

Edge*:* Line between two vertices.

Cloud Computing: Use of computing resources delivered as a service over a network.

Database*:* Structured collection of data.

Server: A physical computer dedicated to run one or more services (as a host), to serve the needs of the users of other computers on a network.

2. System Overview

This software system will be a mesh editing tool which will work on browsers using a cloud computing system. The GUI (graphic user interface) provides the user options that will allow her/him edit meshes. The program functionalities are planned to be:

- Grouping individual meshes
- Disintegrating groups of meshes
- Importing and exporting meshes
- Scaling objects
- Zooming in and out
- Rotating and translating objects
- Selecting geometric objects from list

The program is generally composed of two main parts. One is WebGL part where the user interacts with the program and the program visualizes the data. The other one is cloud part, where the user and mesh information are stored and the computations are done. However, the program has many modules which consists different parts of program that interact with each other. The modules are:

- Database – login, storing user and mesh information
- Browser – WebGL, GUI
- Python Server
- Cloud component

3. System Architecture

3.1. Architectural Design

Cloudy Mesh system consists of four modules; database, browser, cloud and server module. These modules interact with each other according to user requests. Browser module has GUI elements which are necessary to take the user commands. Commands that do not require mesh editing or database operations are done in browser module using JavaScript. Otherwise, if user commands a mesh editing operation then the function name and input parameters are sent to cloud module through server module. The application in cloud module calls the function with the given name. Functions in the cloud and server module are written in Python. After the function completes, it sends the resulting data back to browser module. Remaining operations are database related and they are done by server and database modules.

In the diagram below, structural decomposition and the functions that form it can be seen. Functions are grouped with respect to their generalized features.

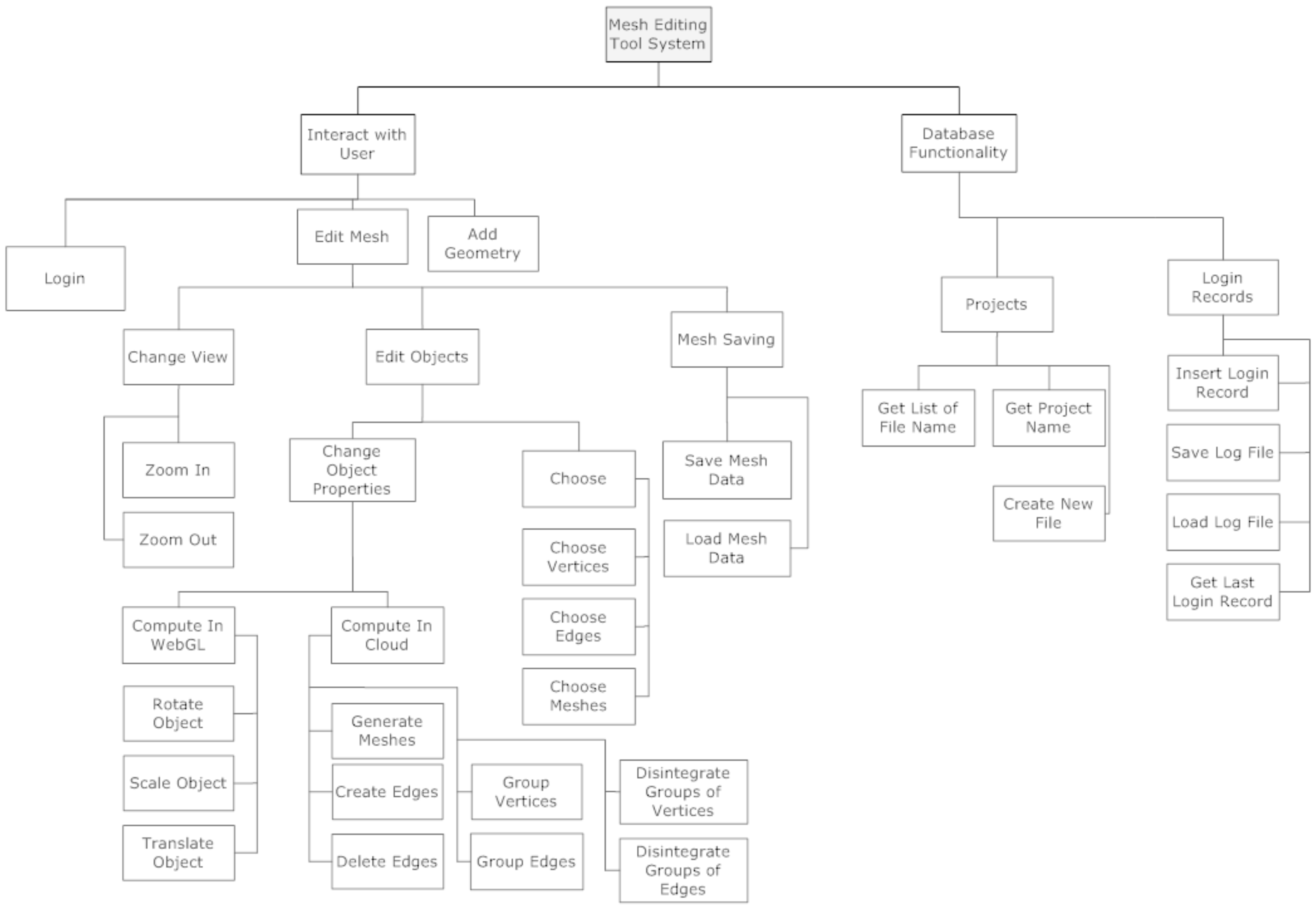


Figure 1: Structural decomposition diagram of the system

3.1.1. Database Module

This module is for storing whole data of the system. Users' log histories, saved mesh data, static geometry lists, information about mesh editing tools are stored in the database. When user wants to use the program, first s/he must login to the system. After logged in the user can use the program and reach the database. By doing this, user's data can be saved to database or previously saved data can be loaded into application page. Besides, some properties of the system can be stored in the database.

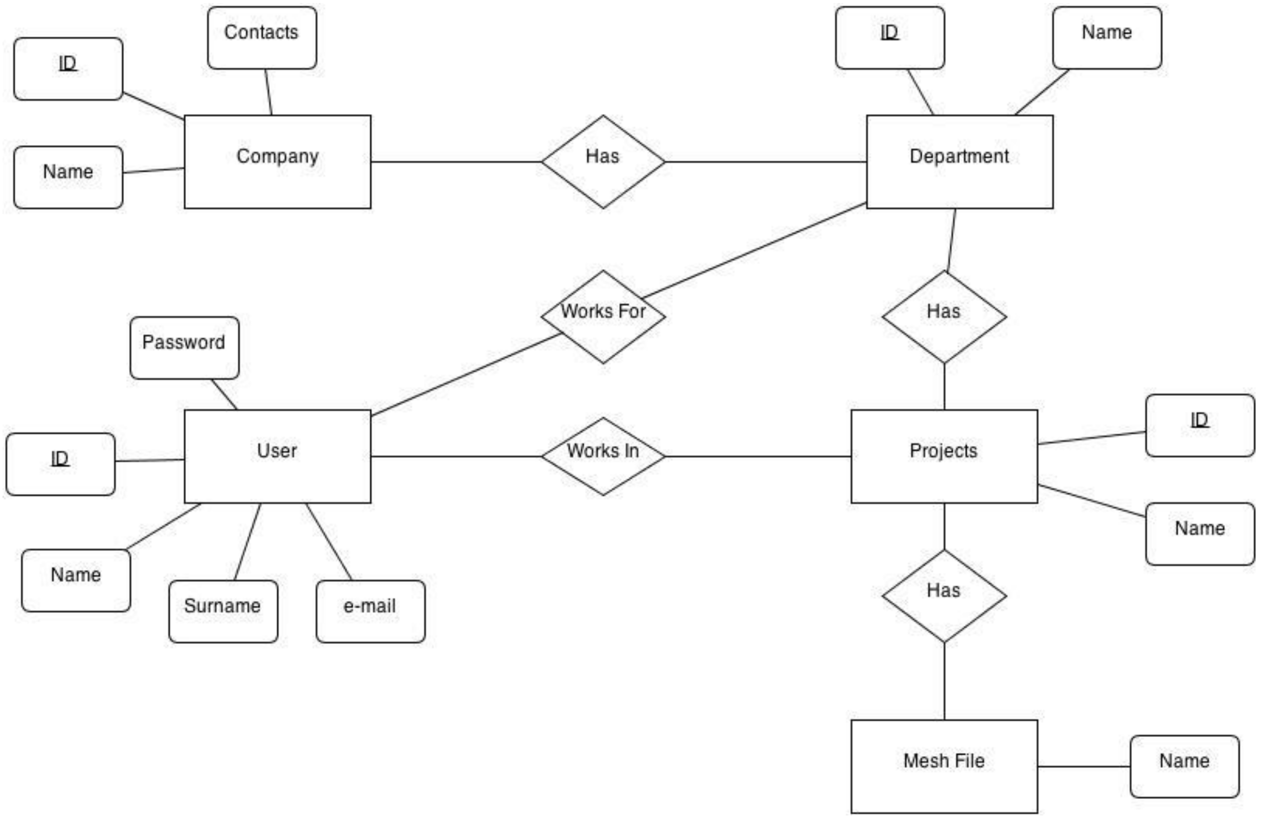


Figure 2: Entity – Relationship Diagram

3.1.2. Browser Module

This module is for interacting with users and displaying meshes. As the name implies it consists web page and the functionalities that exists within the page. Users can select editing options they needed and edit the mesh directly. All the GUI components are part of this module. In addition to these, WebGL component, where shaders and draw calls exist, is included in here. Namely, all the graphical properties are part of this module.

3.1.3. Cloud Module

This module consists of the works that will be handled in the cloud side of the program. When the user wants to edit the mesh, information of the mesh will be sent to cloud via python server. Then required computation such as generating meshes for certain objects, giving specific properties to a group of meshes will be done by this module. After the computations are done, resulting data sent back to browser module to be displayed on the screen.

3.1.4. Server Module

Server Module is hub between database, cloud and browser module. The server supplies connection to database and all the database operations handled by the server. The server also provides the data path between browser and cloud. All the data passes to cloud from the server. Naturally, server also delivers the dynamically created web pages to browser using HTTP. For developing the server, CherryPy framework has been used since it is fast, easy to use and open source.

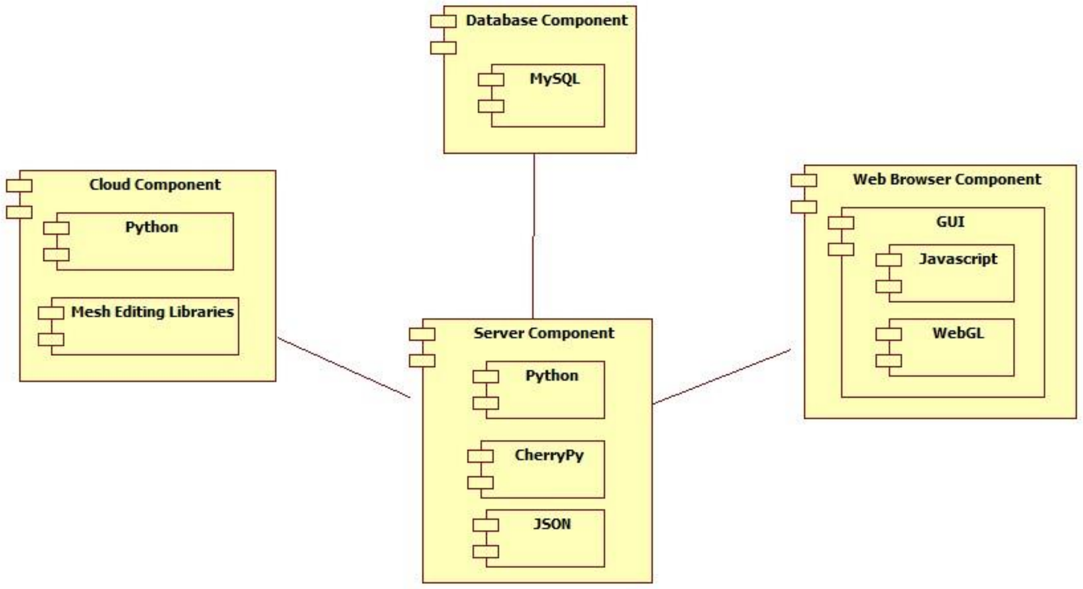


Figure 3: Diagram showing the subsystems 1

3.2. Decomposition Description

Each module has following functions and function descriptions are listed below.

Browser Module:

login(), draw(), saveMeshData(), loadMeshData(), translateObject(), rotateObject(), scaleObject(), addGeometry(), zoomIn(), zoomOut(), chooseVertices(), chooseEdges(), chooseMeshes(), uploadMesh(), findNeighbor();

Server Module:

login(), saveMeshData(), loadMeshData(), saveLogFile(), loadLogFile(), parse(), sendCloudFunction() , uploadMesh();convert();

Cloud Module:

createEdges(), deleteEdges(), generateMeshes(), groupEdges(), groupVertices(), neighborSelection();

Database Module:

getLastLoginRecord(), insertLoginRecord(), createNewFile(), getListofFileName(), getProjectName()

Some of the functions belong to more than one module, since the user request and computation are done in different modules. Those functions are login, saveMeshData, loadMeshData. Modules of each function are mentioned in detail in the section below.

3.2.1. login()

This function takes user name and password as input then checks them. If true it lets the user to use the program and to reach his/her data in database. First, the operation is requested in browser module and after that it is completed in server module. (Reference to 3.21 at SRS)

3.2.2. draw()

This function takes a vertex list and index list as input. Then it draws the given object to the screen with respect to given lists.

3.2.3. saveMeshData()

This function takes a file name as input. Then it saves the mesh data of the current mesh that is displayed on the screen to the file which is given as input. First, the request is done by user in browser module and after that in server module saving operation is finished. (Reference to 3.18 at SRS)

3.2.4. loadMeshData()

This function takes a file name as input. Then it gets the mesh data in the given file, which is stored in the database, and draws the mesh on the screen and adds the object Id to the object list. This function remains in both browser module and server module. User does the request in browser module and operation continues in server module to load mesh data.

(Reference to 3.19 at SRS)

3.2.5. translateObject()

This function takes an object Id and a 3D vector. Then it translates the vertices of the objects by the given vector.

Reference to (3.13 at SRS)

3.2.6. rotateObject()

This function takes an object Id, an axis and a degree as input. Then it rotates the object around the given axis by the given degree.

(Reference to 3.9 at SRS)

3.2.7. scaleObject()

This function takes an object Id and scale factor as input. Then is scales the object with respect to given scale factor.

(Reference to 3.12 at SRS)

3.2.8. addGeometry()

This function takes a pre-computed geometric object Id as input. Then it draws the object in the middle of the screen and adds the object ID to the object list.

(Reference to 3.1 at SRS)

3.2.9. createvertices()

This function takes a point coordinate in the scene as input. Then it adds a new vertex to the vertices list and draws a circle to the given point.

(Reference to 3.14 at SRS)

3.2.10. deleteVertices()

This function takes a list of vertex numbers as input. It removes them from vertex list and index list. Then it rearranges the index list and the object that is displayed in the scene.

(Reference to 3.15 at SRS)

3.2.11. createEdges()

This function takes two vertexes ID as input. It creates a new edge in the index list and draws an edge between the given vertices in the screen.

(Reference to 3.16 at SRS)

3.2.12. deleteEdges()

This function takes a list of index numbers as input. It removes them from the index list then rearranges the list and the object which is displayed in the scene.

(Reference to 3.17 at SRS)

3.2.13. zoomIn()

This function takes a zoom factor value as input. Then it moves the camera through camera’s viewing direction by the given zoom factor.

(Reference to 3.8 at SRS)

3.2.14. zoomOut()

This function takes a zoom factor value as input. Then it moves the camera through inverse direction of the camera’s viewing direction by the given zoom factor.

(Reference to 3.8 at SRS)

3.2.15. chooseVertices()

This function takes mouse coordinates as input when mouse is clicked and hold. It finds the vertices which are in the area that is decided via mouse movement. Then the index numbers of these vertices are stored in a list, created for holding current vertices to be edited.

(Reference to 3.4 at SRS)

3.2.16. chooseEdges()

This function takes mouse coordinates as input when mouse is clicked and hold. It finds the edges which are in the area that is decided via mouse movement. Then the index numbers which, forms these edges, are stored in a list, created for holding current edges to be edited.

(Reference to 3.3 at SRS)

3.2.17. chooseMeshes()

This function takes mouse coordinates as input when mouse is clicked and hold. It finds the meshes which are in the area that is decided via mouse movement. Then the index numbers of these meshes are stored in a list, created for holding current meshes to be edited.

3.2.18. groupVertices()

This function takes group name and vertices that will be grouped as input parameter. Then it writes the indices stored in the current vertex list to an array that represents this group.

(Reference to 3.10 at SRS)

3.2.19. groupEdges()

This function takes group name and edges that will be grouped as input parameter. Then it writes the edges stored in the current edges list to an array that represents this group.

(Reference to 3.10 at SRS)

3.2.20. generateMeshes()

This function takes an object Id and mesh type as input. Then it divides the object to small groups of meshes with respect to given mesh type. This function uses certain triangulation methods provided by EDA

(Reference to 3.2 at SRS).

3.2.21. saveLogFile()

This function works automatically after user done any operation in the browser module. All the information will be stored in the database.

3.2.22. loadLogFile()

This function returns the entire log that is done by the user to browser module. User can see what s/he has done while working on the project.

3.2.23. getLastLoginRecord()

This function takes userId as parameter. It is called when user logged in. It returns last login date and ip of computer last login happened which are stored on userLogHistory table on database.

3.2.24. insertLoginRecord()

This function takes userId as parameter. It is called when user logged in. It inserts userID, current date, and ip of users computer to userLogHistory table on database.

3.2.25. insertFile()

This function called when new file is created and takes filename and name of project as parameter. It inserts these records to MeshFile table on database.

3.2.26. getProjectName()

This function takes project userId as parameter. It returns names of projects for which user works currently which are stored on project table on database.

3.2.27. getListOfFile()

This function takes project name as parameter. It returns list of file names of the project which are stored on meshFile table on database.

3.2.28. uploadMesh()

This function takes mesh file name with ".obj", ".ply" or ".3ds" format as input then it calls convert function of server module. First, the operation is requested in browser module and after that it is completed in server module.

(Reference to 3.6 at SRS)

3.2.29. convert()

This function takes name that comes from uploadMesh() function as input and converts the file data to JSON format then stores it in the server.

3.2.30. neighborSelection()

This function takes type and index of selected part (edge, vertex, polygon) and depth of neighborhood as input. Then finds its neighbors with respect to given depth and add them to selected section.

(Reference to 3.23 at SRS)

3.2.31. findNeighbor()

This function takes type and index of selected part (edge, vertex, polygon) and depth of neighborhood as input and sends it to sendCloudFunction() function in server module.

3.2.32. sendCloudFunction()

This function provides interaction between browser module and cloud module. Function that is called by user in browser module sends its data to sendCloudFunction() then matching cloud function will be called.

Data flow diagram of the system

In our system, the user starts the data flow by giving commands to the browser module, using the functions mentioned above in 3.2. But for a user to use all of the commands, first s/he needs to verify his/her account with login command. This verifies operation is handled between server and database module. User can interact only with browser module as can be seen from the diagram, and only the displayed scene is shown to the user as result. All other operations are done between other components without user interaction.

After verification, the user can load his/her mesh data by using browser module but browser module gets the data from the database module through server module, which acts as a core module. Then the user can edit the mesh, change the position of camera, edit the scene and do all the specified functionalities by user commands.

When a user gives a command, if it does not require mesh editing computation then it is handled in browser module. However, if it requires, the data is sent to cloud module through server module. After the computation is finished then the result is sent back to browser module to be displayed to user. Displaying the result is done by browser module with draw calls of WebGL.

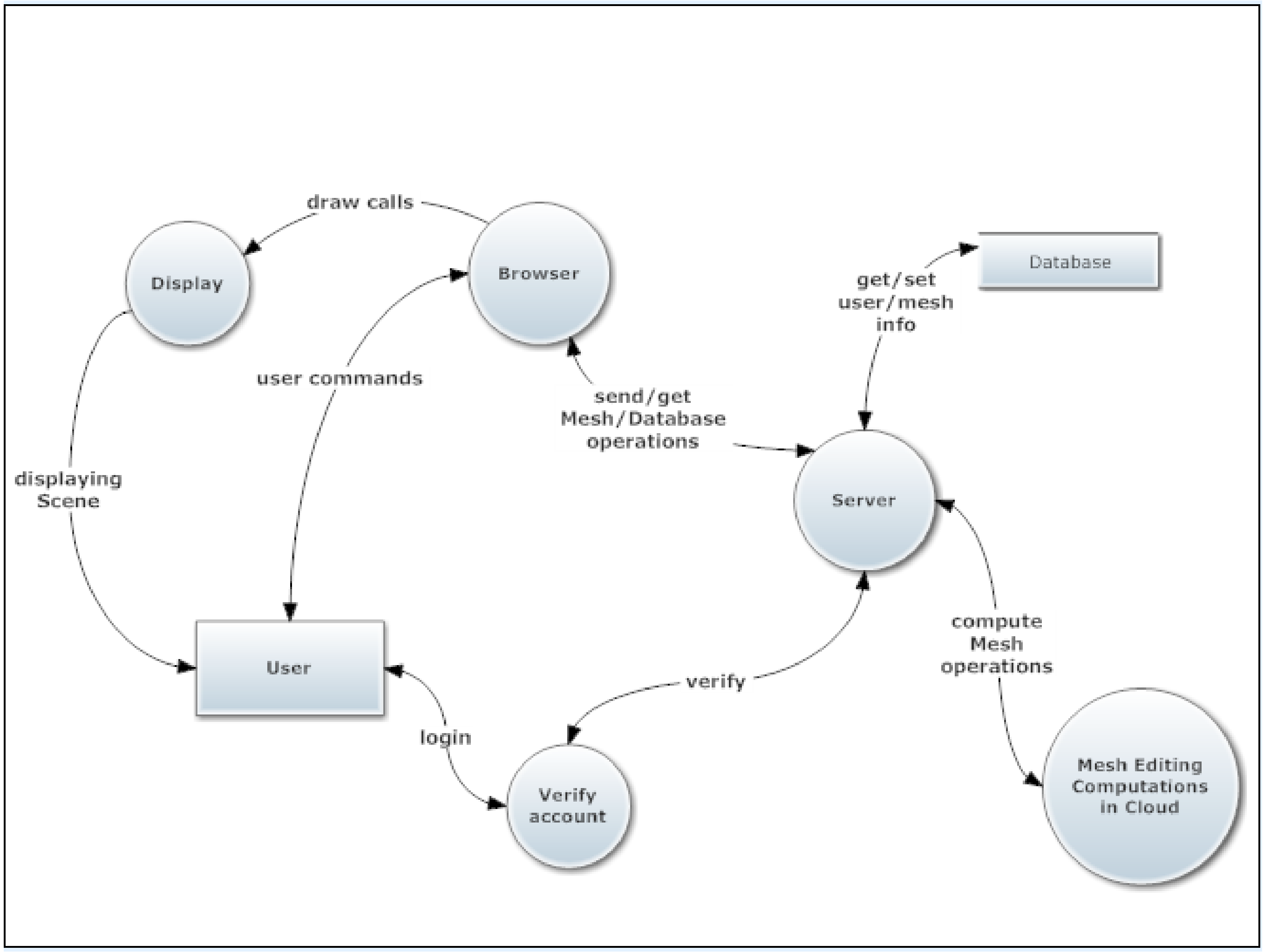


Figure 4: Data flow diagram of the system

3.3. Design Rationale

Developers of the Cloudy Mesh decided to use this architecture that is mentioned above. Developing and designing the system will be simpler by using this architecture. Also testing and debugging will be much simpler by packaging the system.

4. Data Design

4.1. Data Description

4.1.1. Vertices Array

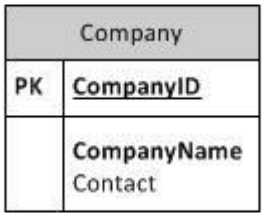
Vertices array stores all the vertices data in tightly packed array. For every vertices there are three floats which contains x, y, z coordinates of the vertices then there are three floats which contains r, g, b components of the color of the vertices.

4.1.2. Indices Array

Indices array stores faces of the polygons as integer doublets for lines, integer triplets for triangles, integer quadruplets for quads.

4.1.3. Tables on Database

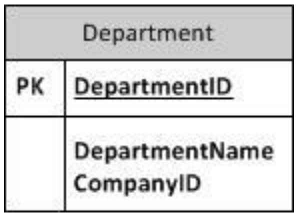
4.1.3.1. Company



Company	
PK	**CompanyID**
	CompanyName Contact

This table keeps information of Companies. CompanyID is primary key of this table and CompanyName is unique field of this table.

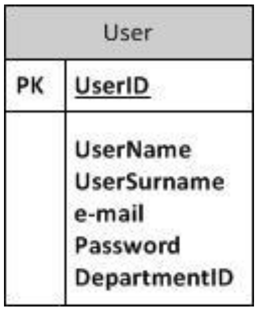
4.1.3.2. Department



Department	
PK	DepartmentID
	DepartmentName CompanyID

Department table stores information of departments. DepartmentID is primary key of this table and CompanyID is foreign key of Company table. DepartmentName is weak entity, i.e., each company has unique DepartmentName.

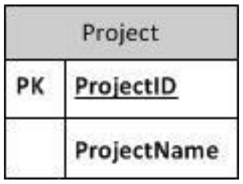
4.1.3.3. User



User	
PK	UserID
	UserName UserSurname e-mail Password DepartmentID

User table stores information of users. UserID is primary key of table. E-mail is unique filed of table, because each user have to have unique e-mail. DepartmenID is foreign key of Department table.

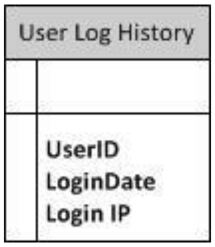
4.1.3.4. Project Table



Project	
PK	ProjectID
	ProjectName

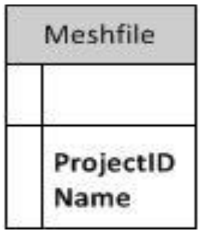
This table keeps data of projects. On this table, ProjectID is primary key.

4.1.3.5. User Log History



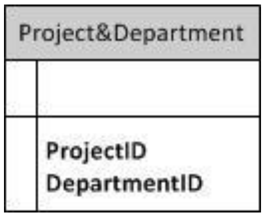
This table is created to save login history of each user. When user is logged in, user id, login date and ip is inserted to table. UserID is foreign key of user table.

4.1.3.6. Mesh File



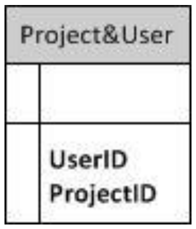
ProjectID is foreign key of project table and name field is a weak entity, i.e., each project has unique file name.

4.1.3.7. Project & Department



In this table ProjectID is foreign key of Project table and DepartmentID is foreign key of department table. This table is created to keep which departments are involved to do each project.

4.1.3.8. Project & user



In this table ProjectID is foreign key of Project table and UserID is foreign key of user table. This table is created to keep which users are involved to do each project.

4.2. Data Dictionary

Server Module		
login()	Function Name	
username	Username of the user	string
password	Password of the user	string
saveMeshData()	Function Name	
filename	Filename of the mesh	string
loadMeshData()	Function Name	
filename	Filename of the mesh	string
saveLogFile()	Function Name	
loadLogFile()	Function Name	
uploadMesh()	Function Name	
filename	Filename of the mesh	string
convert()	Function Name	
filename	Filename of the mesh	string
sendCloudFunction()	Function Name	
functionname	Name of the Function	string
parameters	Parameters for function	String

Browser Module		
login()	Function Name	
username	Username of the user	string
password	Password of the user	string
draw()	Function Name	
vertices	Vertex array of the object	float3*
indices	Index array of the object	int*
saveMeshData()	Function Name	
filename	Name of the file	string
loadMeshData()	Function Name	
filename	Name of the file	string
translateObject()	Function Name	
id	Id of the Object	int
translation	Changes in coordinates of the object	float3
rotateObject()	Function Name	
id	Id of the Object	int
axis	Axis of the rotation	int
angle	Change in angle of the object	float3
scaleObject()	Function Name	

id	Id of the Object	int
factor	Change in size coefficient of the object	float
addGeometry()	Function Name	
id	Id of the Object that will added to screen	int
zoomIn()	Function Name	
factor	Coefficients that will multiplied with distance	float
zoomOut()	Function Name	
factor	Coefficients that will multiplied with distance	float
chooseVertices()	Function Name	
coord1	Mouse coordinates before start dragging	float3
coord2	Mouse coordinates after finish dragging	float3
chooseEdges()	Function Name	
coord1	Mouse coordinates before start dragging	float3
coord2	Mouse coordinates after finish dragging	float3
chooseMeshes()	Function Name	
coord1	Mouse coordinates before start dragging	float3
coord2	Mouse coordinates after finish dragging	float3
uploadMesh()	Function Name	
filename	Filename of the mesh	string

findNeighbor()	Function Name	
index	Index of Edge,Vertex or Polygon	int
type	Type of selected part of mesh	string
depth	Depth of neighborhood	int

Cloud Module		
createEdges()	Function Name	
vertex1	Vertex ID	int
vertex2	Vertex ID	int
deleteEdges()	Function Name	
delete_indices	Array of indices that will be deleted	int*
generateMeshes()	Function Name	
Id	ID number of the object	int
Type	mesh type of the triangulation operation	string
groupEdges()	Function Name	
groupName	name of the group that will be generated	string
Edges	list of edges that will make a group	edges*
groupVertices()	Function Name	

groupName	name of the group that will be generated	string
Vertices	list of vertices that will make a group	vertex*
neighborSelection()	Name of Function	
index	Index of Edge,Vertex or Polygon	int
type	Type of selected part of mesh	string
depth	Depth of neighborhood	int

Database Module		
getLastLoginRecord()	Function Name	
userId	User ID	int
insertLoginRecord()	Function Name	
userId	User ID	int
createNewFile()	Function Name	
fileName	Name of file which will be created	string
getProjectName()	Function Name	
userId	User ID	int

getFileName()	Function Name	
projectName	Name of project	string

5. Component Design

Here pseudo-codes of the functions can be seen:

5.1. login()

```
login(string username, string password):
    result = database query result with username and password
    if result == empty:
        throw error
    else:
        create session with username
        redirect to main page
```

5.2. draw()

```
draw(float3 vertices[], int indices[]):
    map vertex buffer
    copy vertices array into vertex buffer
    map index buffer
    copy indices array into index buffer
    make draw call
```

5.3. saveMeshData()

```
saveMeshData(string filename):
    open file
    write number of the vertices
    for each vertex in vertices in meshes:
        write position vector to file
    write number of indices
    for each index in indices in meshes:
        write index to file
    close file
```

5.4.loadMeshData

```
    loadMeshData(string filename):
        open the file
        read number of vertices
        for i=0 to number of vertices:
            read position vector
            push position vector into vertices array
        read number of indices
        for i=0 to number of indices:
            read index
            push index into vertices array
        close the file
```

5.5. translateObject()

```
    translateObject(Object id, float3 translation):
        get the object with id
        construct translation matrix from translation vector
        for each vertex in vertices:
      vertex_position = translation_matrix * vertex_position
```

5.6. rotateObject()

```
    rotateObject(Object id, int axis, float degree):
        get the object with id
        construct rotation matrix from axis and degree
        for each vertex in vertices:
```

```
    vertex_position = rotation_matrix * vertex_position
```

5.7. scaleObject()

```
scaleObject(Object id, int axis, float factor):
        get the object with id
        construct scale matrix from scale factor
        for each vertex in vertices:
    vertex_position = scale_matrix * vertex_position
```

5.8. addGeometry()

```
addGeometry(int Id)
        for each object in objectArray
                if object.Id == Id
                        return draw()
```

5.9. createVertices()

```
createVertices(float3 coord):
        push coord into vertices array
```

5.10. deleteVertices()

```
deleteVertices(int vertices_indices[])
        for each delete_index in vertices_indices:
                delete vertex from vertices
                for each index in indices
                        if index == delete_index :
                            delete index from indices
```

5.11. createEdges()

```
createEdges(int vertex1, int vertex2):
        push vertex1 into indices array
        push vertex2 into indices array
```

5.12. deleteEdges()

```
deleteEdges(int delete_indices):
        for each delete_index in delete_indices:
                for each in index in indices_array:
                        if index  == delete_index:
                                delete index from indices
```

5.13. zoomIn()

```
zoomIn(float factor)
        cam_translate =  factor * view direction
        add cam_translate to view matrix’s traslation coluonm
```

5.14. zoomOut()

```
zoomOut(float factor)
        cam_translate =  factor * -1 * view direction
        add cam_translate to view matrix’s traslation coluonm
```

5.15. chooseVertices()

```
chooseVertices(float3 coord1,float3 coord2)
    find and push vertices index into chosenVerticesList
    for each vertices in chosenVerticesList
            give different color to vertices
```

5.16. chooseEdges()

```
chooseEdges(float3 coord1,float3 coord2)
    find and push vertices index into chosenEdgesList
    for each vertices in chosenEdgesList
    give different color to edges
```

5.17. chooseMeshes()

```
chooseMeshes(float3 coord1,float3 coord2)
    find and push vertices index into chosenMeshesList
    for each vertices in chosenMeshesList
    give different color to meshed
```

5.18. groupVertices()

```
groupVertices(string groupName,vertex* Vertices)
    create an group array with name groupName
    push every Vertices in Array
```

5.19. groupEdges()

```
groupEdges(string groupName,edges* Edges)
    create an group array with name groupName
    push every Edges in Array
```

5.20. generateMeshes()

```
generateMeshes(int Id, string Type)
    find Object with id Id
    execute mesh algorithm with Type on Object
```

5.21. saveLogFile()

```
saveLogFile():
    insert last operation to database
```

5.22. loadLogFile()

```
loadLogFile():
    query database to acquire log information
    return the result
```

5.23. getLastLoginRecord()

```
getLastLoginRecord(int userId)
    get last login date from database
    get ip of computer last login happened
    return ip and date
```

5.24. insertLoginRecord()

```
insertLoginRecord(int userID)
    use procedure implemented in SQL, give userID and ip as parameter
```

5.25. createNewFile()

```
createNewFile(string filaName)
    create file on hard disc
    insert records on database
```

5.26. getProjectName()

```
getProjectName(int userID)
    getProjectNames from database
    for each name
        push name on nameList
return nameList
```

5.27. getFileName()

```
getListofFileName(string projectName)
    getfilename from database
    for each name
        push name on fileNameList
return fileNameList
```

5.28. uploadMesh()

```
uploadMesh()
    open file chooser dialog
    post file name and file data to server's uploadMesh() function
```

5.29 upload Mesh()

```
uploadMesh(string filename, string filedata)
    convert(filename, filedata)
```

5.30. convert()

```
Convert(string filename, string filedata)
    Parse filedata according to obj specification
    Write mesh data in json format with filename
```

5.31 findNeighbor()

```
findNeighbor(string type, int index, int depth)
    check type
    according to type find vertices, edge or face
    sendCloudFunction("findNeighbour", parameterlist)
```

6. Human Interface Design

6.1. Overview of User Interface

Cloudy Mesh has a convenient user interface that can be used by a user that has introductory level of experience in CAD programs. In order to achieve this reliability, developer of the Cloudy Mesh has implemented and designed the system as clearly as possible.

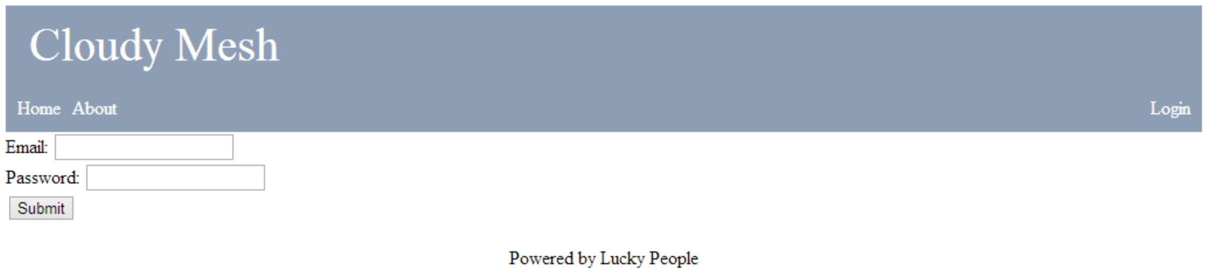
Buttons and menus are grouped according to its context. Therefore users can easily find what they are searching. Since Cloudy Mesh is a visualized program all the operations that done by the user have a feedback. And all the feedbacks can be seen in the program environment.

As it can be seen in the pictures of the GUI design below, project team first main aim is to keep the system simple.

All the pictures in the section 6.2 will be explained thoroughly in the section 6.3.

One can see the GUI design, screen objects and actions below, but the environment may be changed according to the project leader opinions and changes in the requirements.

6.2. Screen Images



Powered by Lucky People

Figure 1 : Login Screen

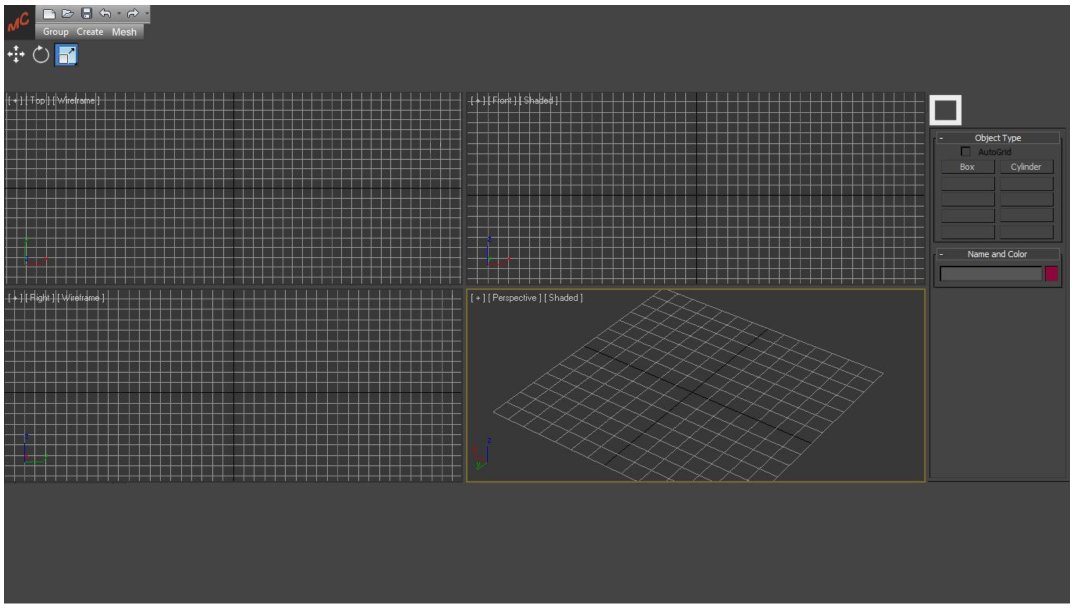

Group
Create
Mesh
[+] [Top] [Wireframe]
[+] [Front] [Shaded]
[+] [Right] [Wireframe]
[+] [Perspective] [Shaded]
Object Type
AutoGrid
Box
Cylinder
Name and Color

Figure 6 : Menu Design (SI 01)

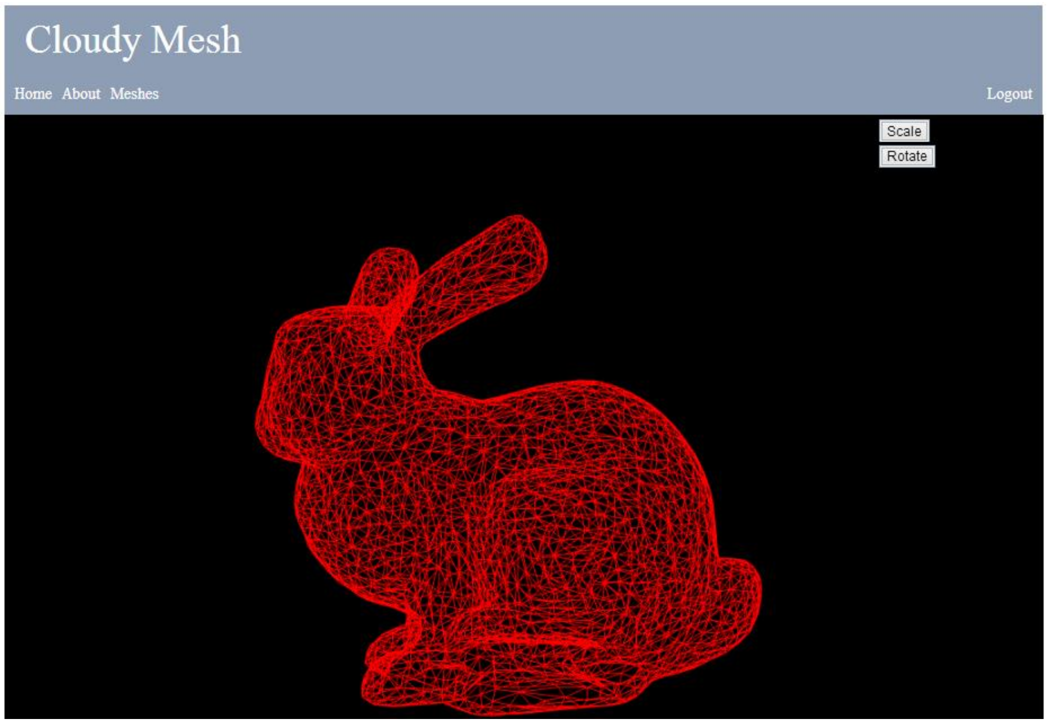


Figure 7 : Objects (SI 012)

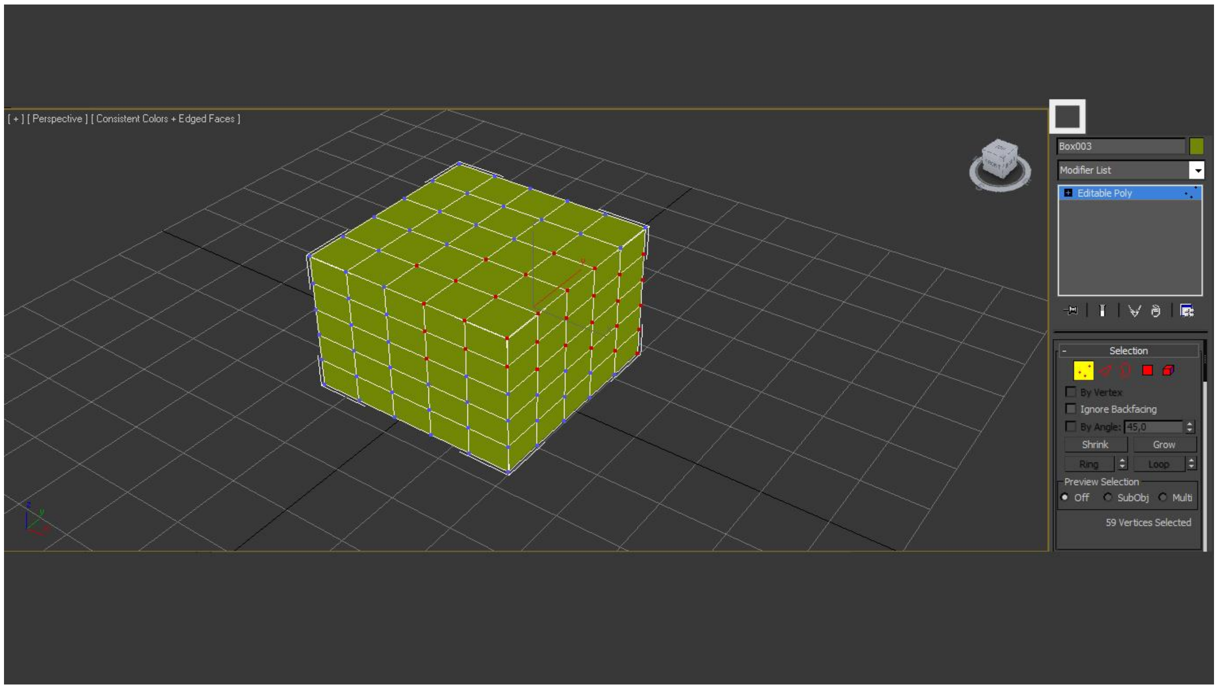


Figure 8 : Vertex Selection (SI 03)

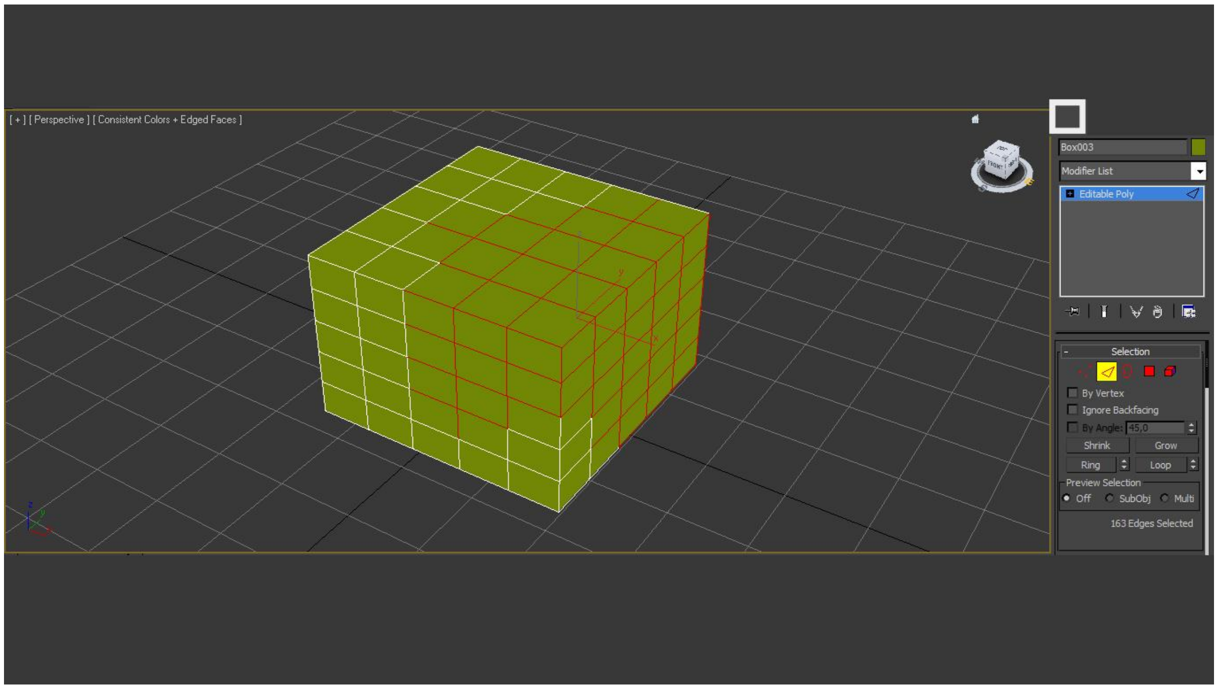


Figure 9 : Edge Selection (SI 04)

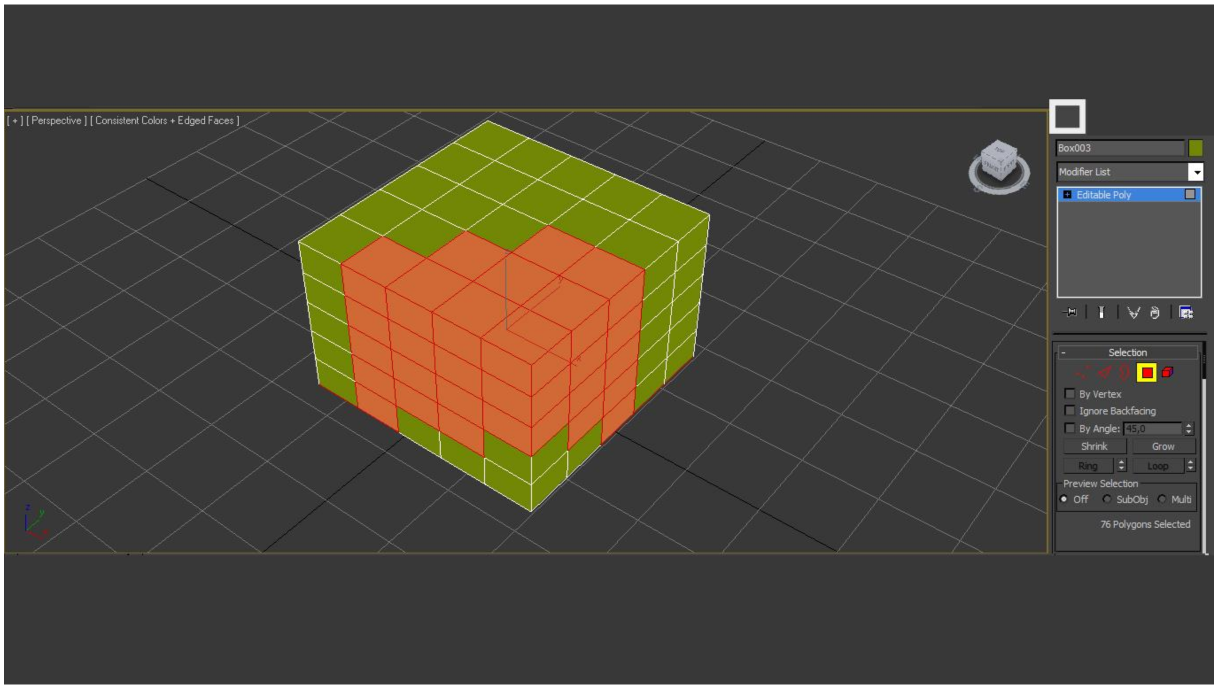


Figure 10 : Mesh Selection (SI 05)

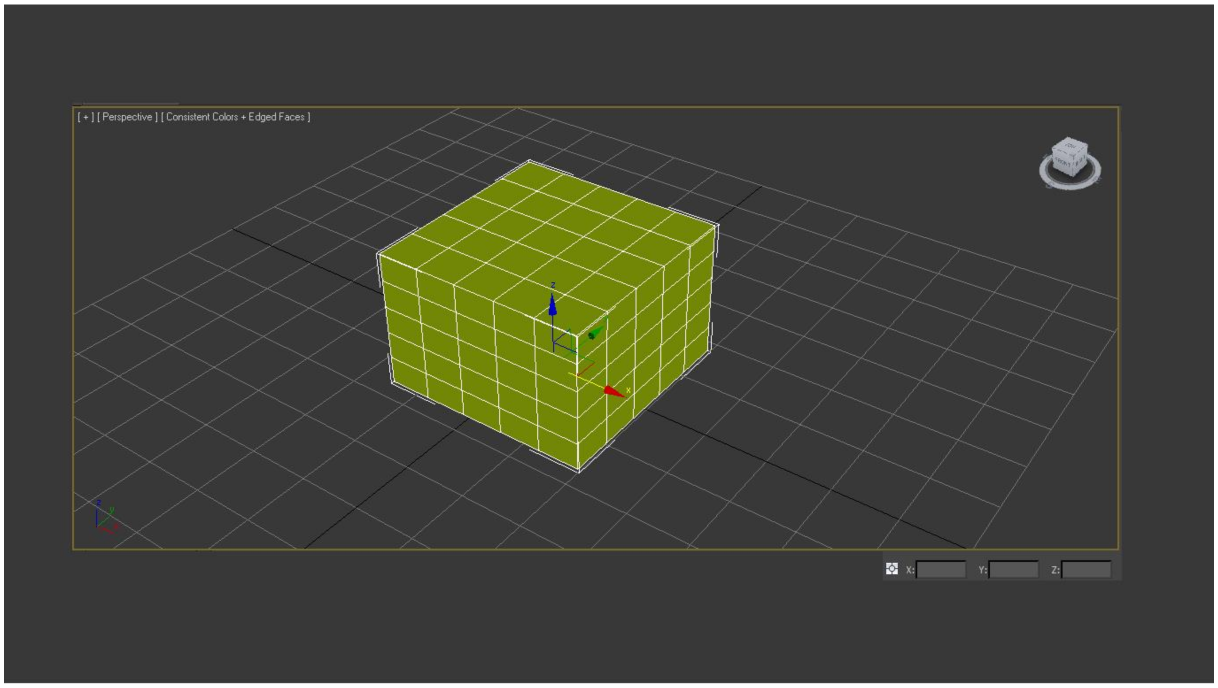


Figure 2 : Translate (SI 06)

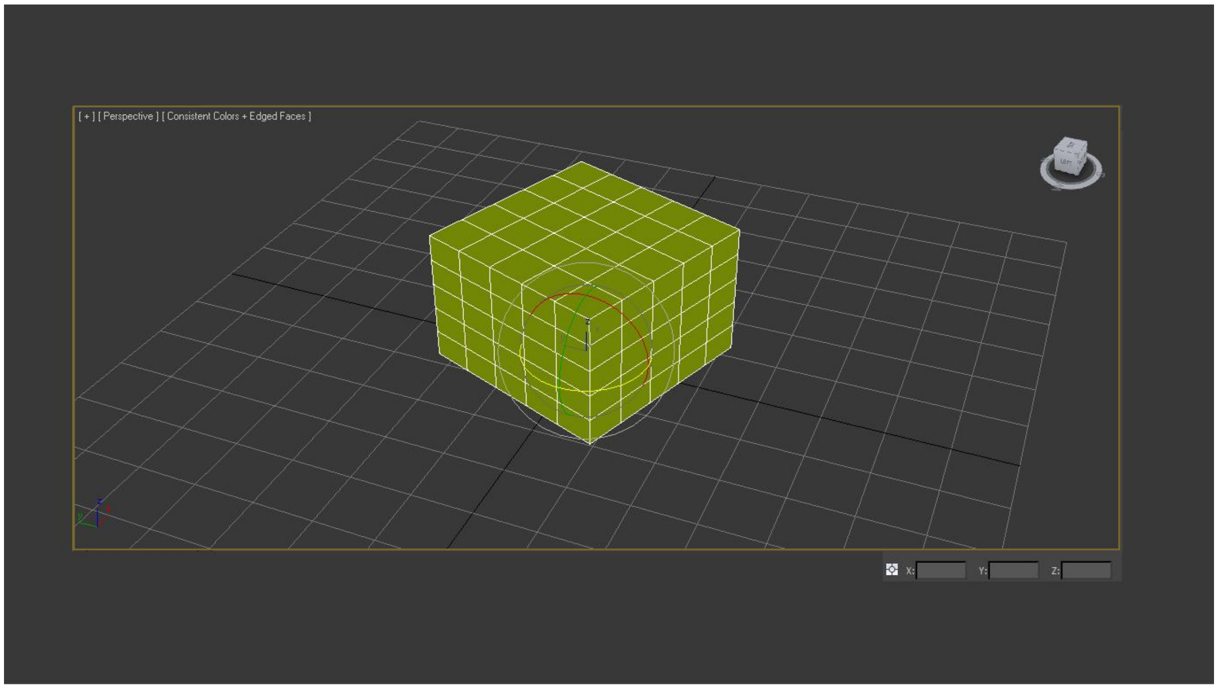


Figure 3 : Rotate (SI 07)

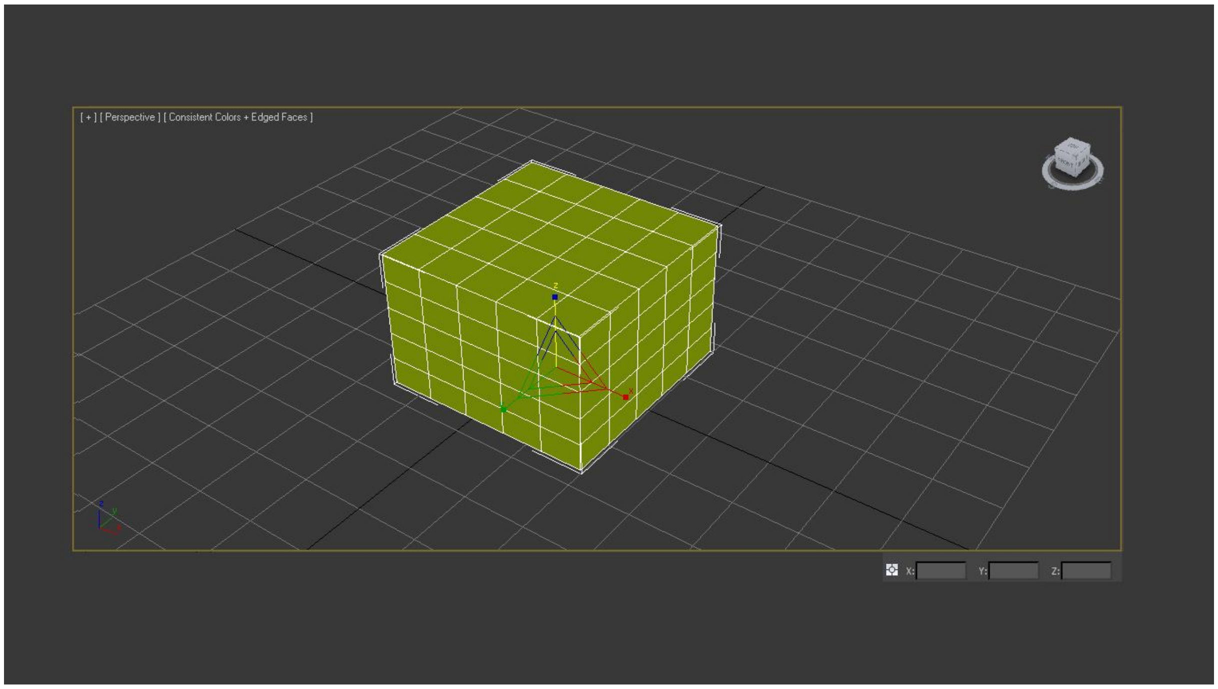


Figure 4 : Scale (SI 08)

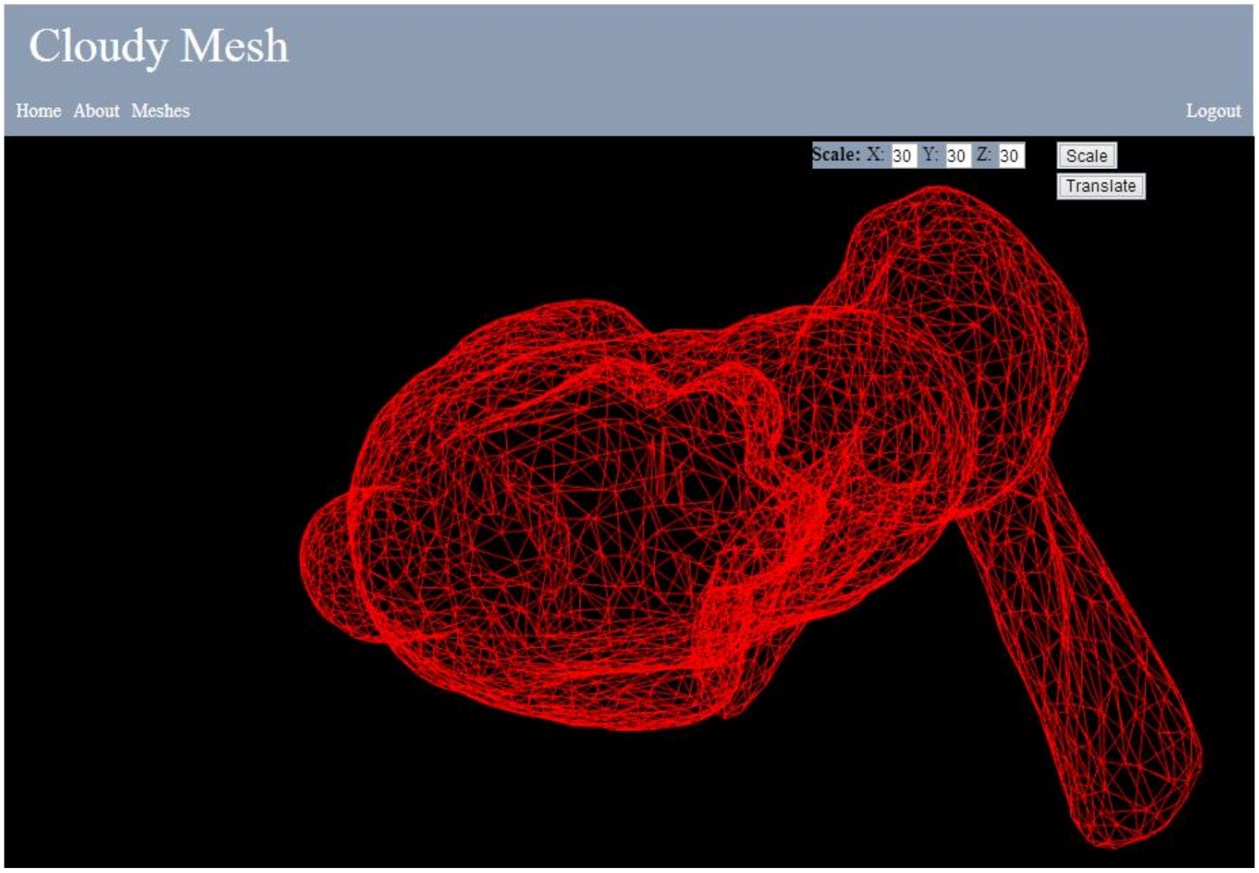


Figure 5 : Early Prototype (SI 09)

6.3. Screen Objects and Actions

In the image SI 01, one can see the environment area that has 4 different screen and buttons. The screens are enable users to see their objects from different positions and make decisions according to it easily. The perspective camera screen is the most commonly used screen, therefore other images are taken while perspective camera screen is maximized.

On the top left of the environment screen, New File, Open File, Save File, Undo and Redo button are placed accordingly. Below of these buttons, there are Group, Create and Mesh menu accordingly. In the Group menu Group and Ungroup buttons are exist. User can group objects or ungroup them by using these buttons. In the Create menu, user will see all the predefined objects and can choose one from there. User can also create object by clicking hollow square that is at the right side of the screen. After user clicks square Object Type and Name and Color parts will be visible (that can be also seen in the image SI 01) and user can choose a predefined object from there. Predefined objects can be drawn easily. In the image SI 02 one can see an environment area that contains predefined objects.

Vertex Selection is one of the important features of Cloudy Mesh. User can select vertices by clicking left button of and dragging mouse. But in order to select vertices user must click on the object that s/he want to choose vertices from first. After user clicks the object Selection menu will be visible. In that menu user clicks the first button (that is highlighted as yellow in IS 03) and then user can choose vertices.

Edge Selection can be done easily in Cloudy Mesh. User can select edges by clicking left button of and dragging mouse. But in order to select edges user must click first on the object that s/he want to choose edges from. After user clicks the object Selection menu will be visible. In that menu user clicks the second button (that is highlighted as yellow in IS 04) and then user can choose vertices.

Mesh Selection also can be done easily in Cloudy Mesh. User can select meshes by clicking left button of and dragging mouse. But in order to select meshes user must click first on the object that s/he want to choose meshes from. After user clicks the object Selection menu will be visible. In that menu user clicks the second button (that is highlighted as yellow in IS 05) and then user can choose vertices.

User can translate the object by clicking the first button in 3rd row at the top left corner of the screen (navigation like button). After user clicks that button and choose an object, there will be a 3D coordinate system at the middle of the object. By dragging the line that is intended dimension, user can drag the object. User can also translate the object with using the input text boxes at the right bottom of the screen, coordinates x, y, z accordingly. Coordinate system can be seen at the image IS 06

User can rotate the object by clicking the second button in 3rd row at the top left corner of the screen (circle like button). After user clicks that button and choose an object, there will be a 3D sphere coordinate system at the middle of the object. By dragging the arc that is intended dimension, user can rotate the object. User can also rotate the object with using the input text boxes at the right bottom of the screen, angle x, y, z accordingly. Coordinate system can be seen at the image IS 08

User can scale the object by clicking the third button in 3rd row at the top left corner of the screen. After user clicks that button and choose an object, there will be a 3D coordinate system at the middle of the object. By dragging the line that is intended dimension, user can scale the object. User can also scale the object with using the input text boxes at the right bottom of the screen, size coefficients of x, y, z accordingly.

In IS 09, one can see the early version of the Cloudy Mesh. In that image the object “bunny” is rotated and translated. After those operations are done, scale function is applied. Buttons “Scale” and “Translate” and Textbox for Scale interact with the browser module to do its operation.

7. Requirements Matrix

Adding geometries from list to the scene	✔
Generating meshes for certain objects	✔
Choosing Edges with Mouse	✔
Choosing Vertices with Mouse	✔
Choosing Meshes with Mouse	✔
Importing Meshes	✔
Exporting Meshes	✔
Zooming	✔
Rotating Objects	✔
Grouping meshes & vertices & edges and giving specific attributes to groups	✔
Disintegrating the groups of meshes into individual meshes	✔
Scale Objects	✔
Translate Objects	✔
Creating Vertices	✔
Deleting vertices	✔

Creating Edges	✔
Deleting Edges	✔
Save Mesh	✔
Load Mesh	✔
Create an Account	✔
Login into an Account	✔
Logout from an Account	✔

8. Appendices

8.1 Appendix A:Glossary

GUI: Graphical User Interface

CAD: Computer Aided Design.

WebGL: Web graphics library.

HTML5: Hypertext Markup Language

Python: General purpose programming language.

HTTP: Hypertext Transfer Protocol

UML: Unified Modeling Language

SRS: Software Requirement Specification

IEEE: Institute of Electrical and Electronics Engineers