Content

**1.1 Why we need this software.**

1.1.1 Speed

1.1.2. Precision and control

1.1.3. Scenario visualization

1.1.4. Reduced lead times

1.1.5. You will avoid costly mistakes.

**1.2 Purpose**

**2.1 Instruments**

2.1.1 Java

2.1.2 JavaFX

2.1.3 SWING

2.1.4 Git

2.1.5 Maven

Abstract

Peoples’ Friendship University a student’s first Computational Civil Engineering course includes instruction in a high-level computer programming language. Students have learned programming as part of instruction in the JAVA modeling language. Inclusion of computer programming early in the curricula has been seen by the Civil Engineering faculty as a way of improving the students’ skills in logical reasoning, application of technical knowledge, and quantitative problem solving.

Speaking as a structural engineer at least, for the most part you will be doing hand calculations, and using pre-existing software. Of course, these software packages usually allow you to write custom scripts, but they are frosting on the cake, and are usually not necessary for the day to day aspects of your job.

Many software programs are available for each discipline of civil engineering. Most civil engineers practice in specialized subsets of civil engineering, such as geotechnical engineering, structural engineering, transportation engineering, hydraulic engineering, environmental engineering, project and construction management and land surveying.

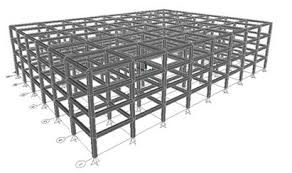
The trend to implement software programs into the civil engineering industry began as educational concerns for the future as civil engineering prepared to enter the twenty-first century. Today, these concerns and trends are centered on the continuing education unit which have become required as part of maintaining the professional license. As a result of the expanding use and demand for these software programs, there was less of a necessity for occupations such as draftsman, because the engineer began to prepare and input the design parameters into the program, thus eliminating the need for manual drafting. Land surveying, a specialized subset of civil engineering, relies heavily on the computerization of the industry. University textbooks have already since begun to include software applications for students to gain experience with some kind of software interface.

Another specific subset, infrastructure design, relies heavily on estimates of load, pressure, drainage and flow. Some software houses have attempted to provide design software catering for the variety of infrastructure design fields in an integrated manner. However, general-purpose software may be used in the same manner at a fraction of the cost of design software. When planning the construction phase, various project management methods are used to estimate factors such as cost, schedule and resourcing. Different software packages rely on different formulas and theories as the basis for these calculations. Consulting engineers also take advantage of the insight software can provide as far as crossing services are concerned. A road design may have to accommodate the presence of underground pipes for example. Civil Designer is an example of a design package which forms an integrated data gathering, drawing, surface modeling and design system for civil engineering infrastructure.

* 1. **Why we need?**

Architects and engineers often rely on drawings and models of their projects to aid in their work. However, 2D computer-aided design models don't always provide these professionals with the speed and accuracy they need to complete projects or find solutions to challenges. 3D modeling has been used by architects around the world for many years to improve the efficiency and aesthetics of their designs.

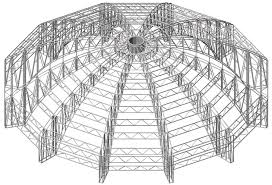
3D modeling provides architects and engineers with the tools they need. Here are the biggest benefits 3D models offer these professionals:



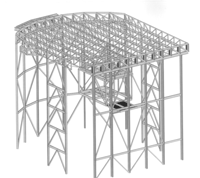
**1.1.1 Speed**

3D modeling is able to virtually construct sites or structures faster than 2D drawings or models can. Professionals need to be able to correctly interpret data the first time, and 3D models leave little question about aspects of a site or structure because they provide a more accurate picture. This means architects and engineers don't have to spend time looking for issues in 2D drawings, allowing them to finish their projects faster.

When comparing newer 3D technology to 2D solutions, 3D offers you some advantages. In 2D, a lot of time and energy is spent ensuring that your plan, section and elevation agree. In 3D, architects and engineers quickly and easily extract that 2D information from a completed model, leaving more time to focus on the design process.



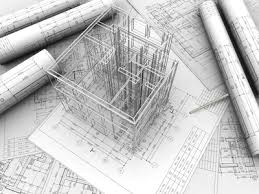
**1.1.2. Precision and control**

3D model, with the help of other structural software, provides accurate as-is data on sites information and professionals can use the output data to create pin-point models of real spaces. This means architects and engineers don't have to spend time measuring and re-measuring parts of a structure or site to develop a precise model, which otherwise

presents expensive mistakes from occurring by ensuring architects and engineers notice design issues or weaknesses in the structural integrity of a site before concrete is poured or a wall is constructed.

**1.1.3. Scenario visualization**

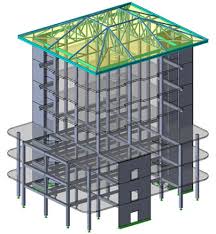
Architects and engineers can also manipulate 3D models in a way they often can't with 2D drawings. Professionals are able to test what-if scenarios with their designs in 3D, helping to validate their plans and identify any problems with design quality. 3D renderings of designs save architects time and money by confirming their project needs. In addition, these types of models can also give architects and engineers an accurate picture of how they can change their designs if they need to. It is much easier and cheaper to alter a project in the design stage rather than after part of the work is already completed.



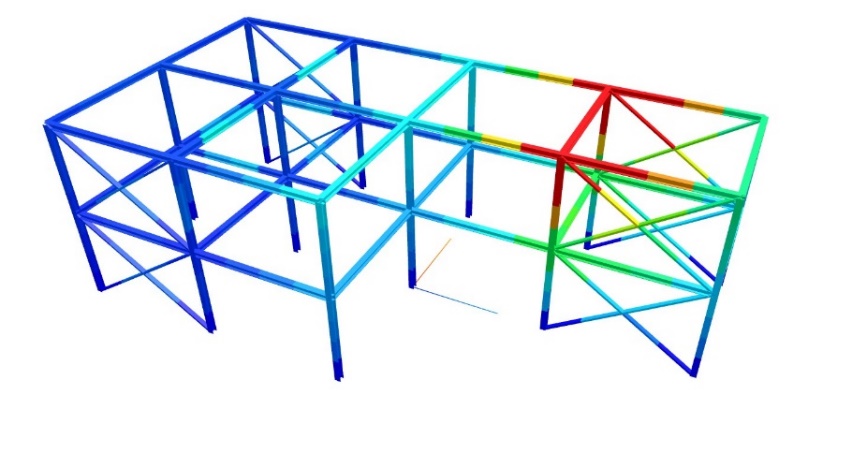
**1.1.4. Reduced lead times**

Because of the accuracy and flexibility of 3D models, architects and engineers are able to spend less time on the design stage of their projects and more time on the actual completion of each task. Professionals are able to identify any issues ahead of time by using 3D modeling, saving them from having to rework schedules and increase budgets.

Models are essential to architects and engineers, but they don't have to remain content with using traditional 2D drawings and renderings of their projects. 3D modeling can give these professionals the advantages they need to finish projects quickly, efficiently and within budget.



**1.1.5. You will avoid costly mistakes.**

3D modeling solutions offer unprecedented insight when creating designs. They give you the ability to test the stress factors and tolerances of a building or product before you build, saving time, money and potentially disastrous consequences. A 3D model also lets you see the end result before it’s built, allowing you to correct issues before it’s too late.3D software goes way beyond design, with support for advanced analysis tools that do everything from simulating the flow of fluid to measuring vibrations in key structural components. Simulating those environmental factors is critical to identifying design flaws and pinpointing serious build problems—before you have to pay for them.

You can also take advantage of 3D printing technologies, which use 3D files to quickly “print” prototypes and models—driving down prototyping costs. With commercial 3D printers coming down in price all the time, even small firms can take part in the action and print on-demand in response to customer need

**1.2 Purpose**

**1.2.1**. I wanted create 3D software for creating beautiful, difficult engineering and architectural models and analyzing them by helping Scad or another softwares. Creating this software I wanted to understand how works 3D software and which processes have.

**2.1 Instruments**

**2.1.1 Java** – I used java because java is object oriented language, for designed to have as few implementation dependencies as possible and it is easy for use. It provides high-level constructs for creating and manipulation 3D geometry and building the structures used in rendering that geometry. With this software, you can efficiently define and render very large virtual worlds.

**2.1.2 JavaFX** – I choose Java FX because it provides some intuitive 3D graphics capabilities in its API library. An interesting aspect is that the 3D API structure of JavaFX has maintained the same lineage of simplicity when compared with its the core API structure. The learning curve is almost flat. If one has dealt with 2D APIs of JavaFX, working with 3D APIs will be almost a cakewalk. The use of functions and objects in 3D programming does not make it look like a hurdle while coding. However, it would be too much of an expectation, at least for now, to have all the function and features that are available in any established 3D API library. However, the progress of JavaFX in the 3D arena is commendably decent, keeping sight of the complexities involved in creating a full-fledged 3D API library.

**2.1.3 SWING –** In Java, it provides a rich set of libraries to create Graphical User Interface (GUI), to make it work independently on different platforms. Swing is definitely the one which is most popular used, because its properties of light-weight, rich-control, easy-customizable. I soon will have to deal with JavaFX in a Swing based and the JFXPanel is a javax.swing.JComponent to embed JavaFX content into Swing-UIs. A JFXPanel can be used similar to a JPanel and can be accessed via the EDT as a generic Swing component except dealing with JavaFX component must be done via the JavaFX application thread.

**2.1.4 GIT -** Version Control Systems are one of the software tools that help developers manage the source code over time. They keep track of the modification, compared with different versions of the source code. As the number of lines increases, it could go up to thousands and thousands lines. In real life, it could be a disaster if we make some irrecoverable mistakes without any backups. However, this will not happen in code-writing for the developers, especially with the help of Git, which is one of the most popular source code management system for software development. Git has been widely used among programmers. Compared with other Version Control Systems, such as SVN (Apache Subversion) and CVS (Concurrent Version System), Git is more powerful by its distributed nature, fast operation and branch handling mechanism. The distributed system makes it easy and efficient for multiple developers work together towards the same project. Git works fast as its lightweight operations. Also every developer can work with the code as its own branch. Others can merge to different branches and this makes cooperation work quite easy. In addition, Git is free and open-source. Because of the efficient of Git, Github, which is the web-based repository hosting service platform, is also popularly used.

**2.1.5 MAVEN** - The maven dependency management mechanism is one of the best features that maven have. I can easily manage all the project dependencies with maven in a easy way, even when I am in a multi-module project with several dependencies, maven can do the task of managing dependencies in a easy way for me. I can inherit dependencies from a parent project, you can define my own dependencies in a project. I can define dependencies scopes. I can import dependencies, this is specially usefull in large projects once I can only inherit from one pom at most.

**2.2.1 Model –** For creating model of our software we need to create a package model and into this package we must create a class with name of Model. In this class we must to create some method for creating and saving our model. Fistful we need to create list for Nodes and Bars. It will help us to save every adding and removing nodes and bars in our model. Then we need to create list for adding removing and changing subscribers. It will help us for giving information to our model about adding, removing and changing process. Then we need to create same models for adding, removing nodes and bars. This method are in the bottoms.

addNode - By helping this method we can add our node in our node list and save it in our model.

removeNode - this method help us to remove our node from node list and in our model.

addBar - This method will be similar addNode method and it will help us to add bar for our bar list and save it in our model.

removeBar - this method help us to remove our bar from node list and in our model.

subscribeAddElement – this method help us to subscribed every adding element to our addSubscriber list.

subscribeRemoveElement - this method help us to subscribed every removing element from our removeSubscriber list.

subscribeChangeElement - this method help us to subscribed every removing element in our changeSubscriber list.

**2.2.2 Node –** For creating Node we are creating into the package of model Node class. For class Node we are needed x, y, z coordinates. For this we are creating x, y, z variables in our class. Then we are creating selected for select our nodes in our model. And we are creating instance of our model for getting all method to using in Node class. We creating index and name variables for given name our nodes like index. We are creating same method in our class for creating node.

Constructor **-** We are creating in class Node a constructor for getting this class from other class. In this constructor we are given x, y, z coordinates and we are given all nodes a name like its index and every nodes name equals “name” + index.

In our model we are creating getter and setter method for getting all information in other class for creating node visually.

Selected – In this method we are giving our model every select nodes.

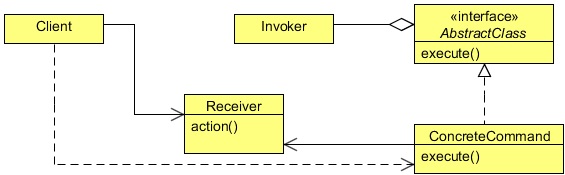
Translate –In this method we are giving our model every translation to our model.

**2.2.3 Bar –** For creating Bar we are creating into the package of model a Bar class. In our class we are creating start node and end node variable for creating bar. Then we need to selected variables for select our bar and for taking all method from Model we are creating instance of model and a name variable for giving our bar a name.

Constructor - for creating bar we give this constructor a stratNode, endNode. We get this startNode and endNode from our model. When we are in our model select first node it will be our startNode and when we select secand node it will be our endNode and for creating bar we must push APPLY or ADD buttons.The name of our bar is startNode add endNode. We are creating a getter and setter for getting our variables from other classes.

2.2.4 Selectable **–** We are creating in our model package a Selectable interface for select our nodes or bar.

**2.2.5 Command** - The Command Design Pattern is a behavioral design pattern and helps to decouples the invoker from the receiver of a request The intent of the Command Design Pattern is to encapsulate a request as an object, thereby letting the developer to parameterize clients with different requests, queue or log requests, and support undoable operations. In general, an object-oriented application consists of a set of interacting objects each offering limited, focused functionality. In response to user interaction, the application carries out some kind of processing. For this purpose, the application makes use of the services of different objects for the processing requirement. In terms of implementation, the application may depend on a designated object that invokes methods on these objects by passing the required data as arguments. This designated object can be referred to as an invoker as it invokes operations on different objects. The invoker may be treated as part of the client application. The set of objects that actually contain the implementation to offer the services required for the request processing can be referred to as Receiver objects. Using the Command pattern, the invoker that issues a request on behalf of the client and the set of service-rendering Receiver objects can be decoupled. The Command pattern suggests creating an abstraction for the processing to be carried out or the action to be taken in response to client requests. This abstraction can be designed to declare a common interface to be implemented by different concrete implementers referred to as Command objects. Each Command object represents a different type of client request and the corresponding processing. A given Command object is responsible for offering the functionality required to process the request it represents, but it does not contain the actual implementation of the functionality. Command objects make use of Receiver objects in offering this functionality.



Command

Declares an interface for executing an operation.

ConcreteCommand

Defines a binding between a Receiver object and an action.

Implements Execute by invoking the corresponding operation(s) on Receiver.

Client

Creates a ConcreteCommand object and sets its receiver.

Invoker

Asks the command to carry out the request.

Receiver

Knows how to perform the operations associated with carrying out a request. Any class may serve as a Receiver.

In our project we create a command package and in this package we create some classes and interfaces for node bar and model commands. This commands help us to execute and unexecuted our codes for model. Also we create a classes for Redo and Undo command.

**2.2.6 Redo Undo command**

I also wanted to implement undo and redo features in an application. We can changes the state of the program, and then wishes to reverse that state to the previous - be it to remove a mistake, make a simple comparison, or follow the progression of their actions. Many java swing components already provide basic undo/redo features, in particular the JTextComponent classes. However these are limited in their ability and quite often it is necessary to implement a custom undo/redo, for example to change a user action that draws to a 3D model, changes a data structure, or a combination of the above. What I wish is to be able to generalize this design, allowing it to be re-used regardless of the user action. Having an interface which defines the undo/redo actions allows any object to implement these actions, to create composites of these interfaces to delegate the action, and therefore allows decoupling of the implementation of the action from how undoes and redoes will be managed.

**3.1.1 User Interfaces**

For user interfaces we create a ui package and in this package we create classes some classes for visualization our model.

**Panel3D –** in this class we create our 3D model. For this we need get all method from our model class, build our 3D camera, build axis, create scene for visualization of 3D model. Also we must to create handle mouse and key mouse for allowing us to use the mouse and keyboard to manipulate the camera's view in the scene. The steps of create 3D model in bottom.

Create the Scene.

Set up the Camera.

Build the Axes.

Build the Model.

Add Camera Viewing Controls.

**Create the Scene** - Create the scene that will hold our Model UI layout. First, copy the contents of Xform.java and save it to a file Xform.java file in the 3D Model source folder of the 3d Model project. This file contains the source code for the Xform sub-class that is derived from the Group class. Using the Xform node prevents the automatic recalculation of the position of a group node's pivot when the children of the group node is changed in a 3D UI layout. The Xform node allows you to add your own types of transforms and rotation. The file contains a translation component, three rotation components, and a scale component. Having the three rotation components is helpful when changing rotation values frequently, such as in changing the angle of the camera in the scene.

**Set up Camera**

Set up the camera in a hierarchy of Group class with Xform instances. Perform the translation and rotation on the camera to change its default location.

1.Add the following lines of code, shown in bold below, so that they appear after the declaration statement for the world object, as shown in Example 8-3.

These lines of code create an instance of a perspectiveCamera and three instances of the public class Xform, which extends the Group class. The Xform class is defined in the Xform.java file you added to your NetBeans project in the previous section of this document.

2.Copy the lines of code for the buildCamera() method, shown in Example 8-4. Add them right after the lines for variable declarations. The buildCamera() method sets the camera to have the view upside down instead of the default JavaFX 2D Y-down. So the scene is viewed as a Y-up (Y-axis pointing up) scene.

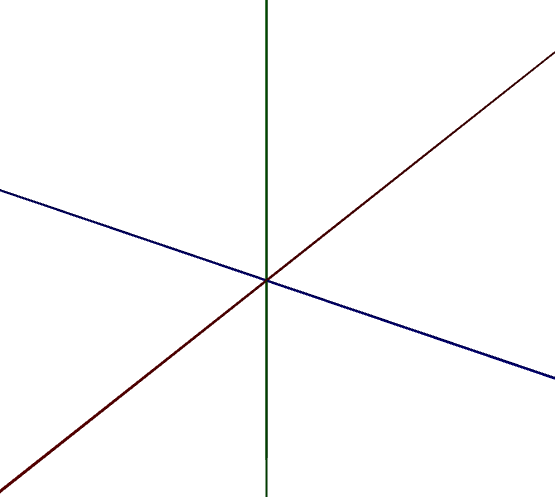
3. In the start() method, add the call to the buildCamera() so that it appears as shown

**Build the Axes**

Add the 3D axes that you will use to build this molecule. The Box class is used to create the axes and the PhongMaterial is used to set the specular and diffused colors. By default in JavaFX, the Y-axis is down.

Per the usual convention, the X-axis is shown in the color red, Y-axis is shown in green, and Z-axis in blue.

Add the following import statements shown in the top of the source file.

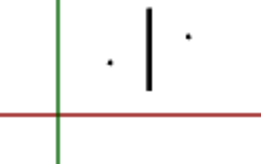


**Build model**

For our model fistful we must to create a node. For creating node we fistful create a box and we give width height and depth (1,1,1) and it will be our model. But when we create our node it create in (0,0,0) coordinates. After creating node in (0,0,0) coordinates we must to translate this node where we want.

For creating a bar it will be a little bit different. To illustrate what’s going on, I created a 2D representation of all individual steps, starting off with the creation of the cylinder. By default, its center is placed at the origin:



The moveToMidpoint transformation moves its center to the final location (the yellow midpoint in the above image):

And finally, the rotateAroundCenter transformation corrects the cylinder’s direction:

Obviously, the tricky part is finding the proper axis and angle of rotation. Both are calculated relative to the yAxis because of the cylinder’s initial direction. The axisOfRotation must be perpendicular to the plane defined by the yAxis and the diff vector. To find such a perpendicular vector, we can (by definition) calculate the cross product of the two vectors. In the 2D case, the result is equivalent to the z-axis, but in 3D it may be tilted.

The angle is calculated based on the dot product (aka scalar product) of the two vectors, which is defined by

A • B = |A| |B| cos(alpha)

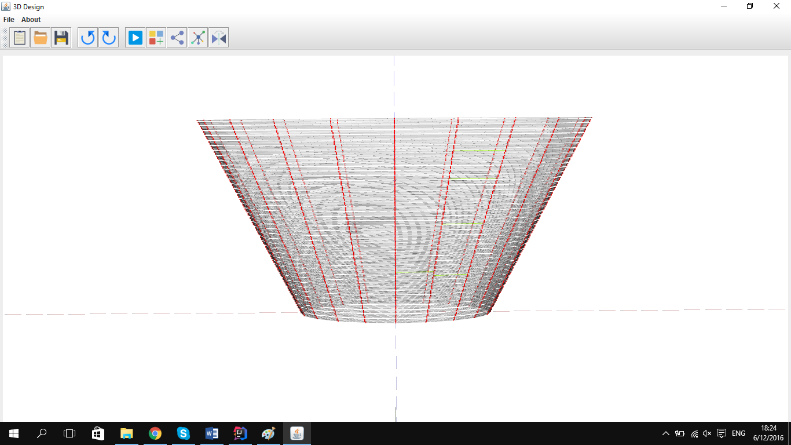
If A and B are both unit vectors, this becomes

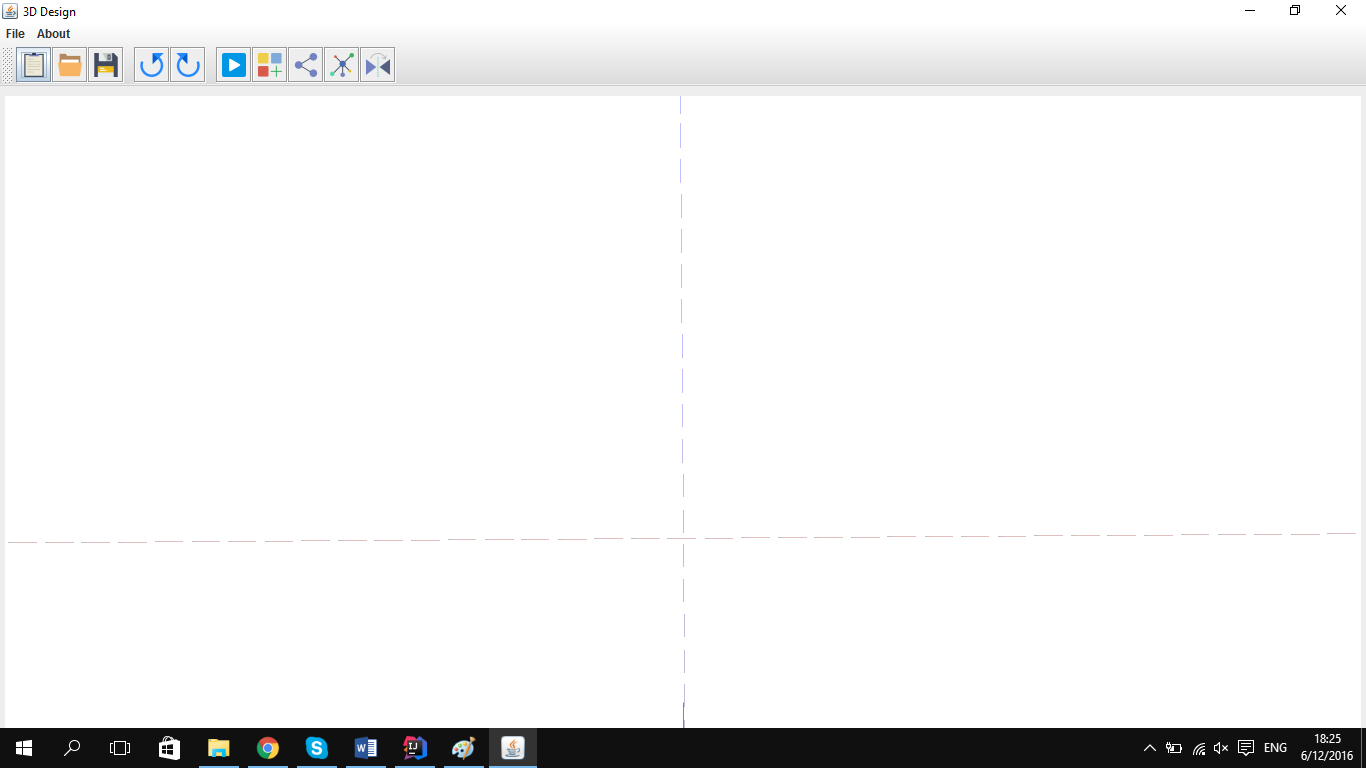
A • B = cos(alpha)

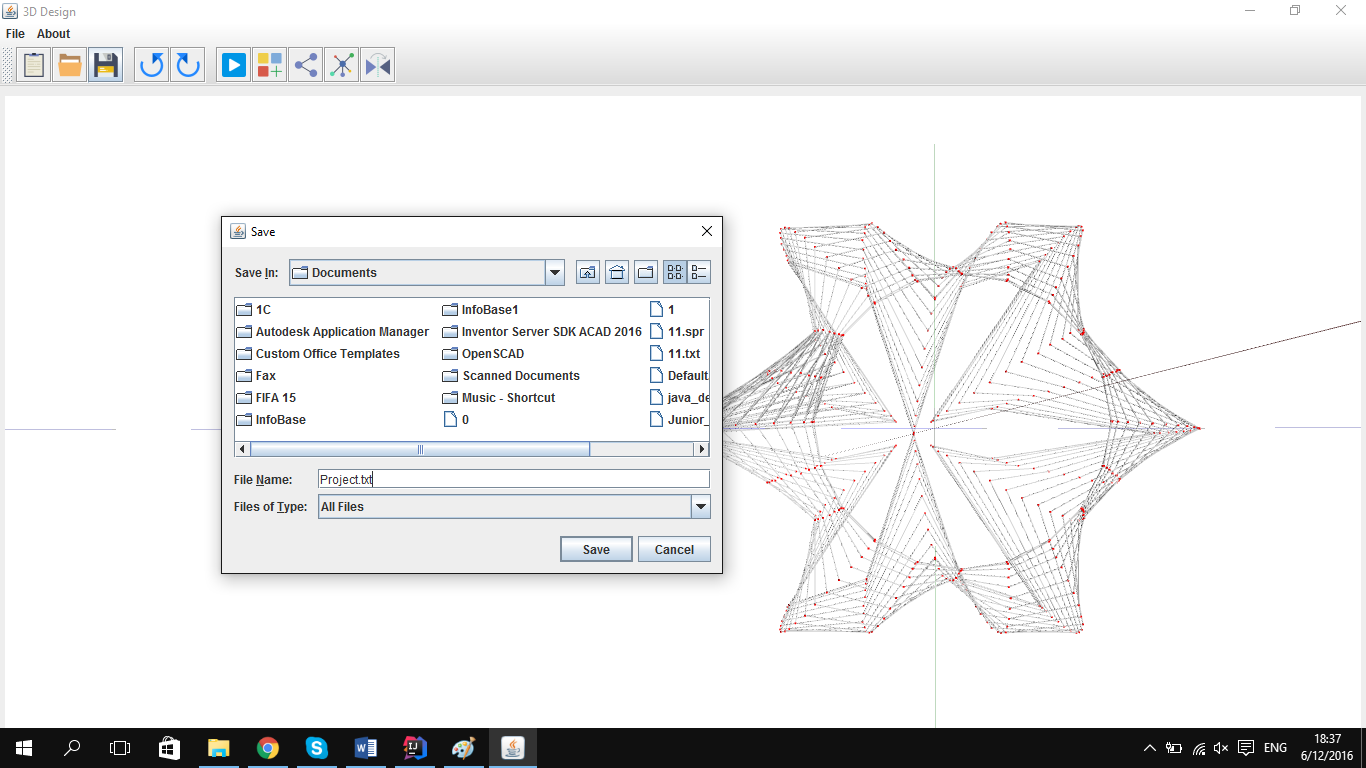
which we can resolve to

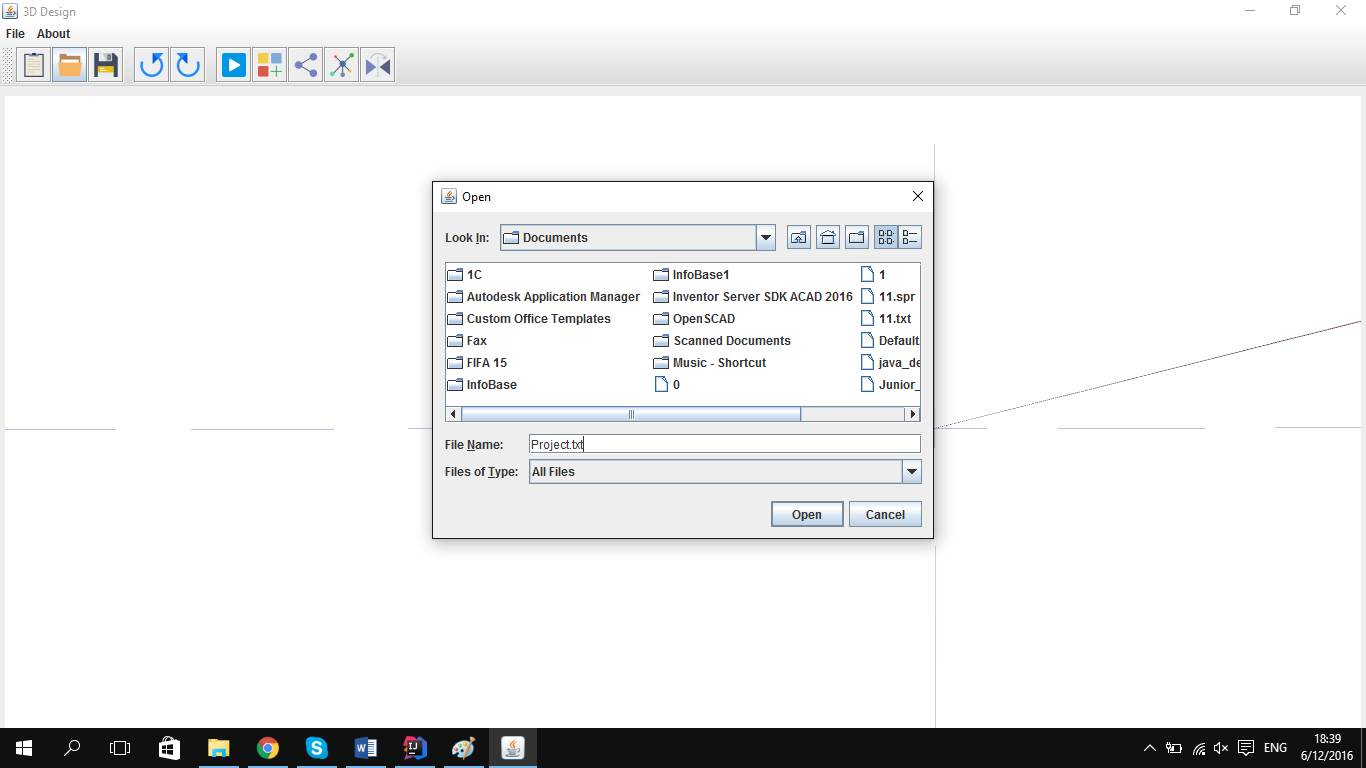
alpha = acos(A • B)

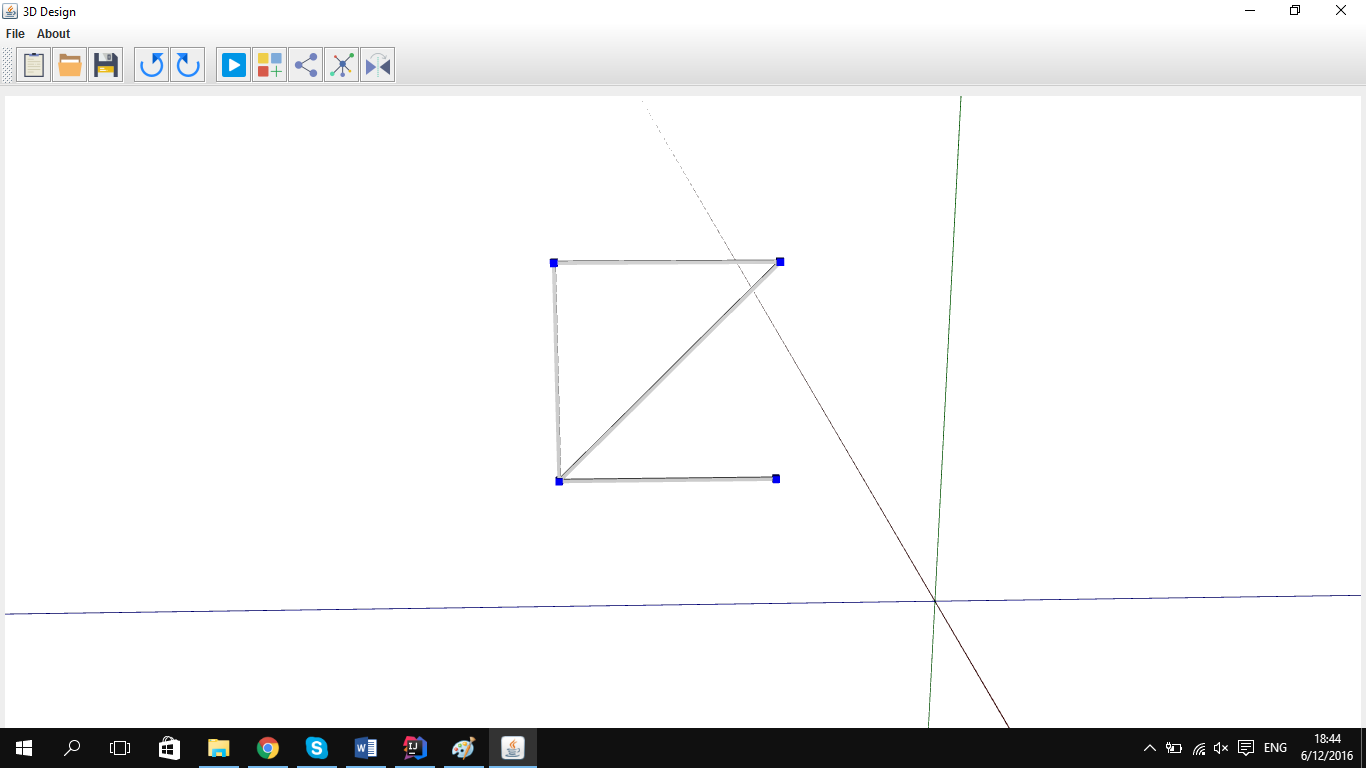
**Frame Design** In this class we are creating User Interface. We are creating buttons for new, open, save, redo, undo, node, bar, truss.

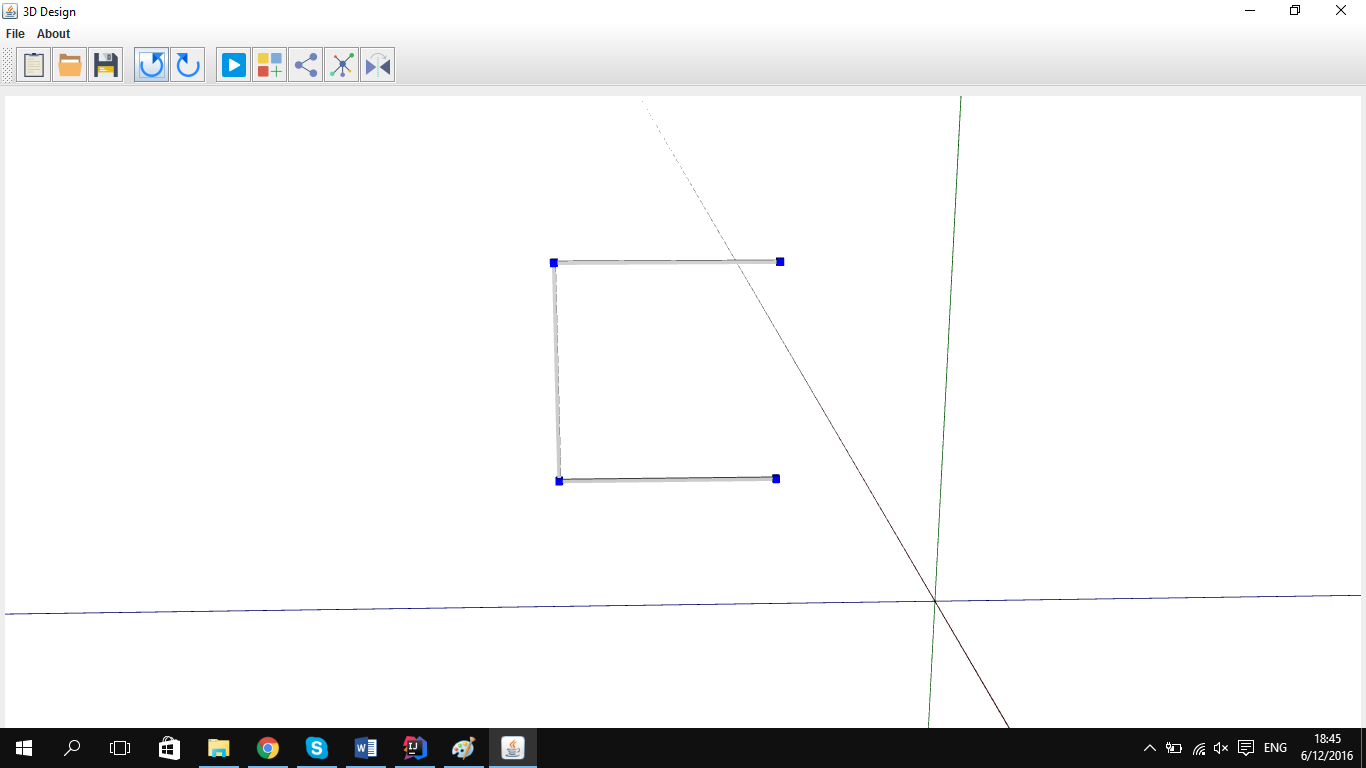
New – If we have same model in our program and we want to create a new one for this we don’t need close our program and run again.

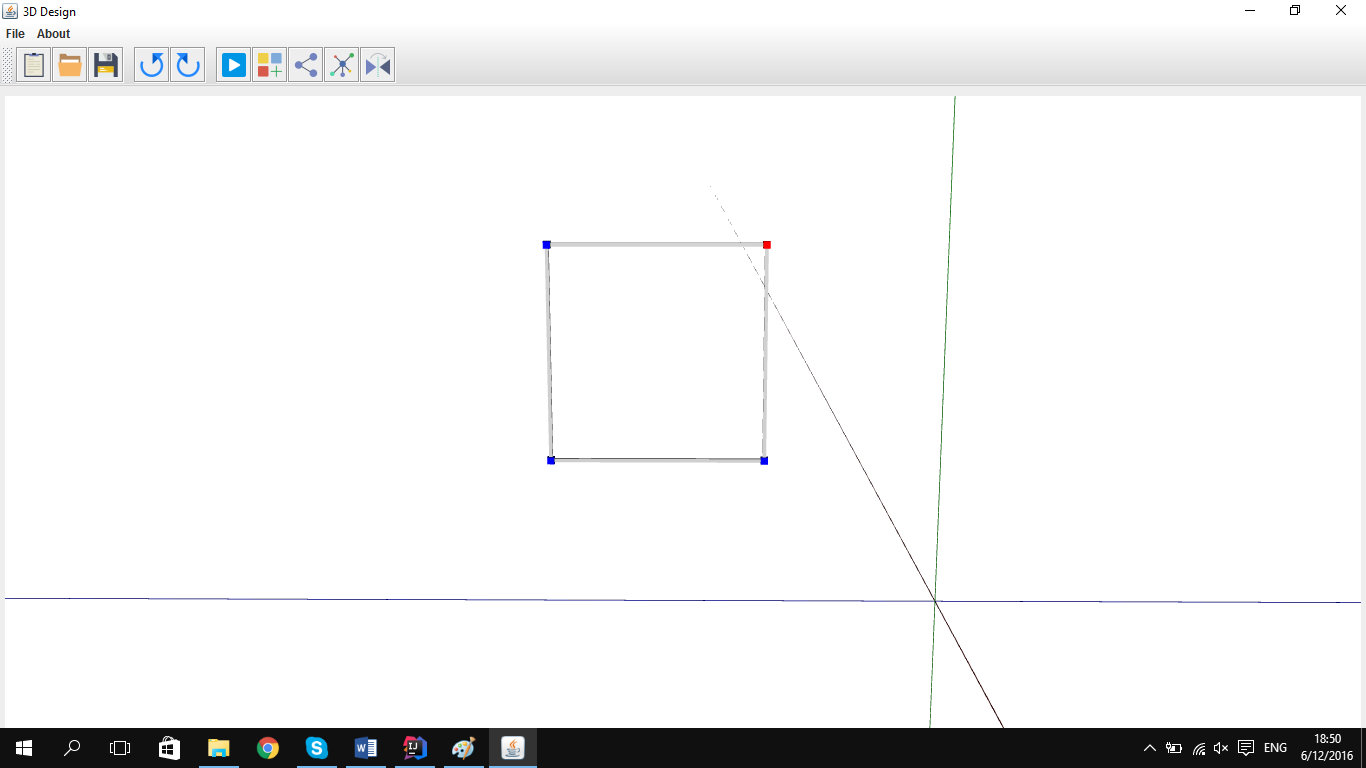
In this case we are using new Button. It will help us to create a new page and we can create our new model in there. For this we only need to push new Button. When we push new button it create a new page for creating new 3D model.

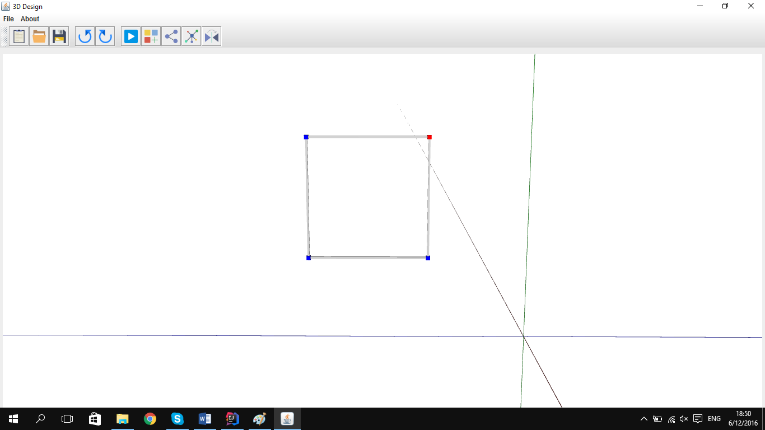
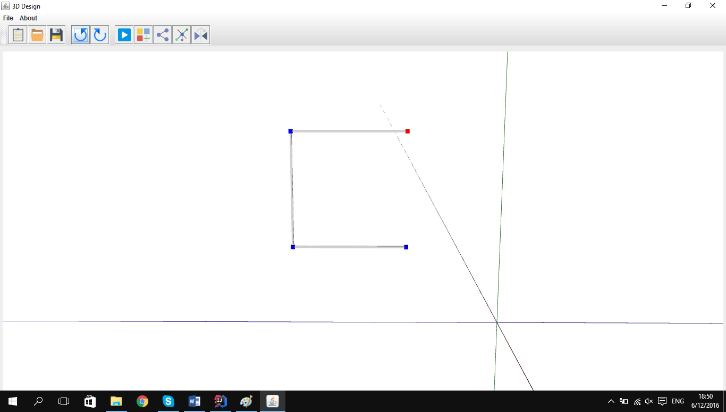
Save – This buttons help us to save our project. By helping this button we can easily save our project in our computer and others time we can open it and work on it.

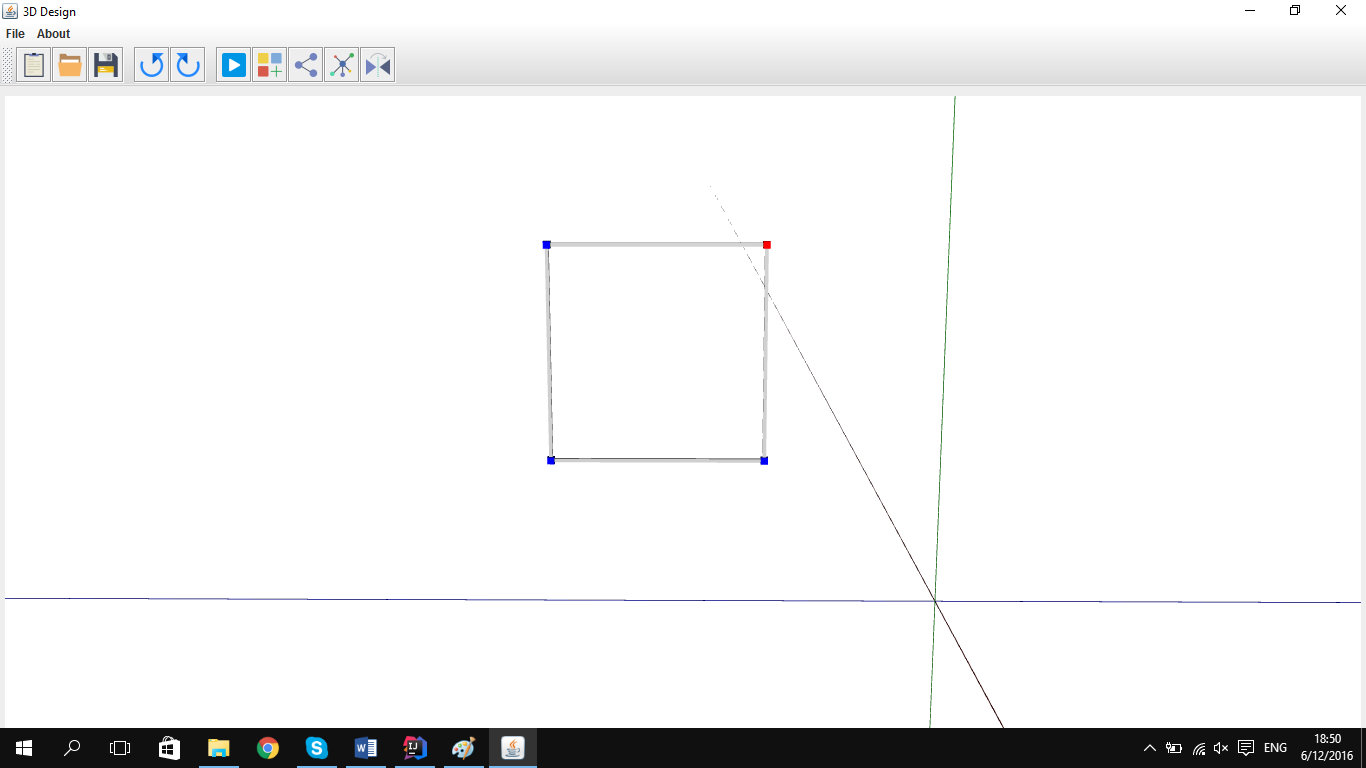
Open - This button help us to open our project which we save before and work on it.

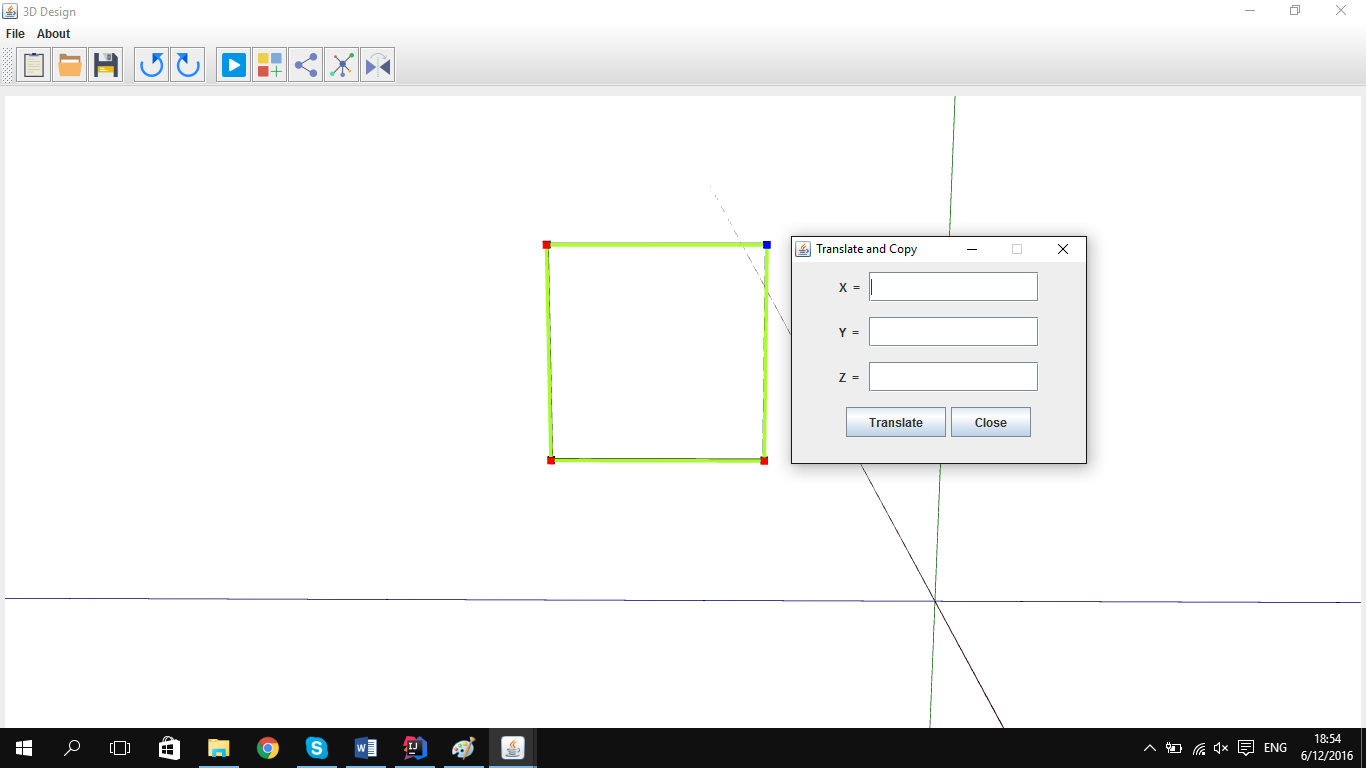
Undo - we are given this button all specification of Undo command and when we push this button it will go one or more step back. For example if we create a wrong line when we create a square by helping this button we can go back and create connect a line correctly.



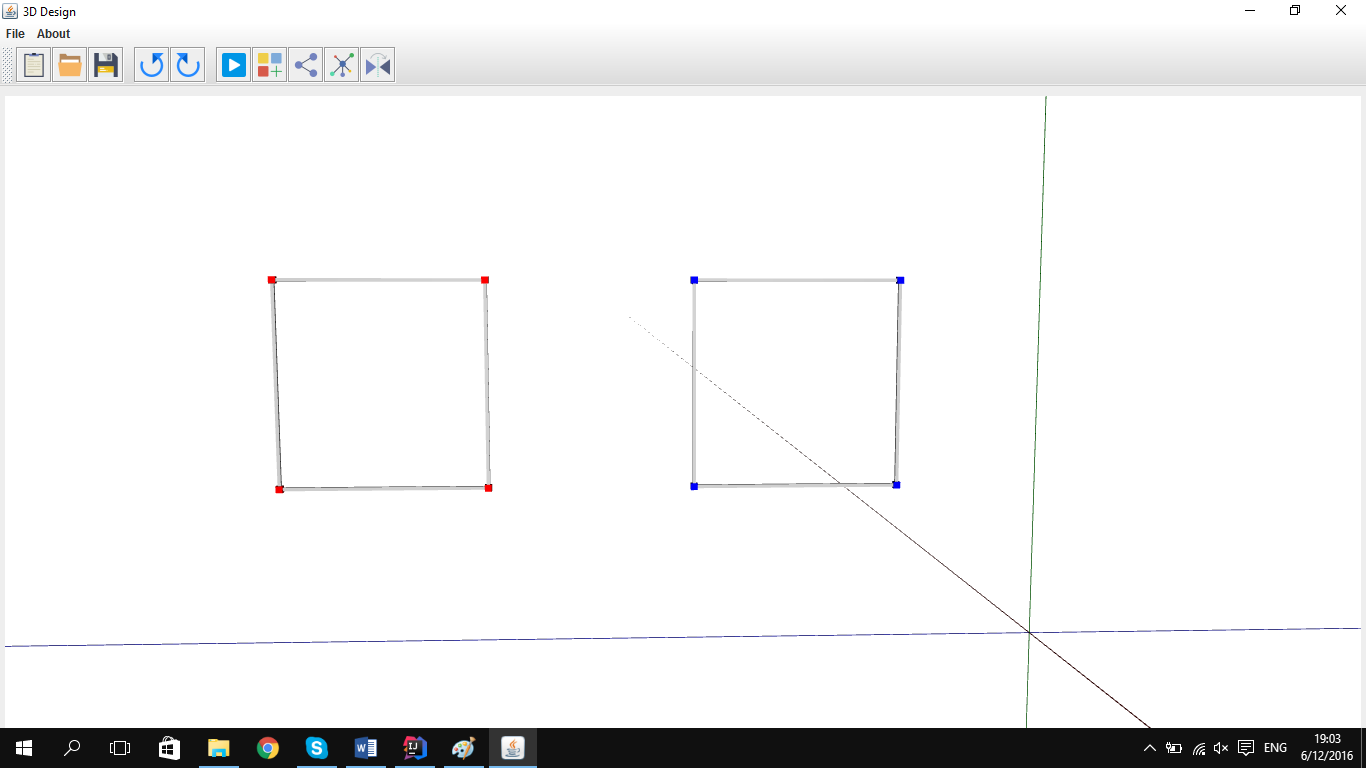
Redo - we are given this button all specification of Redo command and if we push undo before this button we can get back where we push undo buttons. It always work after Undo buttons. If we don’t push our Undo buttons this button will not work. If we incorrectly push our Undo button by helping this button we can correct it.



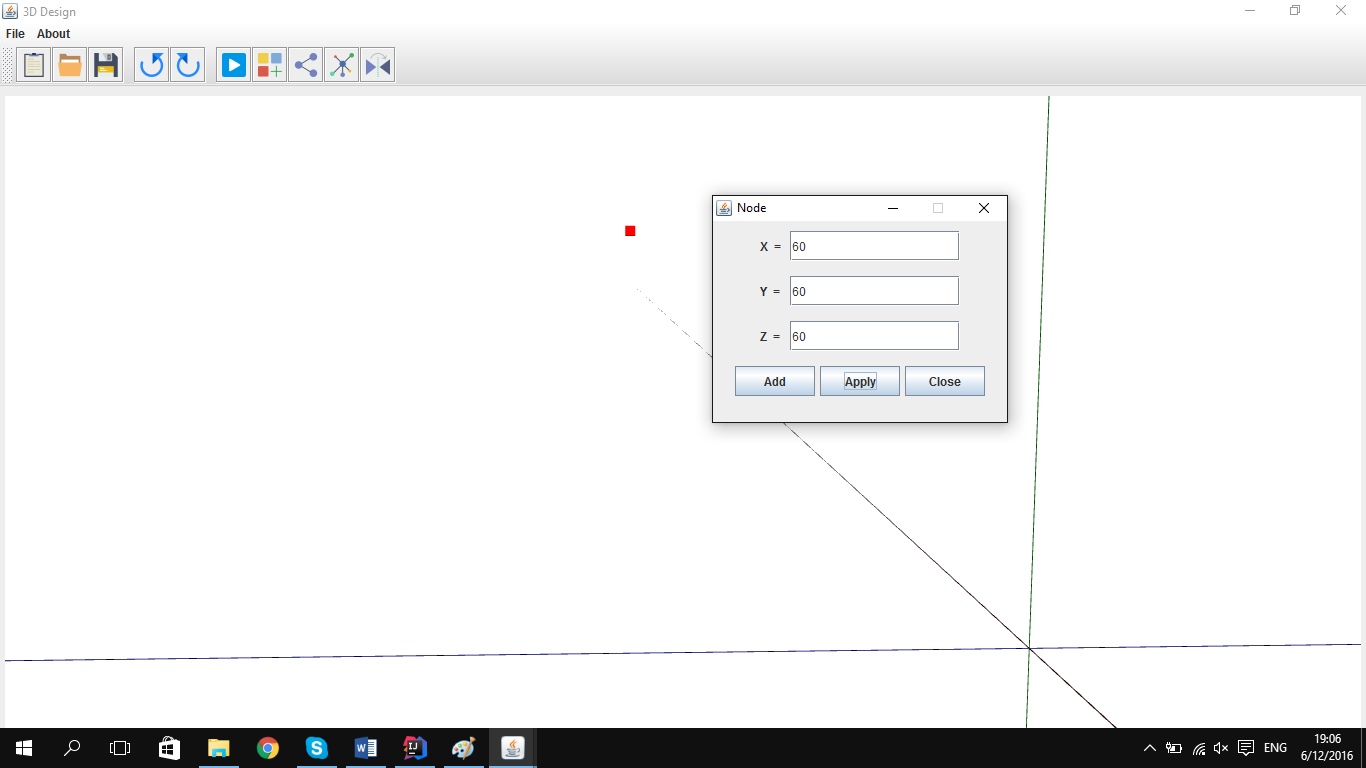
Translate – After selecting our node, bar or our model we are pushing this button. When we push this button we will see a new window with x, y, z coordinate. By giving x, y, z coordinates we can copy our model and translate that place. For example if we create a model

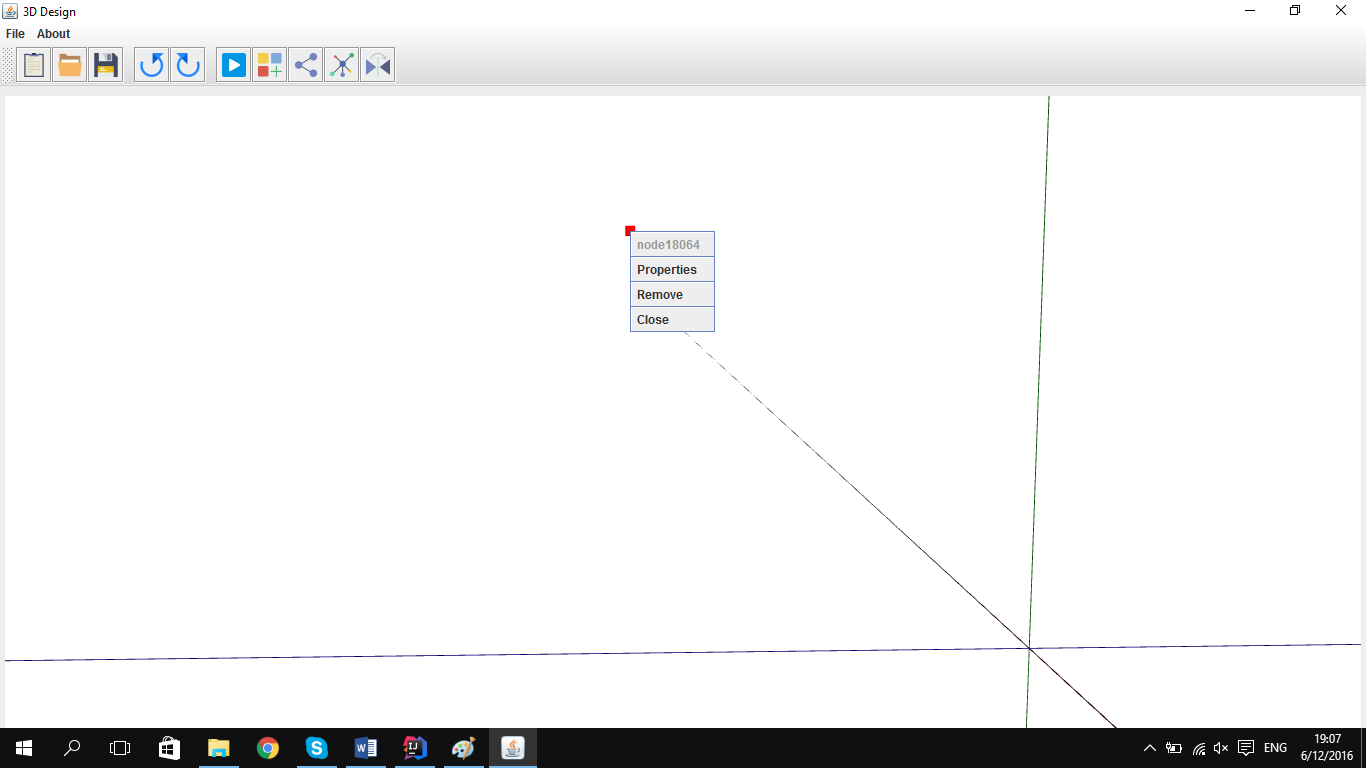
****If we want to copy this square and create new one like this we must select all nodes and bars then click the translate buttons.

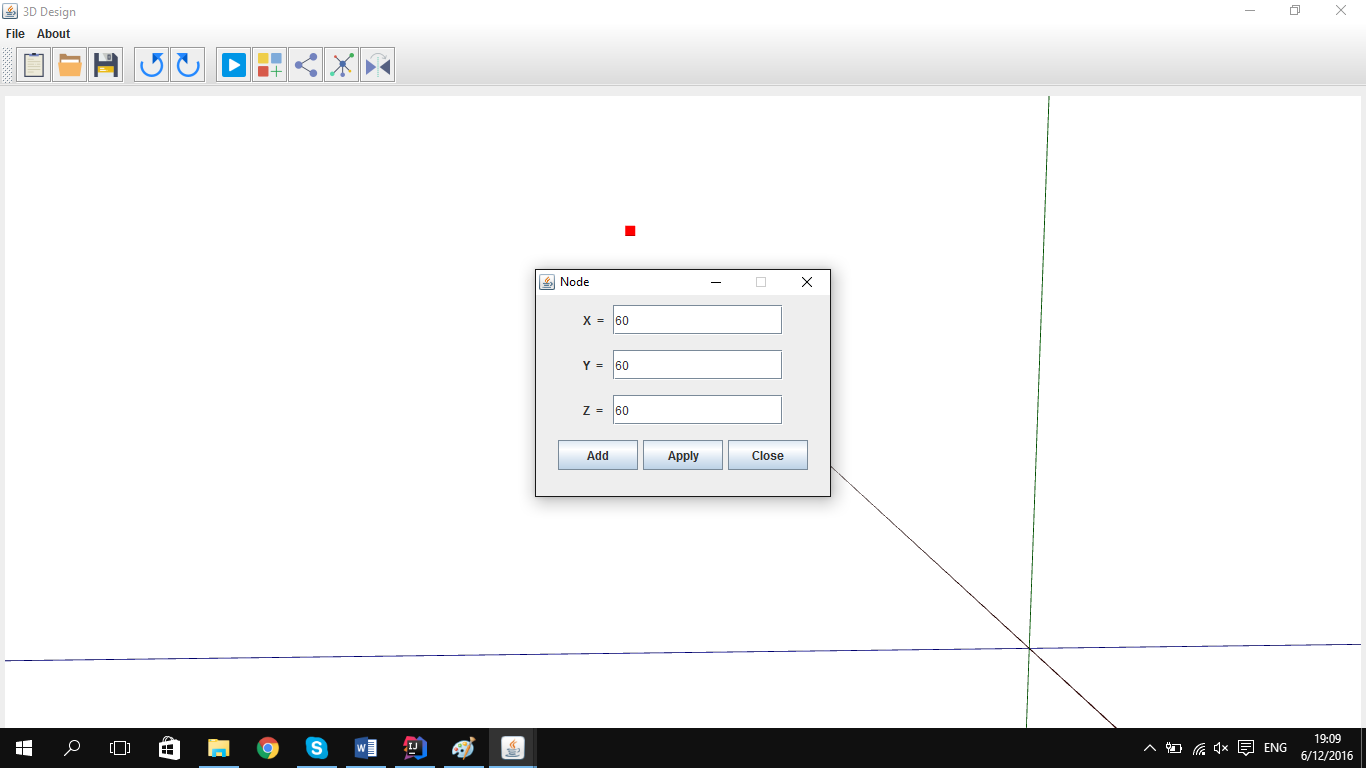
Then we must to put our new coordinates and click Translate button

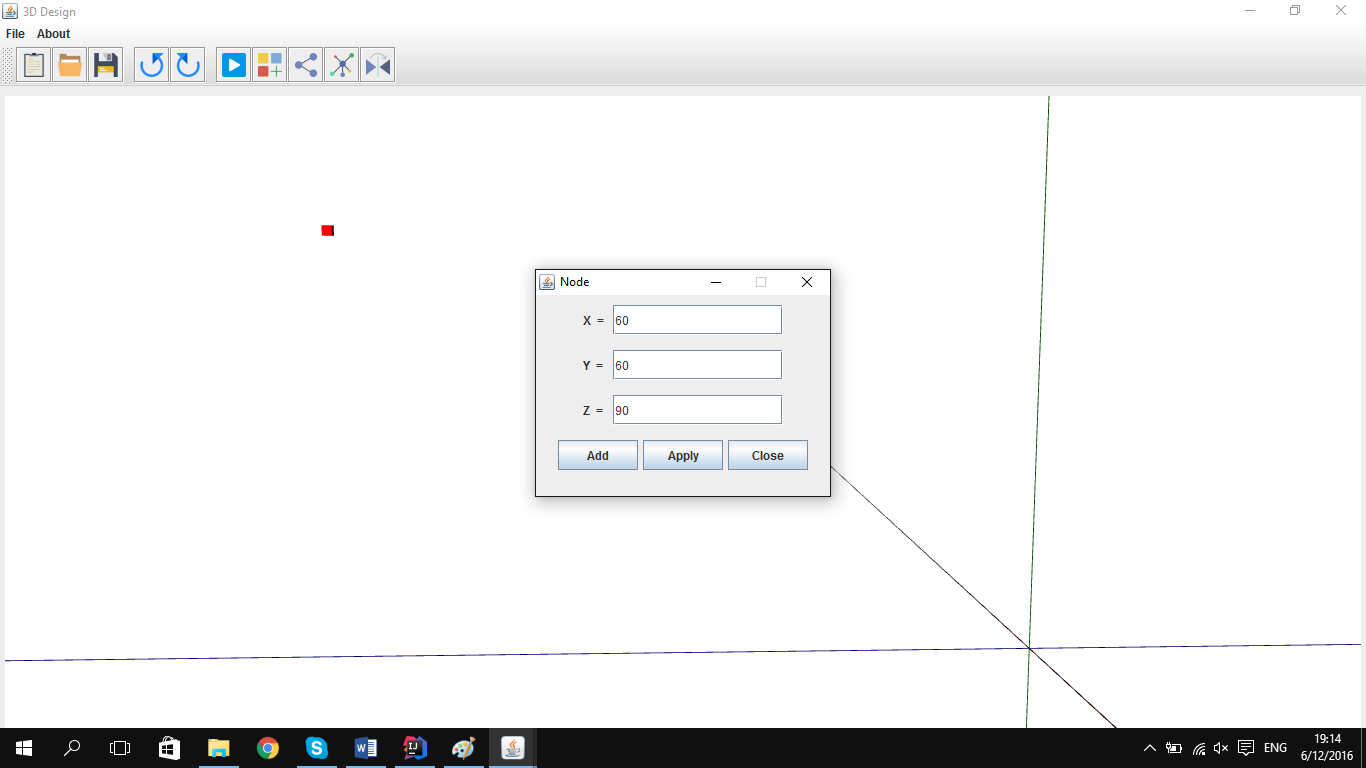


**CreateNode** – When we creating node window we used JLabel (labelX, labelY and labelZ) for getting x,y,z coordinates from our model and JButtons (ADD, APPLY,ClOSE) for showing this coordinates visually in our model 3D.When we push Node button on our interface it open Node window. After putting our coordinates if we push ADD button it will create a node and will close the window. But when we push APPLY button and it will create a node withouting closing window and we can create another nodes too. When we

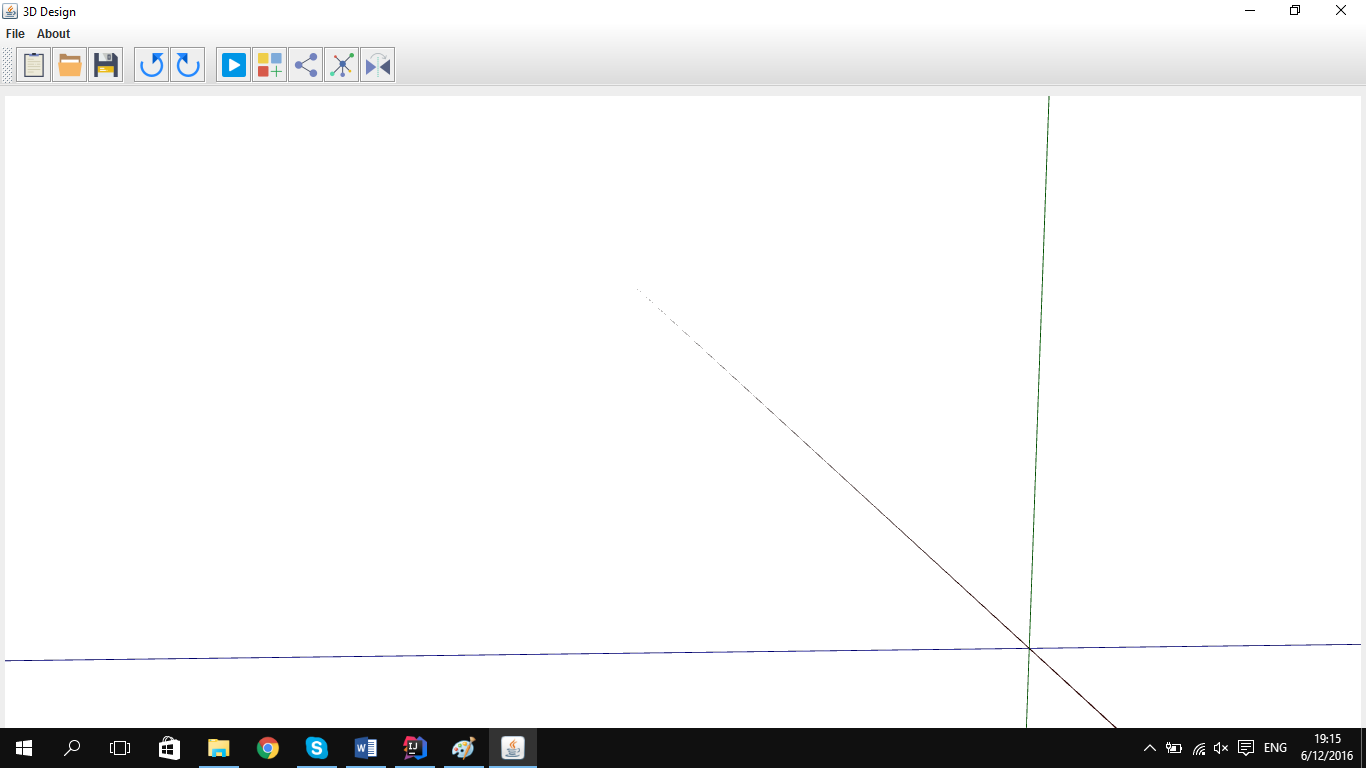
****push CLOSE button it will close window withouting creating a node

**NodeProperites –** For getting nodeProperites vindow we must create a node after this we must to select this node and by pushin right click we can see nodePorperites window. By this window we can see node Prperites window. In this window we can see the name of this node .Properites Remove and close window.If we push Properites button we can see Node window .By this window we can translate this node.if we push Remove button our node will deleted. If we push Close button our window will closed without doing anything

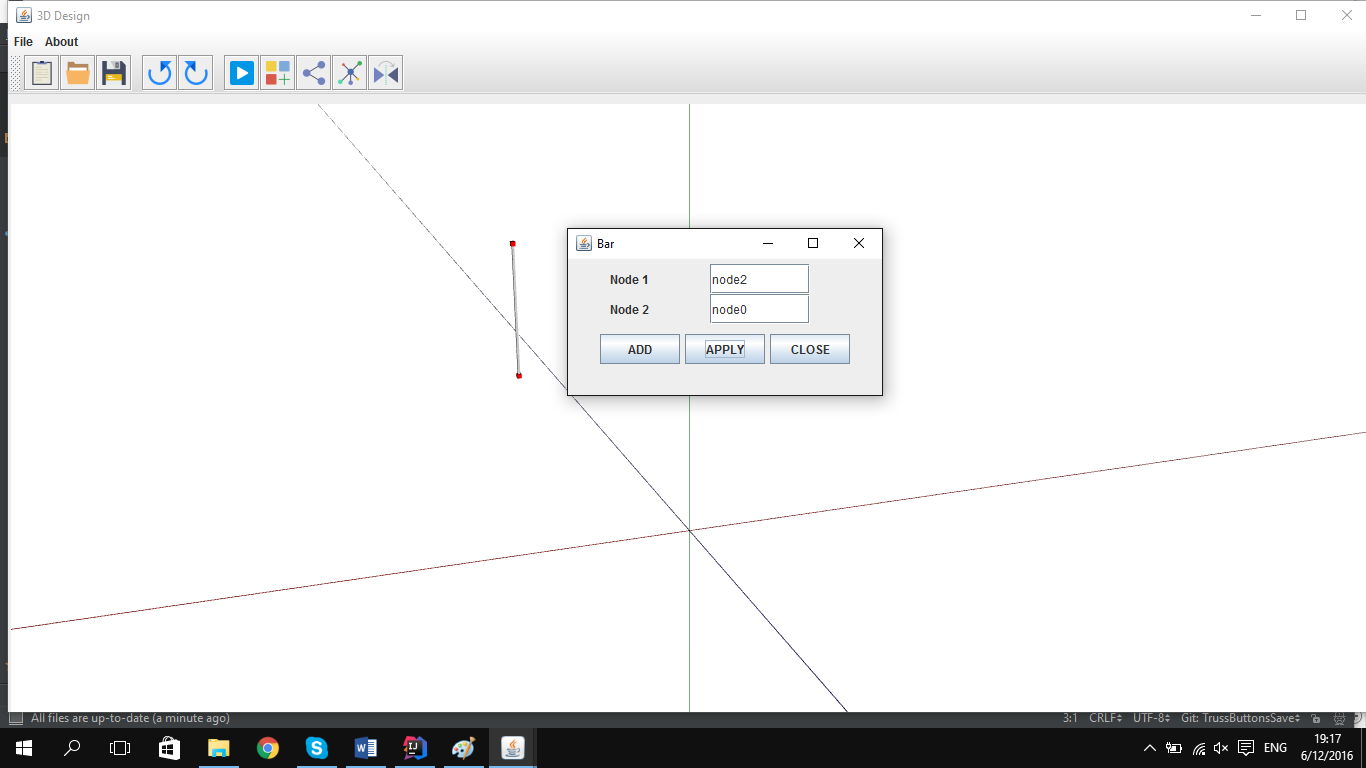
If we push Properties it will open window by x, y, z coordinates of this node.

Changing this coordinates we can change our nodes position.

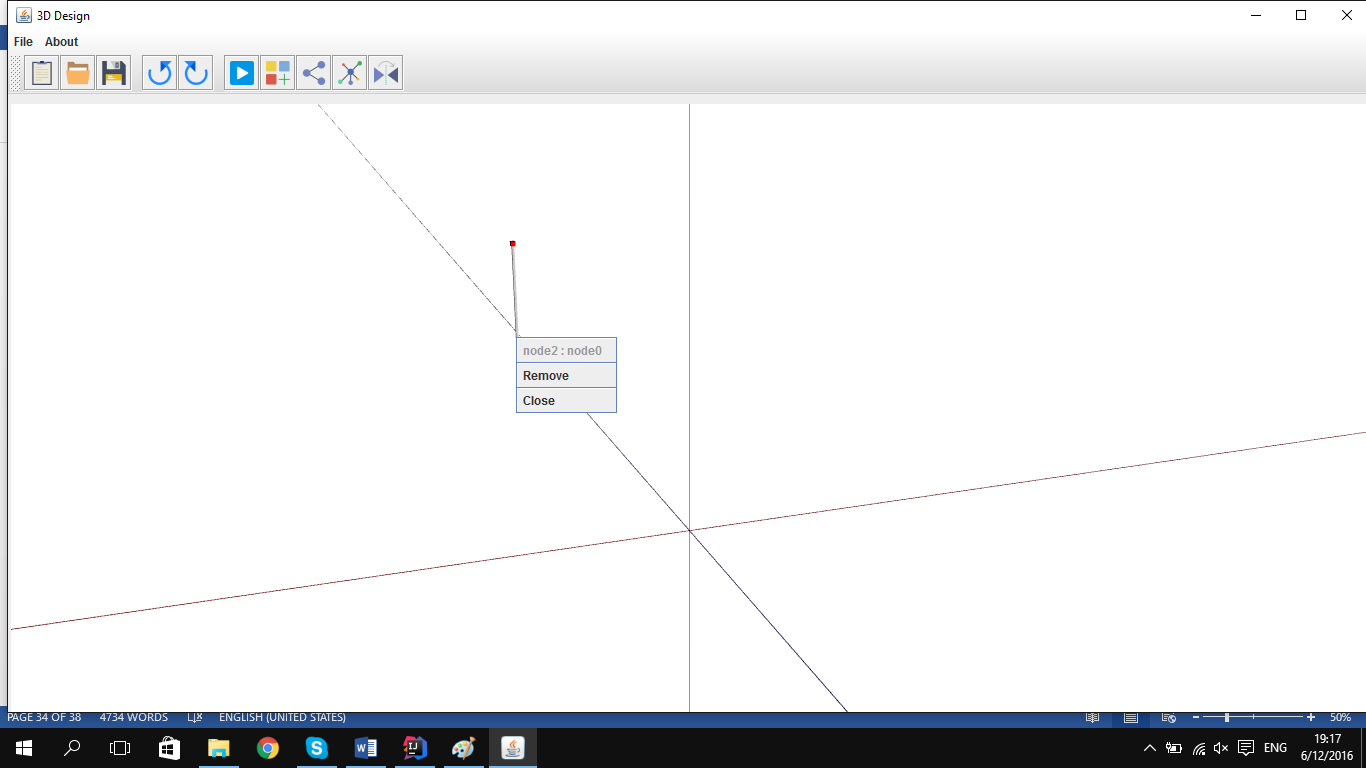
If we push Remove button it will remove our node from our model.

****

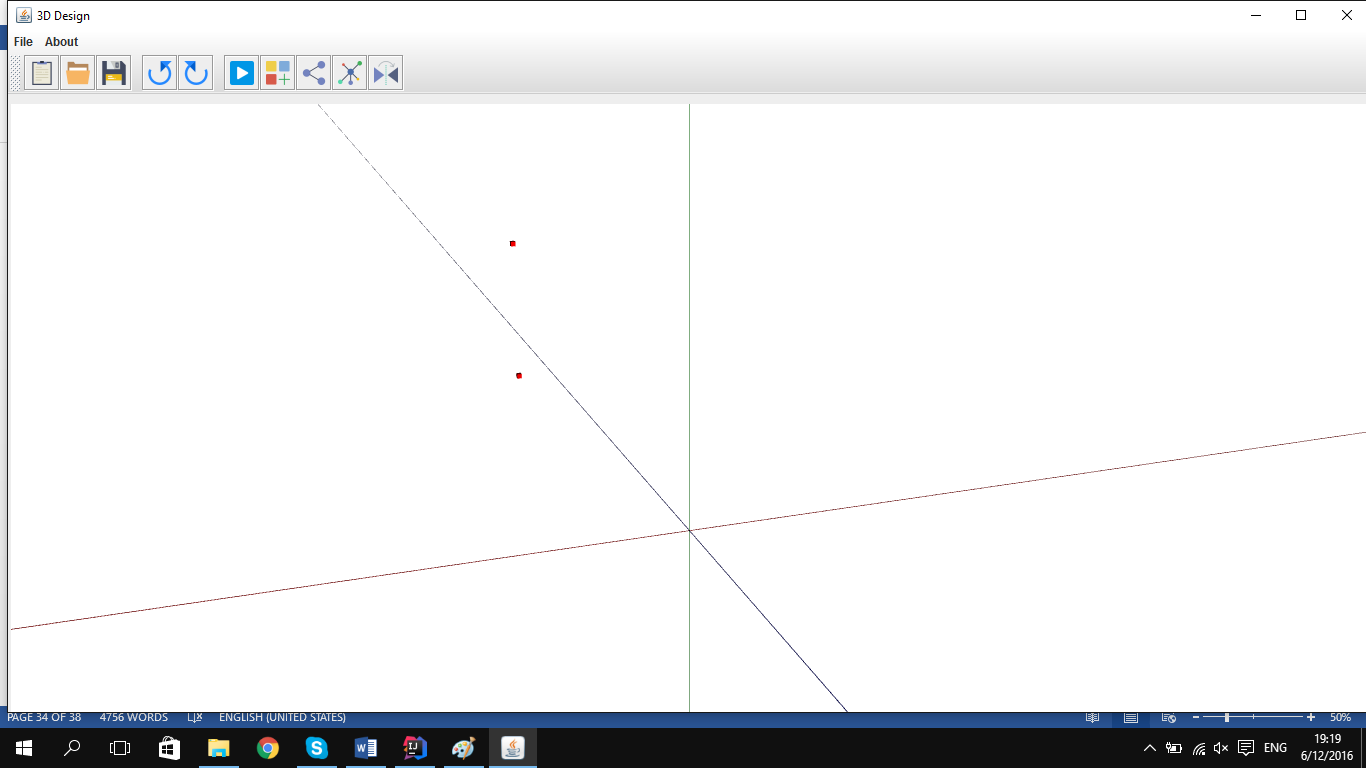
**CreateBar –** We will create bar by selecting start node and end node. For this we create only two label for getting start node and end node froma our model. We will get it by selecting our node in our 3D model and we also create JButtons (ADD, APPLY,ClOSE) for creating our bar. For seeing this window in our model we can push Bar button. For creating The bar fistful we must create at least two node. The we must to select this node and we will see this node in our Bar label.If we push Add Button it will create a bar and will closed window but if we push Applay button it will create a bar and after this we can select another nodes and do this process again.If we push close button it will closed without doing any process.

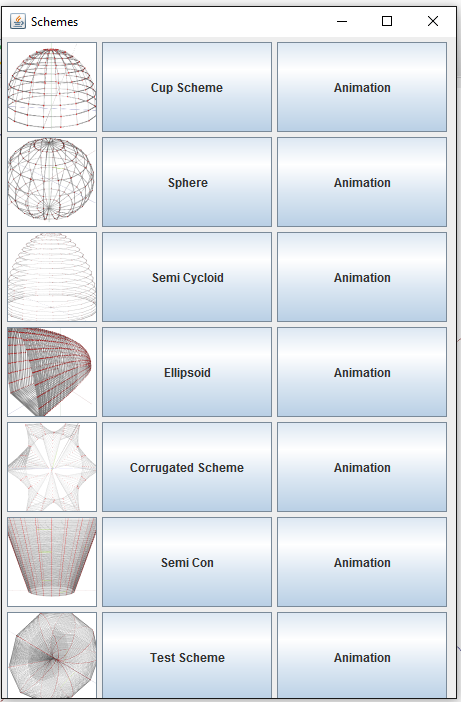


**BarProperites – –** For getting BarProperites vindow we must create a bar after this we must to select this node and by pushin right click we can see barPorperites window. Doing this we can see node Prperites window. In this window we can see the name of this bar.Name of this bar will be start node add End Node . Remove and close window. If we push Remove button our node will deleted. If we push Close button our window will closed without doing anything



If we will push Remove button it will remove our bar from our model. But our nodes will be stay in there.



**CreateTruss-** For getting Truss window we must push truss button on our Frame.In our Truss window we have 3 type of buttons. First button Will show us the pictures of our schemes. The second button will show us the name of schemes and will create it. Third type of button will show us the type of schemes and will create it by animation. We will get our truss from our scheme package. Scheme package is into the model package. Ecery Schemes have ther ovn class.

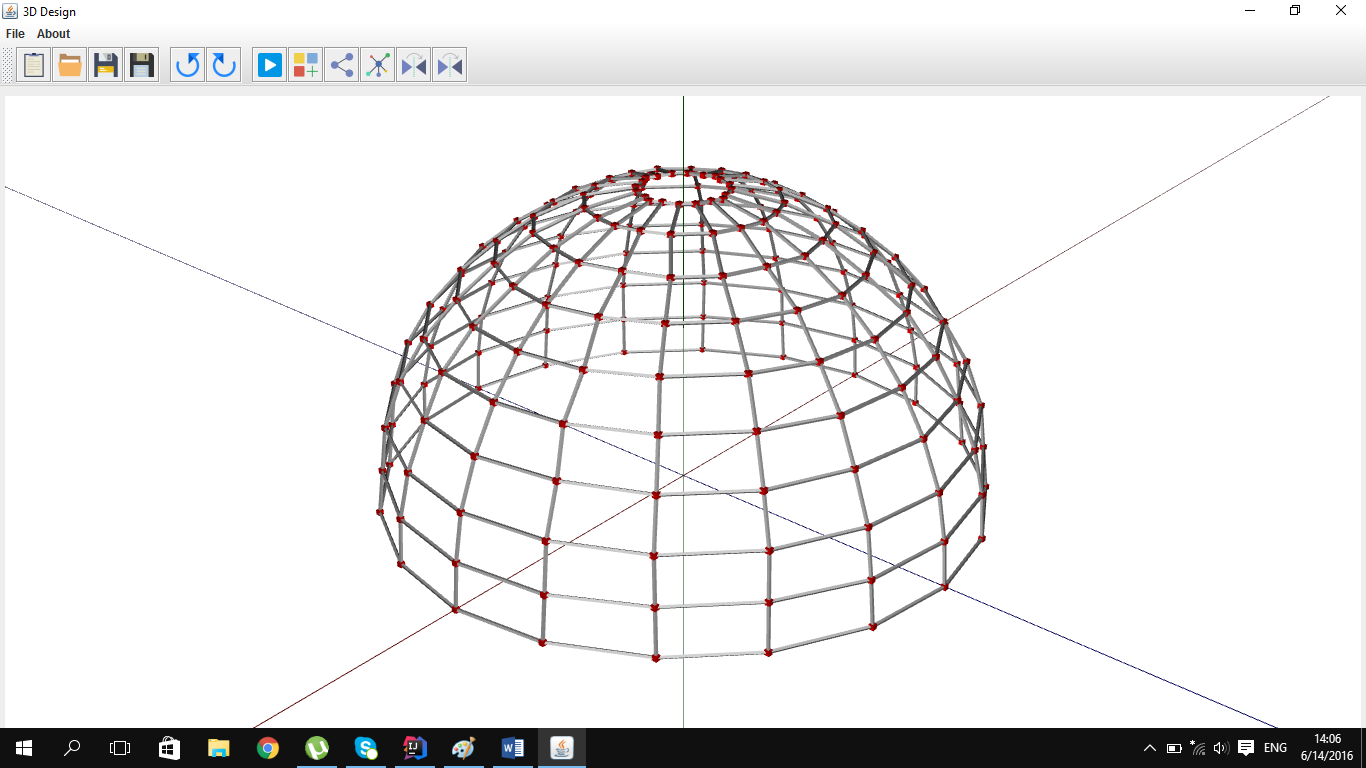
If we push the name of the Cup Scheme button it will create a beautiful Cup schemes. For creating Cup scheme we are create in package model a package scheme and in this scheme we create a class Cup Scheme. In this class we create 2 method. First method generate method. It will create our model directly. For creating our model we create some variables in our method. We create our model by x, y and z equation. In this Scheme x, y, z will be

x= (2 \* cos (180\*t)\*cos (180\*s)

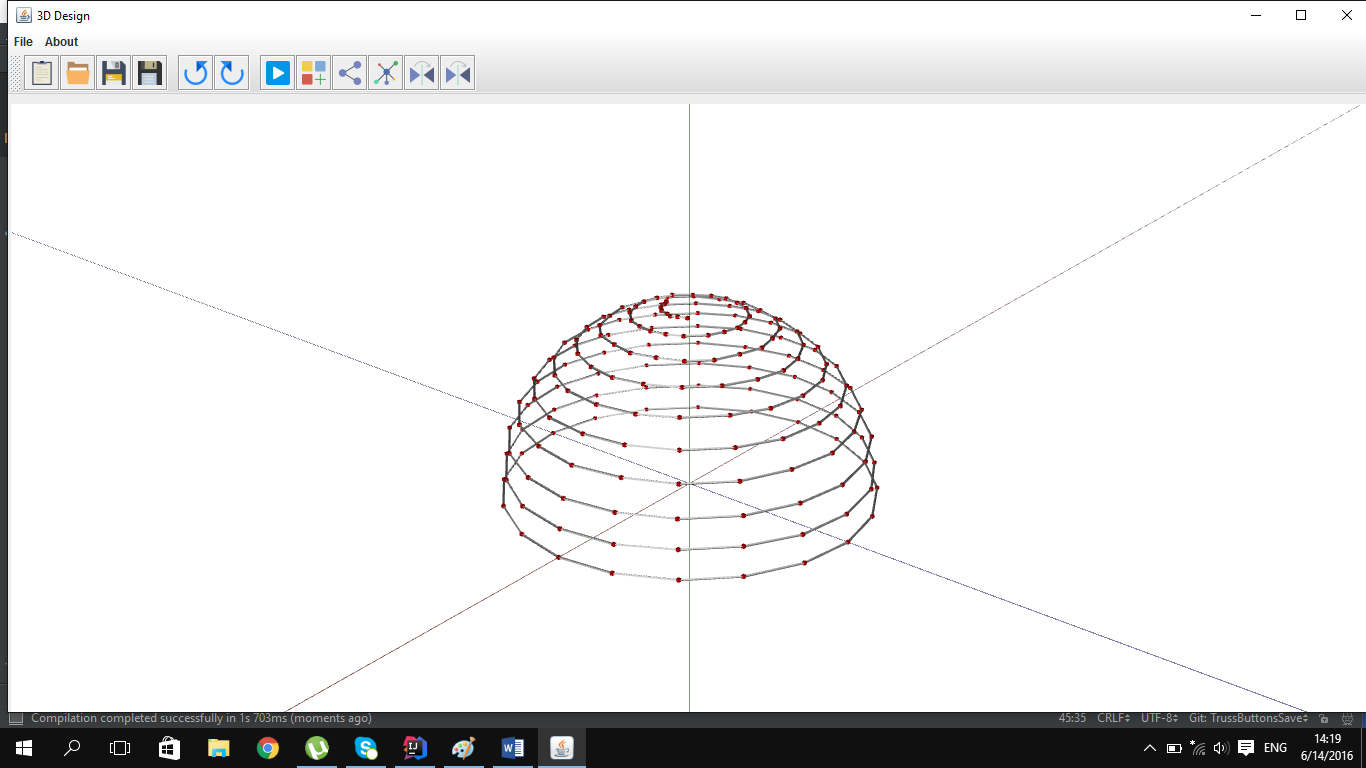
y= (2\*cos (180\*t)\*sin (180\*s)

z= (2\*sin (180\*s)

t is crating nodes and bars in y and z direction

s is creating nodes and bars in y and x direction

Also we can create it by animation for this we must push Animation button and it will create our model step by step. In this method we can see every step of our nodes and models creating. For creating this animation we are create in our Cup Scheme class an action method which will create our model by animation.



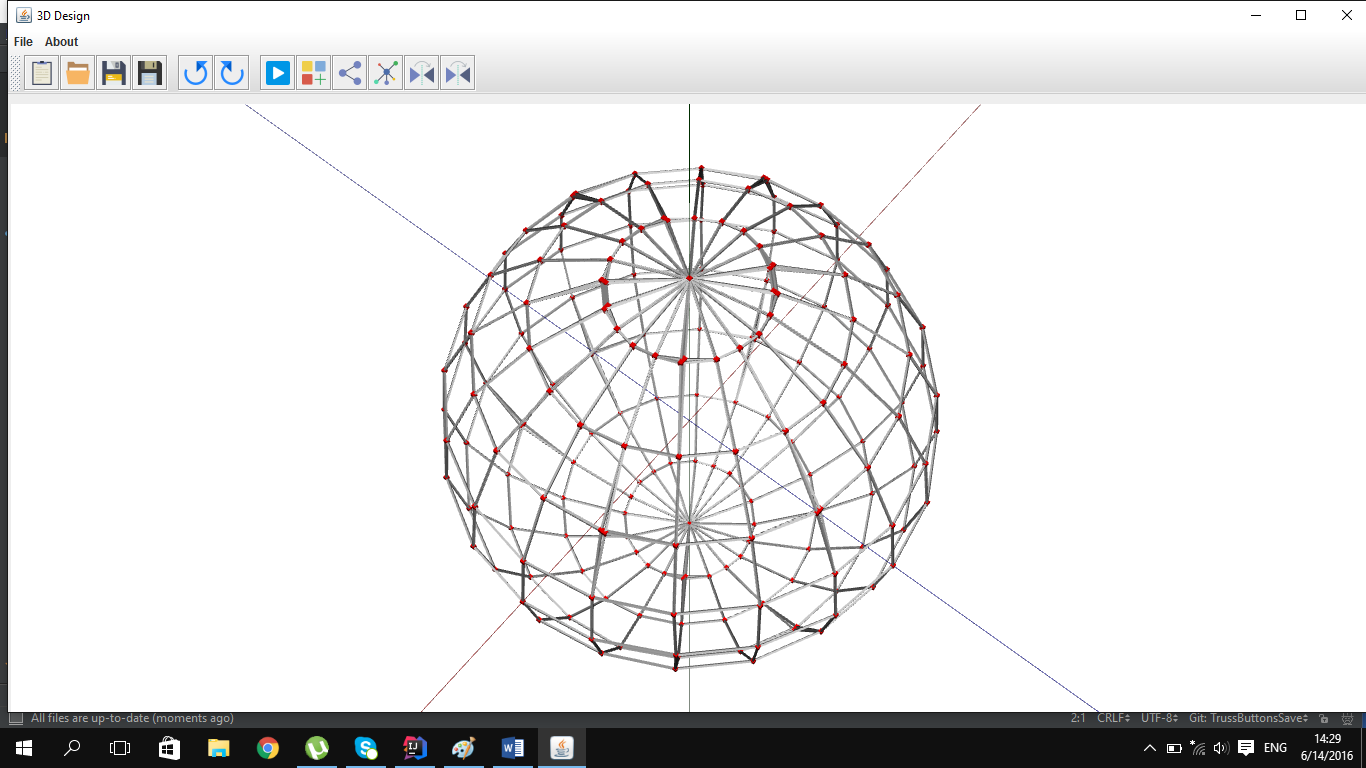
If we push the name of the Sphere Scheme button it will create a beautiful Sphere schemes. For creating Sphere scheme we are create in package model a package scheme and in this scheme we create a class Sphere Scheme. In this class we create 2 method. First method generate method. It will create our model directly. For creating our model we create some variables in our method. We create our model by x, y and z equation. In this Scheme x, y, z will be

x= (2 \* cos (360\*t)\*cos (360\*s)

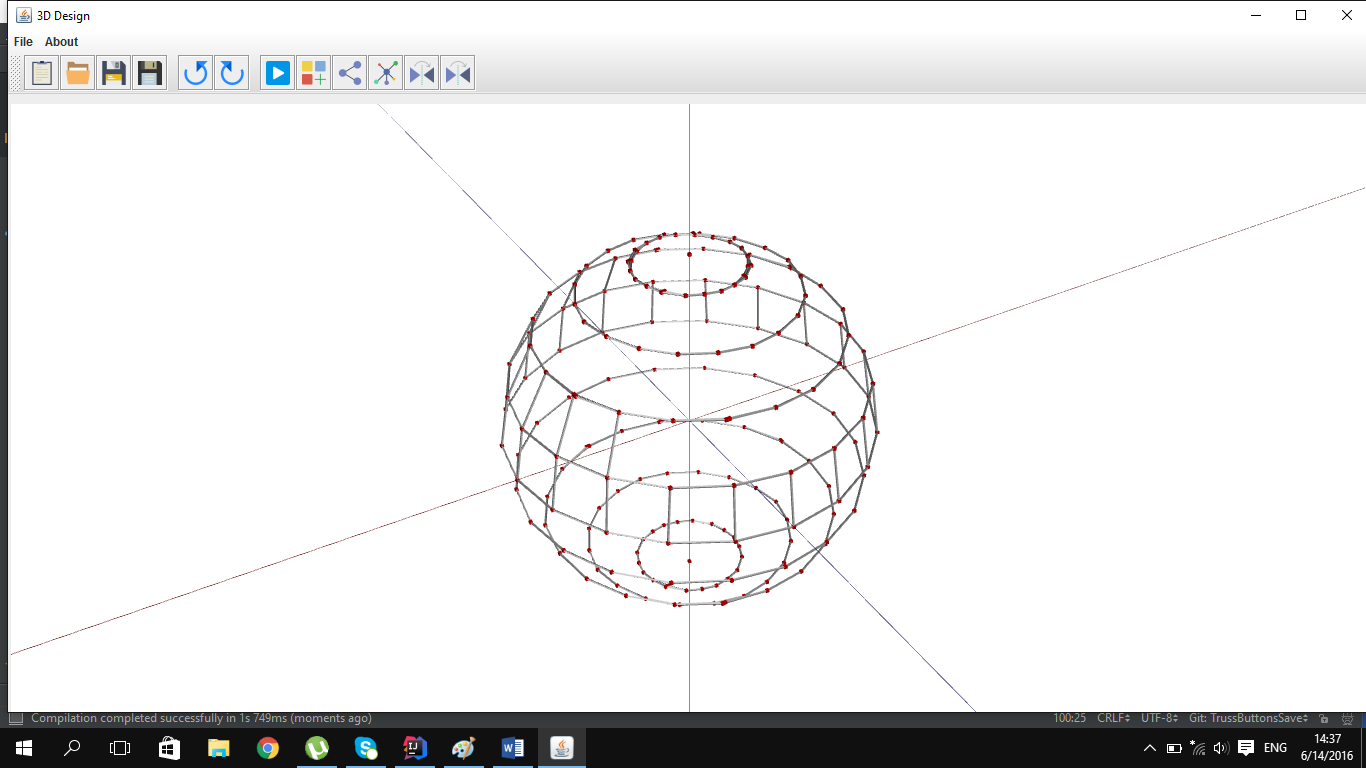
y= (2\*cos (360\*t)\*sin (360\*s)

z= (2\*sin (360\*s)

t is crating nodes and bars in y and z direction

s is creating nodes and bars in y and x direction

Also we can create it by animation for this we must push Animation button and it will create our model step by step. In this method we can see every step of our nodes and models creating. For creating this animation we are create in our Sphere Scheme class an action method which will create our model by animation.



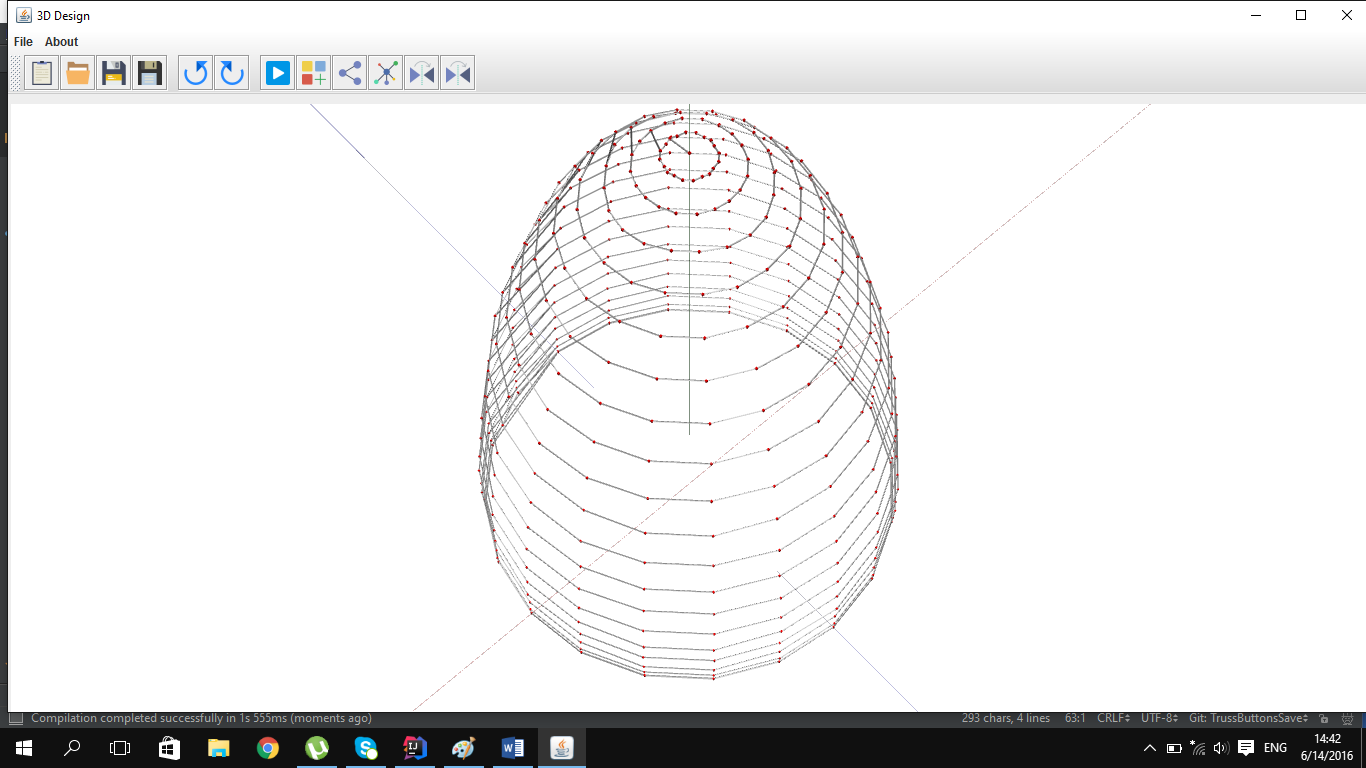
If we push the name of the Semi Cycloid Scheme button it will create a beautiful Sphere schemes. For creating Semi Cycloid scheme we are create in package model a package scheme and in this scheme we create a class Semi Cycloid Scheme. In this class we create 2 method. First method generate method. It will create our model directly. For creating our model we create some variables in our method. We create our model by x, y and z equation. In this Scheme x, y, z will be

x= (2 + (Pi + t +sin (Pi + t)) +sin (2 + Pi + s))

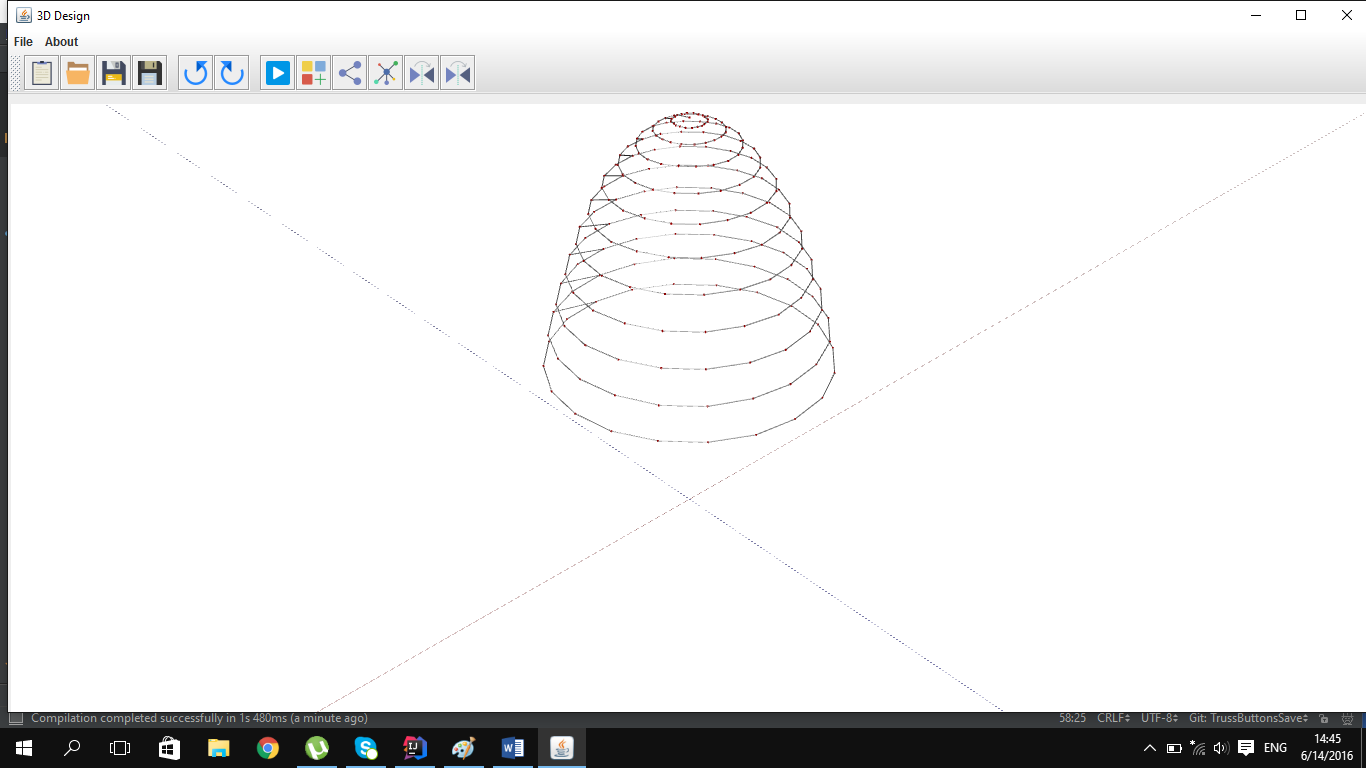
y= (8 + (1 + cos (Pi + t)

z= (2 + (Pi + t +sin (Pi + t)) +cos (2 + Pi + s))

t is crating nodes and bars in y and z direction

s is creating nodes and bars in y and x direction

Also we can create it by animation for this we must push Animation button and it will create our model step by step. In this method we can see every step of our nodes and models creating. For creating this animation we are create in our Semi Cycloid Scheme class an action method which will create our model by animation.



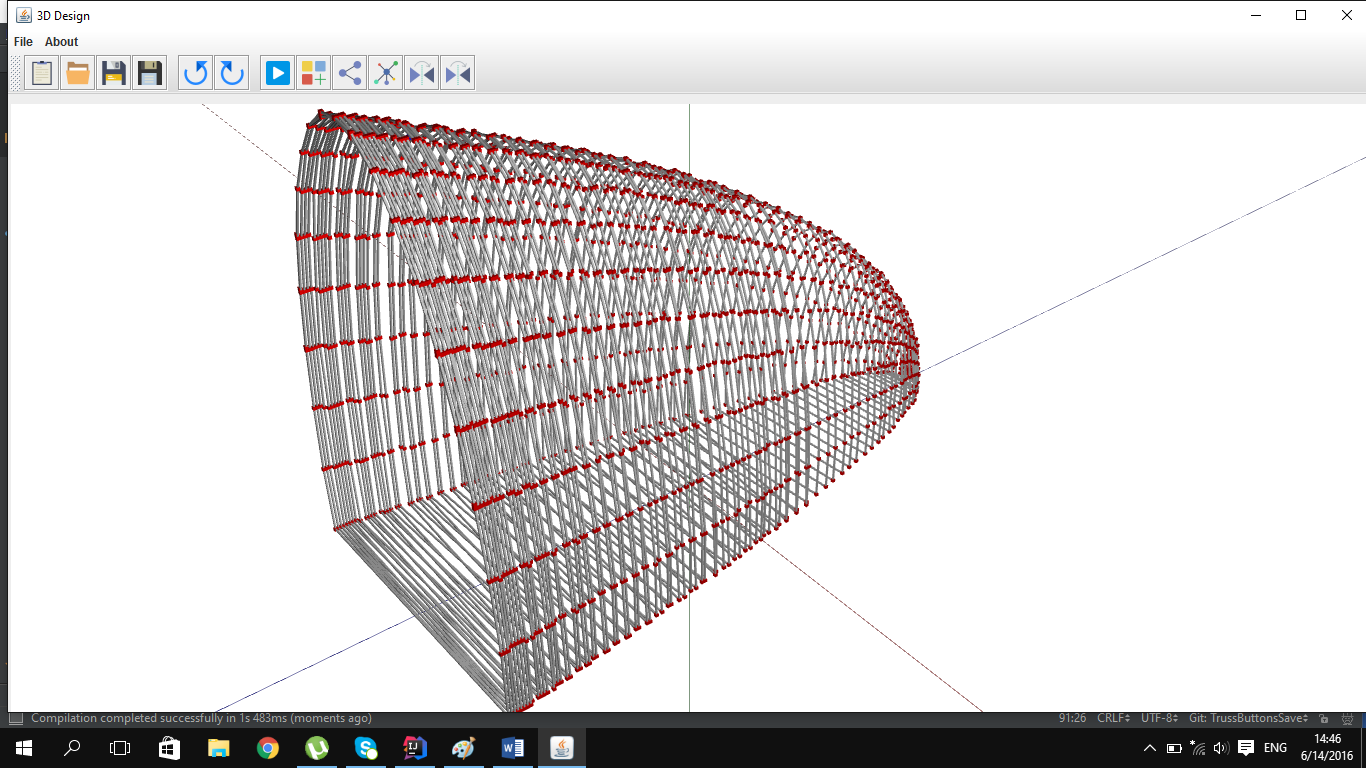
If we push the name of the Ellipsoid Scheme button it will create a beautiful Sphere schemes. For creating Ellipsoid scheme we are create in package model a package scheme and in this scheme we create a class Ellipsoid Scheme. In this class we create 2 method. First method generate method. It will create our model directly. For creating our model we create some variables in our method. We create our model by x, y and z equation. In this Scheme x, y, z will be

x= (sin (Pi /2 +t) + cos (Pi +s))

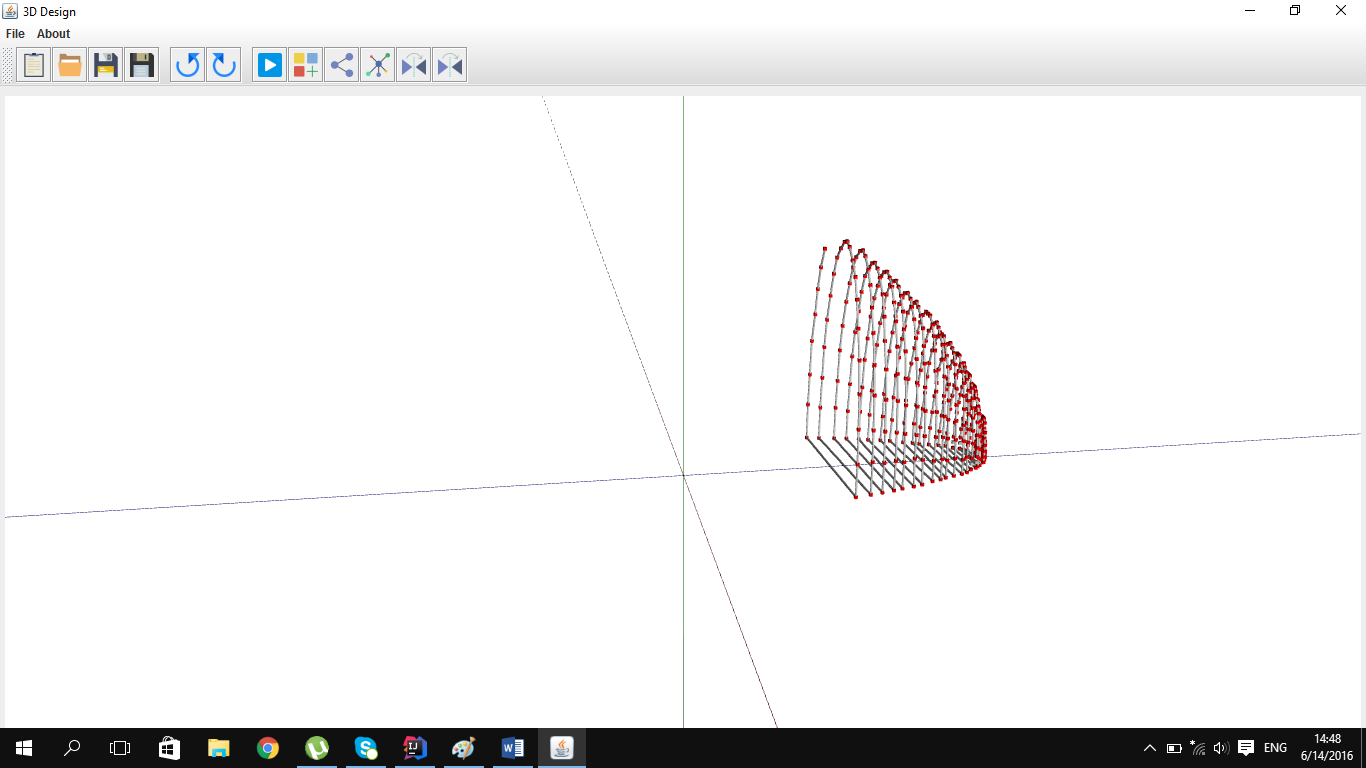
y= (sin (Pi /2 +t) + sin (Pi +s)

z= (2+cos (Pi +t))

t is crating nodes and bars in y and z direction

s is creating nodes and bars in y and x direction

Also we can create it by animation for this we must push Animation button and it will create our model step by step. In this method we can see every step of our nodes and models creating. For creating this animation we are create in our Ellipsoid Scheme class an action method which will create our model by animation.



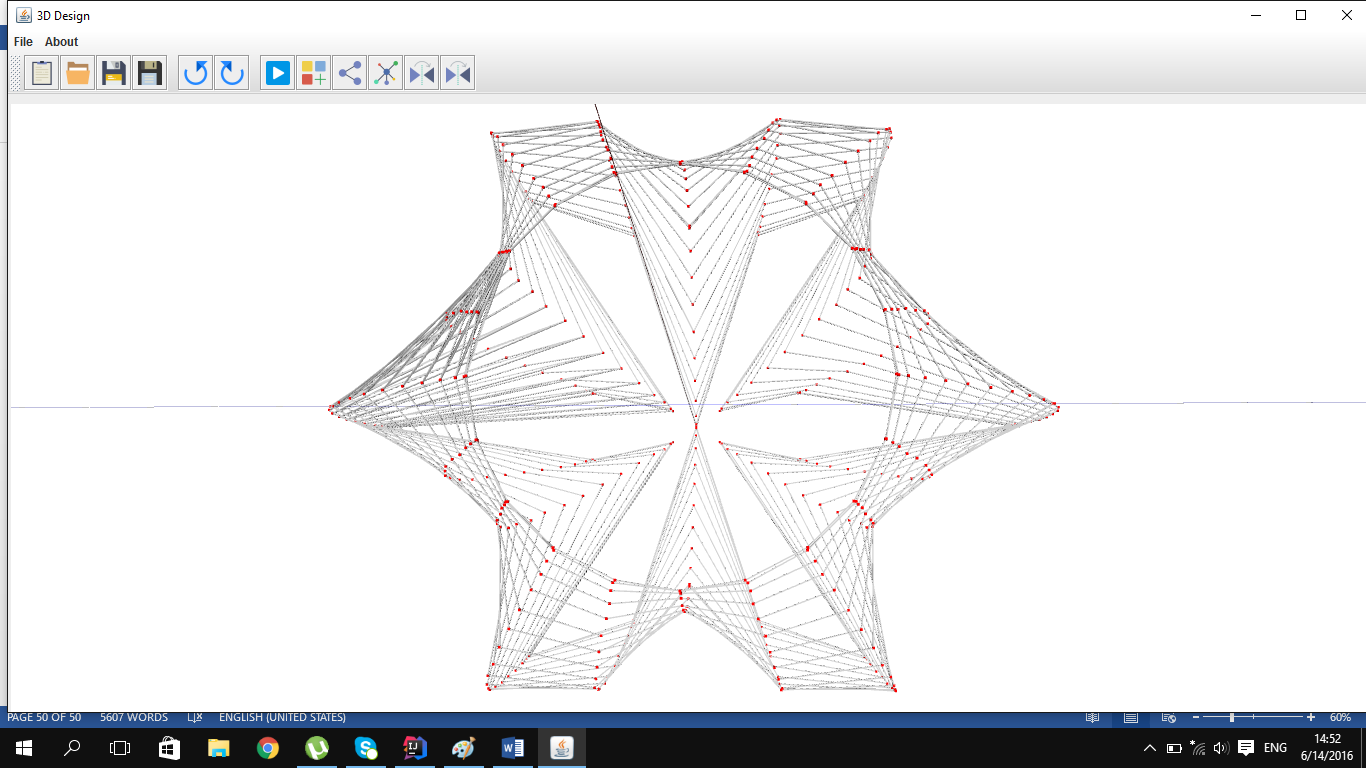
If we push the name of the Corrugated Scheme button it will create a beautiful Sphere schemes. For creating Corrugated scheme we are create in package model a package scheme and in this scheme we create a class Corrugated Scheme. In this class we create 2 method. First method generate method. It will create our model directly. For creating our model we create some variables in our method. We create our model by x, y and z equation. In this Scheme x, y, z will be

x= ((5 + cos (2 \* Pi\*t) + 2\*(1- sin (Pi\* t))\*cos (6 \* 2 \*Pi \* s) \* cos (2 \* Pi \*s)

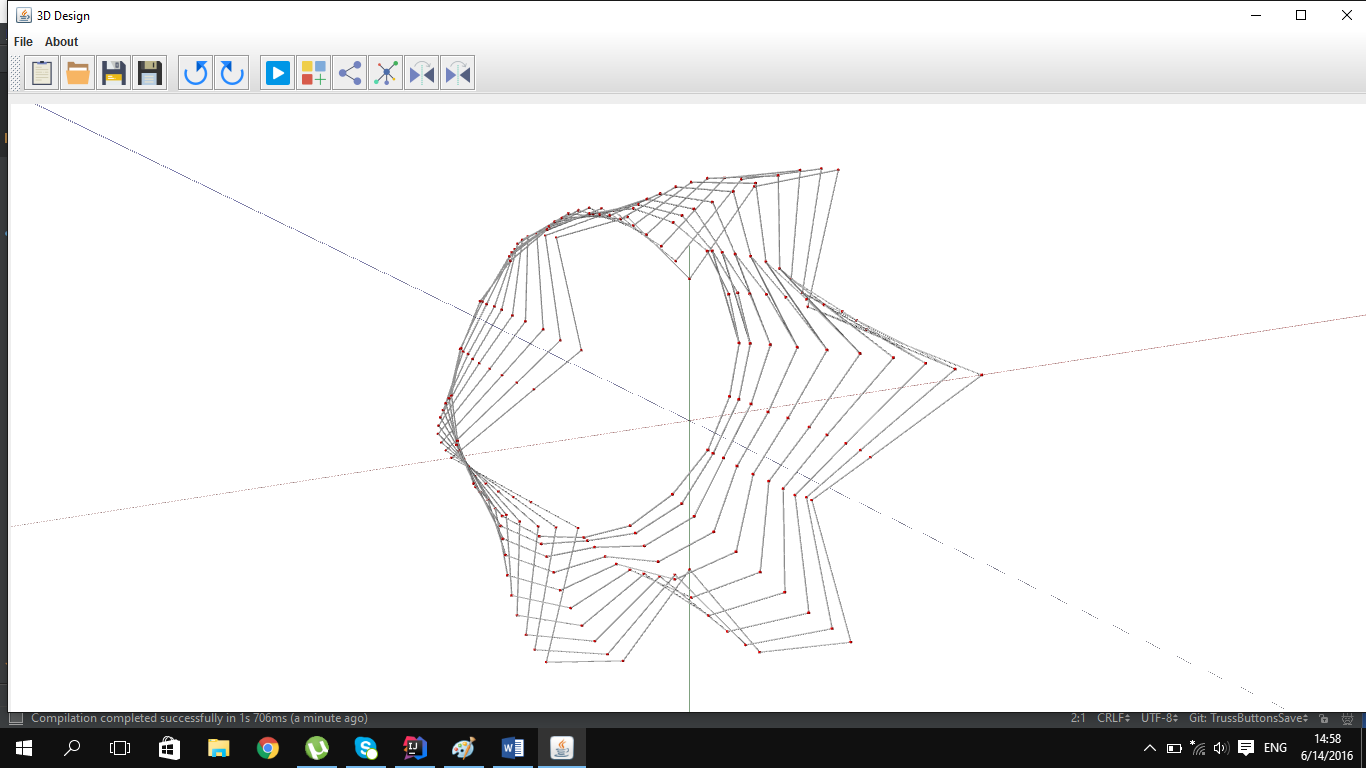
y= ((5 + cos (2 \* Pi\*t) + 2\*(1- sin (Pi\* t))\*cos (6 \* 2 \*Pi \* s) \* sin (2 \* Pi \*s)

z= (5 \*sin (Pi \* t)

t is crating nodes and bars in y and z direction

s is creating nodes and bars in y and x direction

Also we can create it by animation for this we must push Animation button and it will create our model step by step. In this method we can see every step of our nodes and models creating. For creating this animation we are create in our Corrugated Scheme class an action method which will create our model by animation.



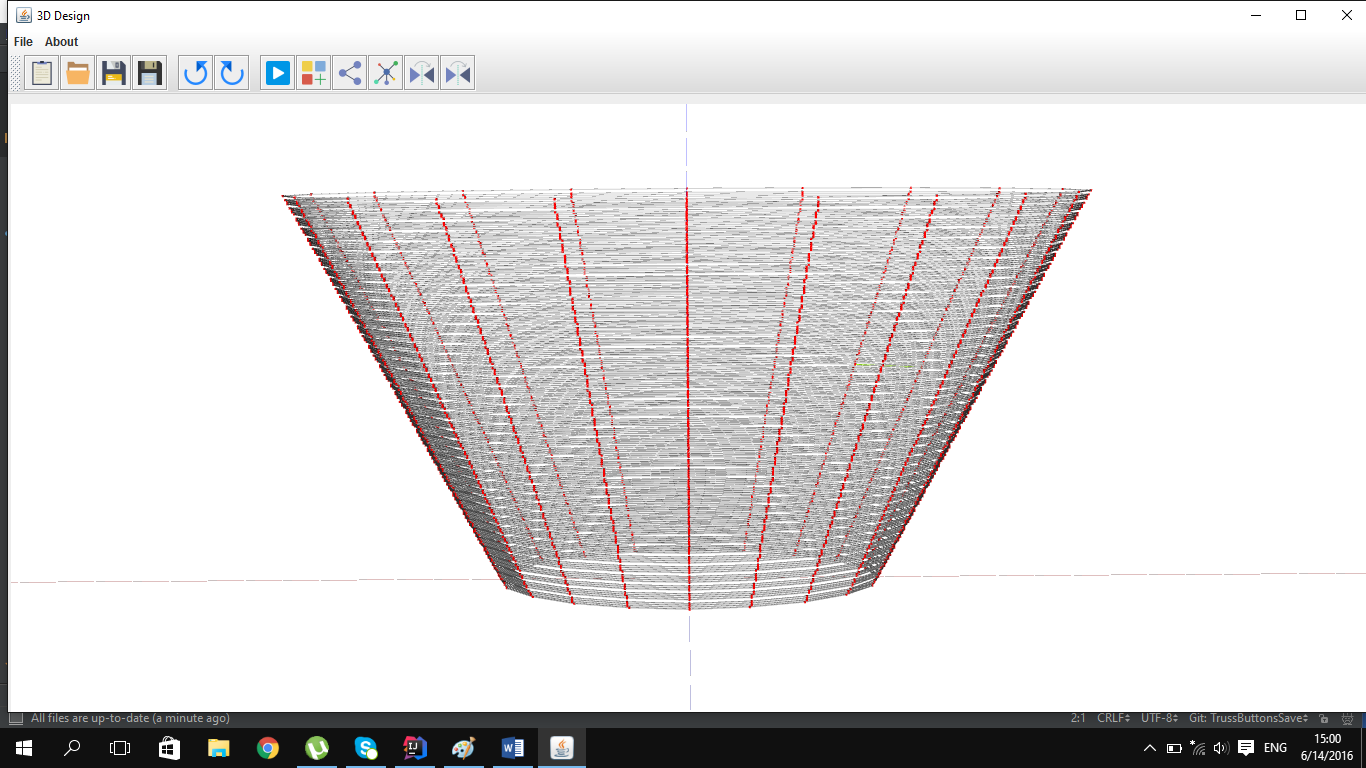
If we push the name of the SemiCon Scheme button it will create a beautiful Sphere schemes. For creating SemiCon scheme we are create in package model a package scheme and in this scheme we create a class SemiCon Scheme. In this class we create 2 method. First method generate method. It will create our model directly. For creating our model we create some variables in our method. We create our model by x, y and z equation. In this Scheme x, y, z will be

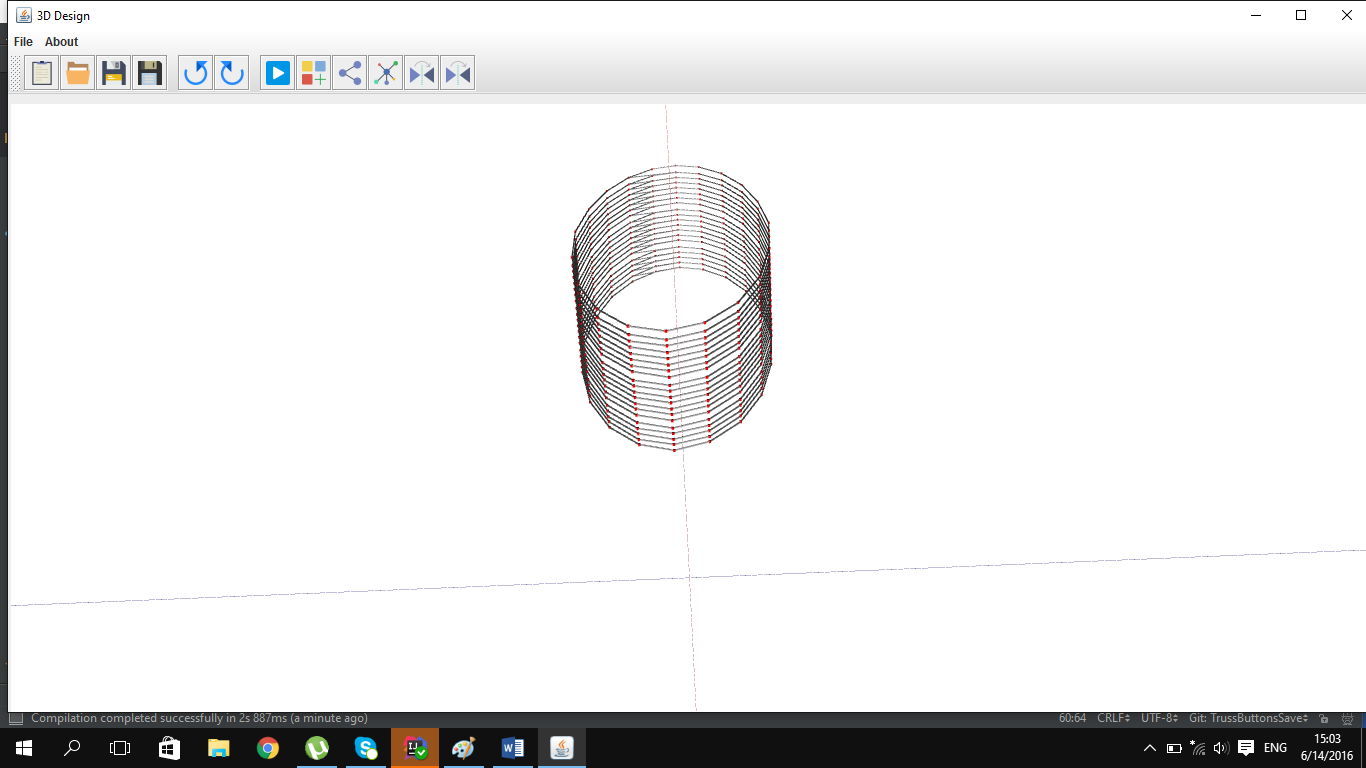
x = (4 \* (1 –t))

y = (4- 2\*t) \*cos (Pi \*s)

z= sin (Pi \* s)

t is crating nodes and bars in y and z direction

s is creating nodes and bars in y and x direction

Also we can create it by animation for this we must push Animation button and it will create our model step by step. In this method we can see every step of our nodes and models creating. For creating this animation we are create in our SemiCon Scheme class an action method which will create our model by animation.

**4.1 Send To Scad**