Criterion B

Flow of Program

Run Program

Welcome Frame

- Click Button to continue to shape select frame

Shape Select Frame

- Select one of 3 shape options out a RadioButton (Sts, Ctc, Stc)
- Click Button to pass option to Input frame

Input Frame

- Input frame will be constructed differently depending on the shape option passed
    - User will enter necessary inputs in input frame
        - If Sts was passed: Name, Height, Side, Side
        - If Ctc was passed: Name, Height, Radius, Radius, Sectors
        - If Stc was passed: Name, Height, Side, Radius, Sectors
- Click button
    - If invalid inputs are entered, an error message will appear
    - Inputs frame will pass inputs into database client, and database client will add entry to database
    - Inputs frame will pass inputs into different class (depending on the input frame construction type) when button is pressed and construct an object of the corresponding funnel type
        - Sts
        - Ctc
        - Stc

Computation Class

- For object construction, vertices array is constructed first, depending on user inputs, then faces array is constructed by assigning vertices to faces
    - Vertices[][3] array holds many vertices, each vertex holding 3 points
    - Faces[][4][3] array holds many faces, each face holding 4 vertices
- To construct the script, faces array is accessed, and database client is accessed to pull user settings out of database

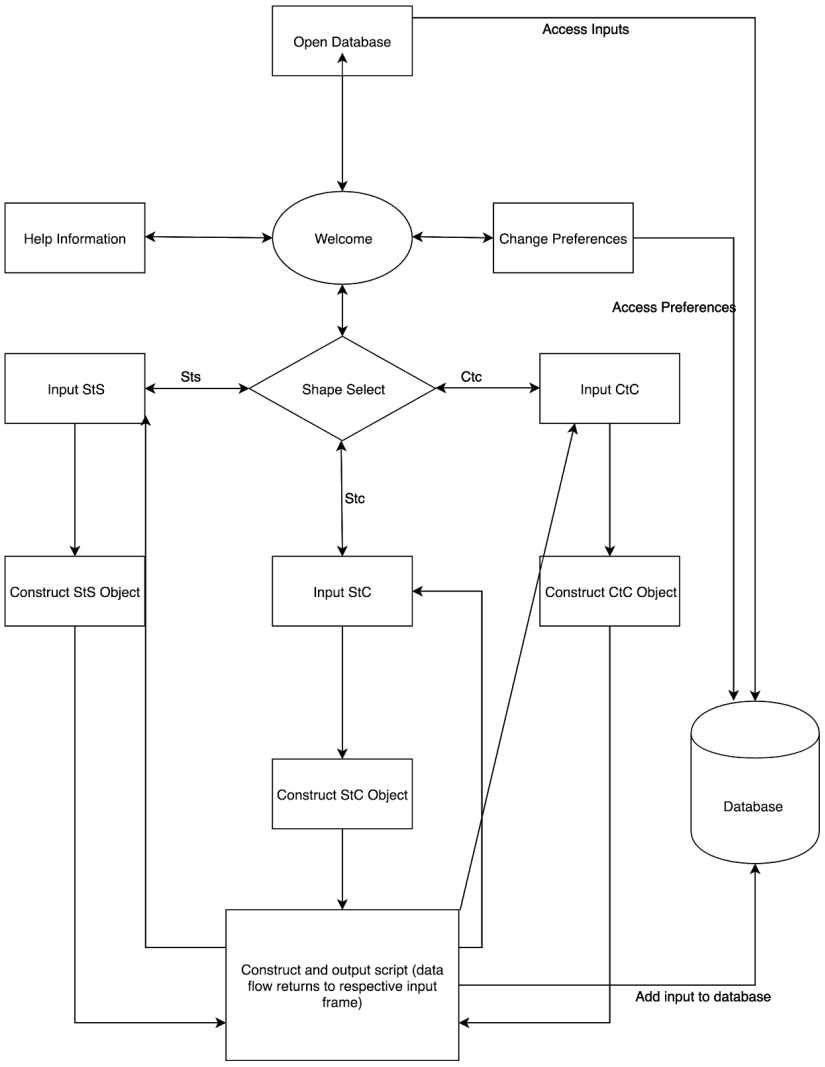
Data flow returns to input frame

- Input frame uses get method to extract string
- A file will be constructed on user computer with the contents of the string script, to get the path name where file will be saved, input frame accesses the database client to retrieve user settings out of database

- Input frame uses get methods to extract face array and passes faces into database client to add the face array to the database

Data flow returns to waiting for a button click in input frame

General Flowchart



Database

n… = row ID n has many after it with same ID number and similar information (actual ID will not include dots)
… = rows after with different IDs and values
# = unspecified text dependant on user input (# is placeholder)
ID = Highlight indicates column is a primary key

*not determined by user input of any kind, is determined by timestamp on computer

Database Tables Before Normalization

Setting Table
Setting value can be changed by user input

Setting ID	Setting
0	%USERPROFILE%
1	0.5
2	S
3	255
4	0
5	0

Funnel Storing Table

Funnel ID	Name	Timestamp*	Height	Radius	Radius	Side	Side	Sectors
1...	#	2019-08-14 12:13:30.024	#	#		#		#
1...								
2...	#	2019-08-14 12:14:07.821	#			#	#	
2...								
3...	#	2020-08-14 12:14:07.821	#	#	#			#
3...								
...								

Database Tables After 3NF
Primary keys are highlighted

Settings
Setting value can be changed by user input

Setting ID	Setting
0	%USERPROFILE%
1	0.5
2	S
3	255
4	0
5	0

Funnels

Funnel ID	Name	Timestamp*
1	#	2019-08-14 12:13:30.024
2	#	2019-08-14 12:14:07.821
3	#	2019-08-14 19:25:28.249
...	...	...

Height

Funnel ID	Height
1	#
2	#
3	#
...	...

Sides

Funnel ID	Side
1	#
1	#
2	#

...	...

Radiuses

Funnel ID	Radius
2	#
3	#
3	#
...	...

Sectors

Funnel ID	Sectors
2	#
3	#
...	...

Data Types

Setting ID: int

Setting: varchar

Funnel ID: int

Name: varchar

Timestamp: varchar

Height: varchar

Sides: varchar

Radius: varchar

Sectors: varchar

Representation of Defaults For Settings

***NOT* A DATABASE TABLE* (PURELY FOR INFORMATION PURPOSES)**

Setting Name	Setting ID	Default	Possible Values
Path	0	%USERPROFILE%	Any valid folder
Lineweight Thickness	1	0.5	0 through 2.11
VsCurrent	2	S	2D W H R C S E G S X
Red Value	3	255	1 through 255
Green Value	4	0	1 through 255
Blue Value	5	0	1 through 255

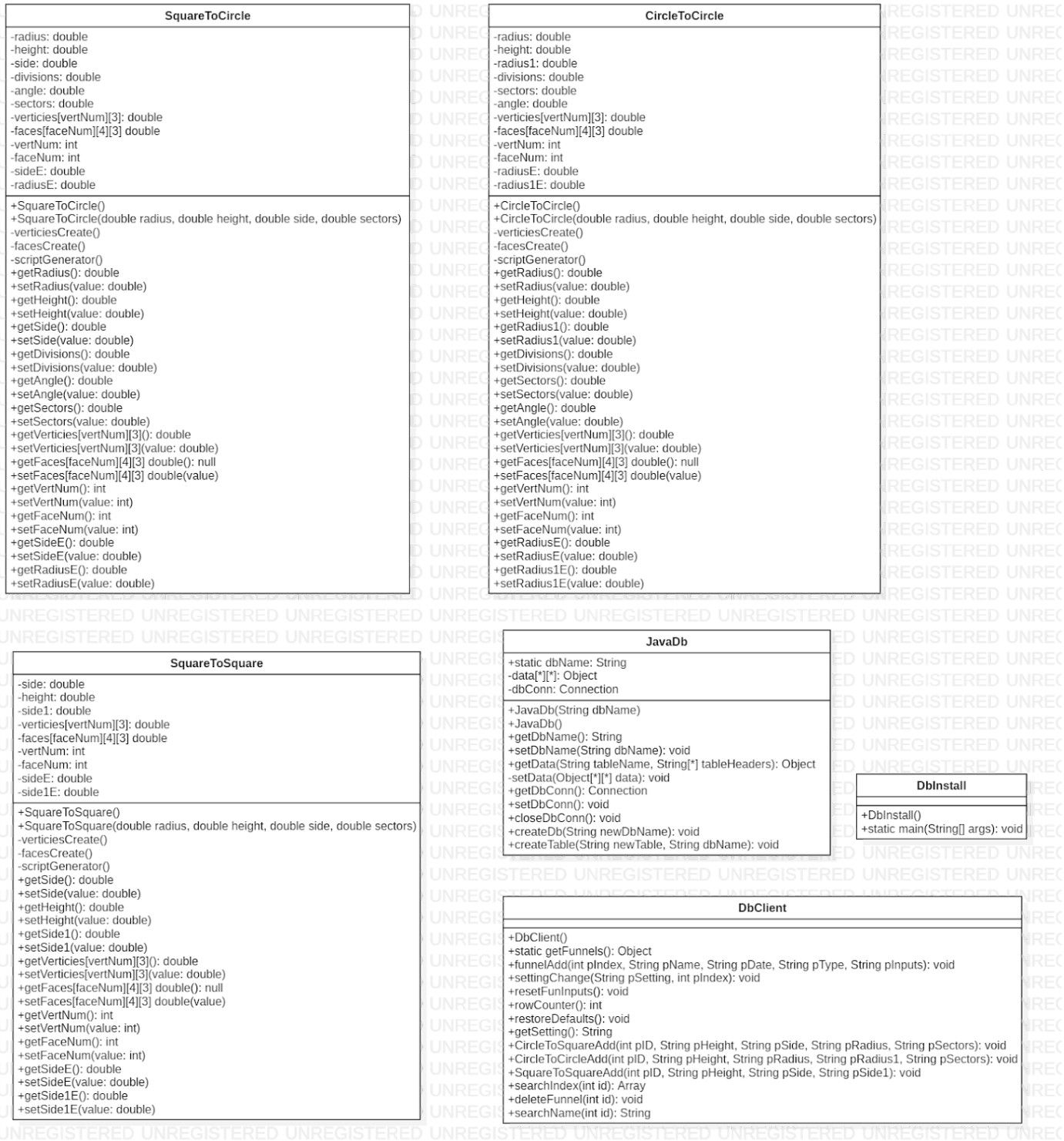
Classes

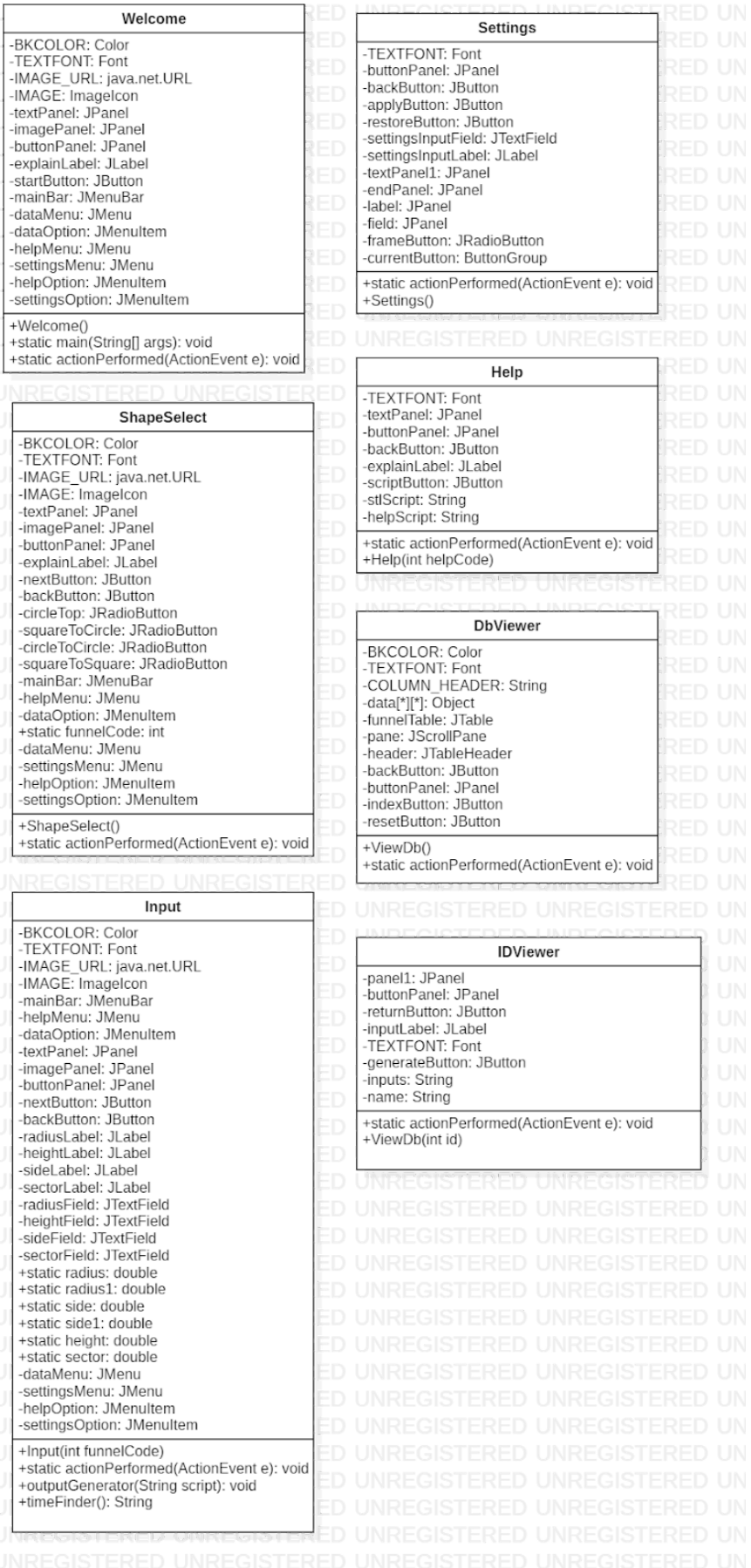
- Welcome
    - This class is the main GUI class that provides an introduction on how the program works, into the GUI, and how this program is structured
- ShapeSelect
    - This program allows the user to select the funnel type, and pass it into the input class
    - This will change the type of funnel object that will later be constructed
- Input
    - This class functions as the main centerpiece for script creation, input insertion, and database calling
    - First, a GUI is provided to insert inputs
    - When the inputs are inserted, funnel objects are constructed with passed funnel inputs
    - This class then extracts the script out of each object and creates a file containing the script
    - This class also adds the inputs into the database
- SquareToCircle

    - This is a class of the SquareToCircle funnel object
    - It contains vertex, face, and script managing methods
- SquareToSquare
    - This is a class of the SquareToSquare funnel object
    - It contains vertex, face, and script managing methods
- CircleToCircle
    - This is a class of the CircleToCircle funnel object
    - It contains vertex, face, and script managing methods
- Settings
    - This class provides a GUI to insert settings
    - This class will then take the values from the GUI, verify they are viable, and insert them into the settings database
- Help
    - This class provides information on how to run a script, and how to convert a funnel into a .stl file
- DbViewer
    - This program constructs a JTable containing the funnels table stored within the database
    - It also provides a GUI to Delete/view a specific ID, or clear the entire database
- IDViewer
    - This class receives a passed ID contained within the database and displays them in a GUI
    - This class can access the inputs class to use the same script generation method
    - This allows the database to be used as a "library" for scripts
- DbClient
    - This class statically communicates with the JavaDb class to view, delete, or insert to database
- DbInstall
    - This class creates the database and its tables
- JavaDb
    - This class communicates with and sends commands to the database manager

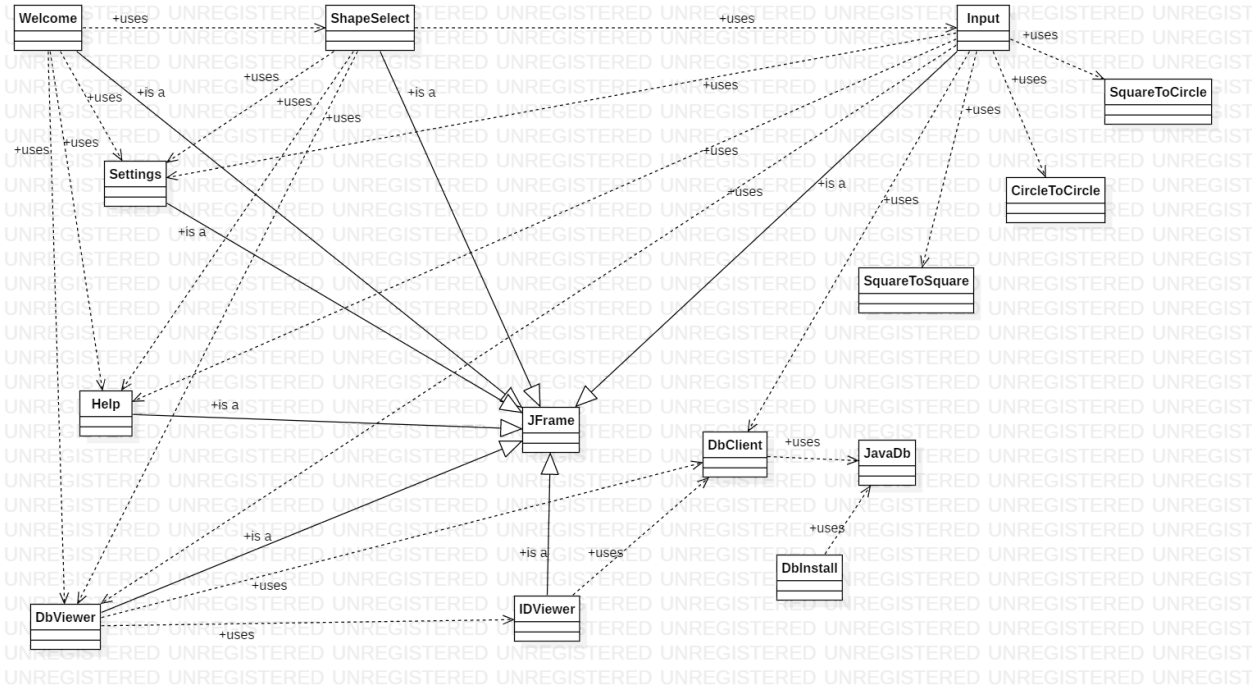
UML DIAGRAM

Classes


SquareToCircle
-radius: double
-height: double
-side: double
-divisions: double
-angle: double
-sectors: double
-verticies[vertNum][3]: double
-faces[faceNum][4][3] double
-vertNum: int
-faceNum: int
-sideE: double
-radiusE: double
+SquareToCircle()
+SquareToCircle(double radius, double height, double side, double sectors)
-verticiesCreate()
-facesCreate()
-scriptGenerator()
+getRadius(): double
+setRadius(value: double)
+getHeight(): double
+setHeight(value: double)
+getSide(): double
+setSide(value: double)
+getDivisions(): double
+setDivisions(value: double)
+getAngle(): double
+setAngle(value: double)
+getSectors(): double
+setSectors(value: double)
+getVerticies[vertNum][3](): double
+setVerticies[vertNum][3](value: double)
+getFaces[faceNum][4][3] double(): null
+setFaces[faceNum][4][3] double(value)
+getVertNum(): int
+setVertNum(value: int)
+getFaceNum(): int
+setFaceNum(value: int)
+getSideE(): double
+setSideE(value: double)
+getRadiusE(): double
+setRadiusE(value: double)
CircleToCircle
-radius: double
-height: double
-radius1: double
-divisions: double
-sectors: double
-angle: double
-verticies[vertNum][3]: double
-faces[faceNum][4][3] double
-vertNum: int
-faceNum: int
-radiusE: double
-radius1E: double
+CircleToCircle()
+CircleToCircle(double radius, double height, double side, double sectors)
-verticiesCreate()
-facesCreate()
-scriptGenerator()
+getRadius(): double
+setRadius(value: double)
+getHeight(): double
+setHeight(value: double)
+getRadius1(): double
+setRadius1(value: double)
+getDivisions(): double
+setDivisions(value: double)
+getSectors(): double
+setSectors(value: double)
+getAngle(): double
+setAngle(value: double)
+getVerticies[vertNum][3](): double
+setVerticies[vertNum][3](value: double)
+getFaces[faceNum][4][3] double(): null
+setFaces[faceNum][4][3] double(value)
+getVertNum(): int
+setVertNum(value: int)
+getFaceNum(): int
+setFaceNum(value: int)
+getRadiusE(): double
+setRadiusE(value: double)
+getRadius1E(): double
+setRadius1E(value: double)
SquareToSquare
-side: double
-height: double
-side1: double
-verticies[vertNum][3]: double
-faces[faceNum][4][3] double
-vertNum: int
-faceNum: int
-sideE: double
-side1E: double
+SquareToSquare()
+SquareToSquare(double radius, double height, double side, double sectors)
-verticiesCreate()
-facesCreate()
-scriptGenerator()
+getSide(): double
+setSide(value: double)
+getHeight(): double
+setHeight(value: double)
+getSide1(): double
+setSide1(value: double)
+getVerticies[vertNum][3](): double
+setVerticies[vertNum][3](value: double)
+getFaces[faceNum][4][3] double(): null
+setFaces[faceNum][4][3] double(value)
+getVertNum(): int
+setVertNum(value: int)
+getFaceNum(): int
+setFaceNum(value: int)
+getSideE(): double
+setSideE(value: double)
+getSide1E(): double
+setSide1E(value: double)
JavaDb
+static dbName: String
-data[*][*]: Object
-dbConn: Connection
+JavaDb(String dbName)
+JavaDb()
+getDbName(): String
+setDbName(String dbName): void
+getData(String tableName, String[*] tableHeaders): Object
-setData(Object[*][*] data): void
+getDbConn(): Connection
+setDbConn(): void
+closeDbConn(): void
+createDb(String newDbName): void
+createTable(String newTable, String dbName): void
DbInstall
+DbInstall()
+static main(String[] args): void
DbClient
+DbClient()
+static getFunnels(): Object
+funnelAdd(int pIndex, String pName, String pDate, String pType, String pInputs): void
+settingChange(String pSetting, int pIndex): void
+resetFunInputs(): void
+rowCounter(): int
+restoreDefaults(): void
+getSetting(): String
+CircleToSquareAdd(int pID, String pHeight, String pSide, String pRadius, String pSectors): void
+CircleToCircleAdd(int pID, String pHeight, String pRadius, String pRadius1, String pSectors): void
+SquareToSquareAdd(int pID, String pHeight, String pSide, String pSide1): void
+searchIndex(int id): Array
+deleteFunnel(int id): void
+searchName(int id): String

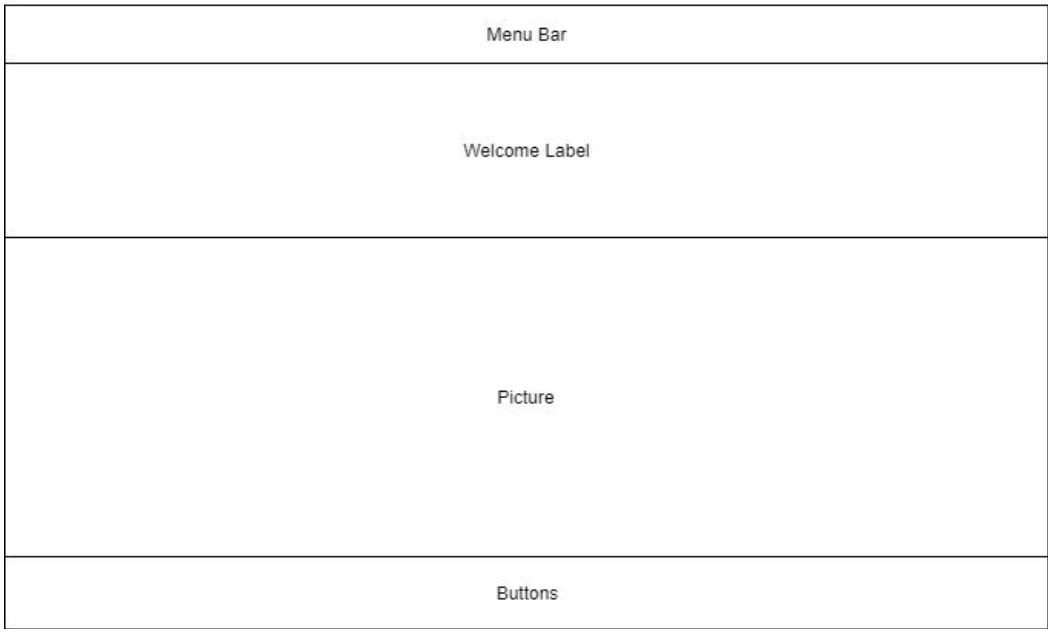

Welcome
-BKCOLOR: Color
-TEXTFONT: Font
-IMAGE_URL: java.net.URL
-IMAGE: ImageIcon
-textPanel: JPanel
-imagePanel: JPanel
-buttonPanel: JPanel
-explainLabel: JLabel
-startButton: JButton
-mainBar: JMenuBar
-dataMenu: JMenu
-dataOption: JMenuItem
-helpMenu: JMenu
-settingsMenu: JMenu
-helpOption: JMenuItem
-settingsOption: JMenuItem
+Welcome()
+static main(String[] args): void
+static actionPerformed(ActionEvent e): void
Settings
-TEXTFONT: Font
-buttonPanel: JPanel
-backButton: JButton
-applyButton: JButton
-restoreButton: JButton
-settingsInputField: JTextField
-settingsInputLabel: JLabel
-textPanel1: JPanel
-endPanel: JPanel
-label: JPanel
-field: JPanel
-frameButton: JRadioButton
-currentButton: ButtonGroup
+static actionPerformed(ActionEvent e): void
+Settings()
Help
-TEXTFONT: Font
-textPanel: JPanel
-buttonPanel: JPanel
-backButton: JButton
-explainLabel: JLabel
-scriptButton: JButton
-stlScript: String
-helpScript: String
+static actionPerformed(ActionEvent e): void
+Help(int helpCode)
ShapeSelect
-BKCOLOR: Color
-TEXTFONT: Font
-IMAGE_URL: java.net.URL
-IMAGE: ImageIcon
-textPanel: JPanel
-imagePanel: JPanel
-buttonPanel: JPanel
-explainLabel: JLabel
-nextButton: JButton
-backButton: JButton
-circleTop: JRadioButton
-squareToCircle: JRadioButton
-circleToCircle: JRadioButton
-squareToSquare: JRadioButton
-mainBar: JMenuBar
-helpMenu: JMenu
-dataOption: JMenuItem
+static funnelCode: int
-dataMenu: JMenu
-settingsMenu: JMenu
-helpOption: JMenuItem
-settingsOption: JMenuItem
+ShapeSelect()
+static actionPerformed(ActionEvent e): void
DbViewer
-BKCOLOR: Color
-TEXTFONT: Font
-COLUMN_HEADER: String
-data[*][*]: Object
-funnelTable: JTable
-pane: JScrollPane
-header: JTableHeader
-backButton: JButton
-buttonPanel: JPanel
-indexButton: JButton
-resetButton: JButton
+ViewDb()
+static actionPerformed(ActionEvent e): void
Input
-BKCOLOR: Color
-TEXTFONT: Font
-IMAGE_URL: java.net.URL
-IMAGE: ImageIcon
-mainBar: JMenuBar
-helpMenu: JMenu
-dataOption: JMenuItem
-textPanel: JPanel
-imagePanel: JPanel
-buttonPanel: JPanel
-nextButton: JButton
-backButton: JButton
-radiusLabel: JLabel
-heightLabel: JLabel
-sideLabel: JLabel
-sectorLabel: JLabel
-radiusField: JTextField
-heightField: JTextField
-sideField: JTextField
-sectorField: JTextField
+static radius: double
+static radius1: double
+static side: double
+static side1: double
+static height: double
+static sector: double
-dataMenu: JMenu
-settingsMenu: JMenu
-helpOption: JMenuItem
-settingsOption: JMenuItem
+Input(int funnelCode)
+static actionPerformed(ActionEvent e): void
+outputGenerator(String script): void
+timeFinder(): String
IDViewer
-panel1: JPanel
-buttonPanel: JPanel
-returnButton: JButton
-inputLabel: JLabel
-TEXTFONT: Font
-generateButton: JButton
-inputs: String
-name: String
+static actionPerformed(ActionEvent e): void
+ViewDb(int id)

Class Relationships

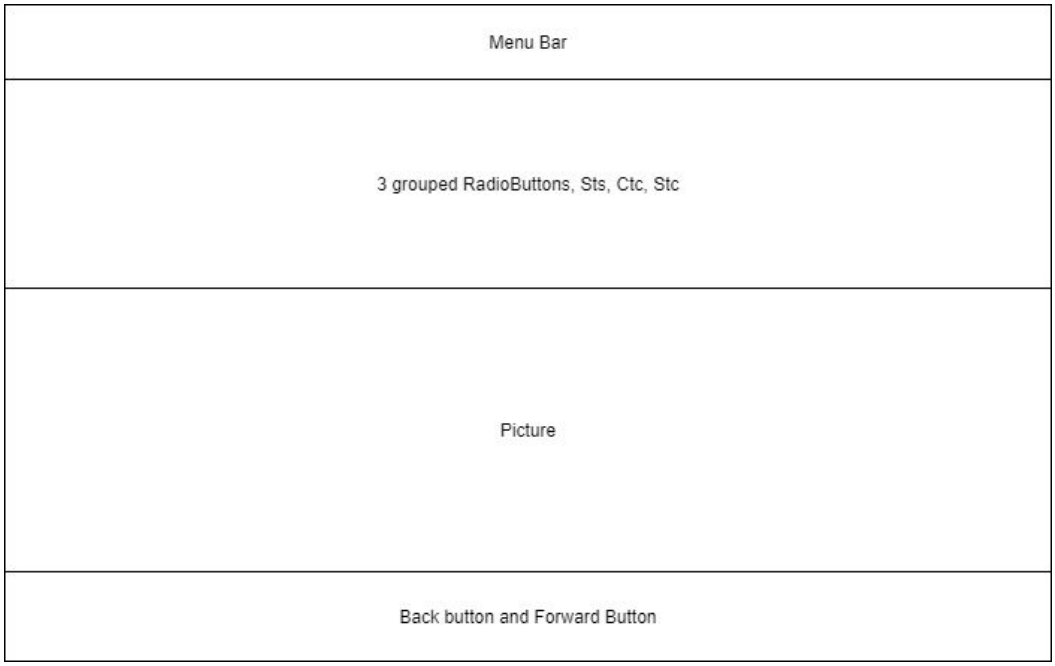


Graphical Visualization

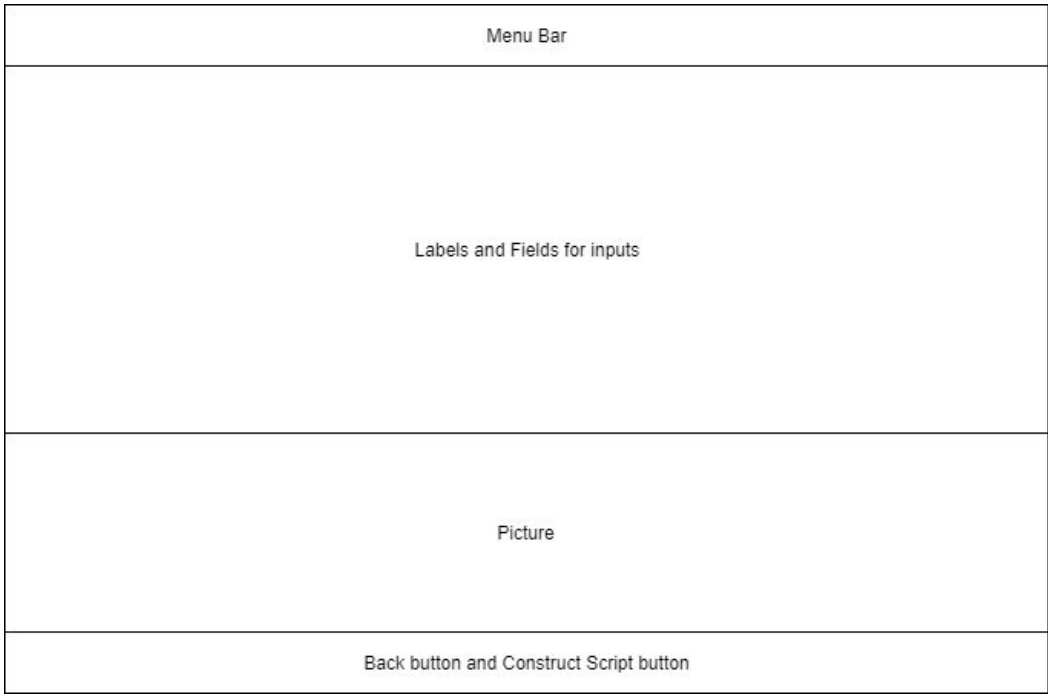
Welcome Frame



Shape Select

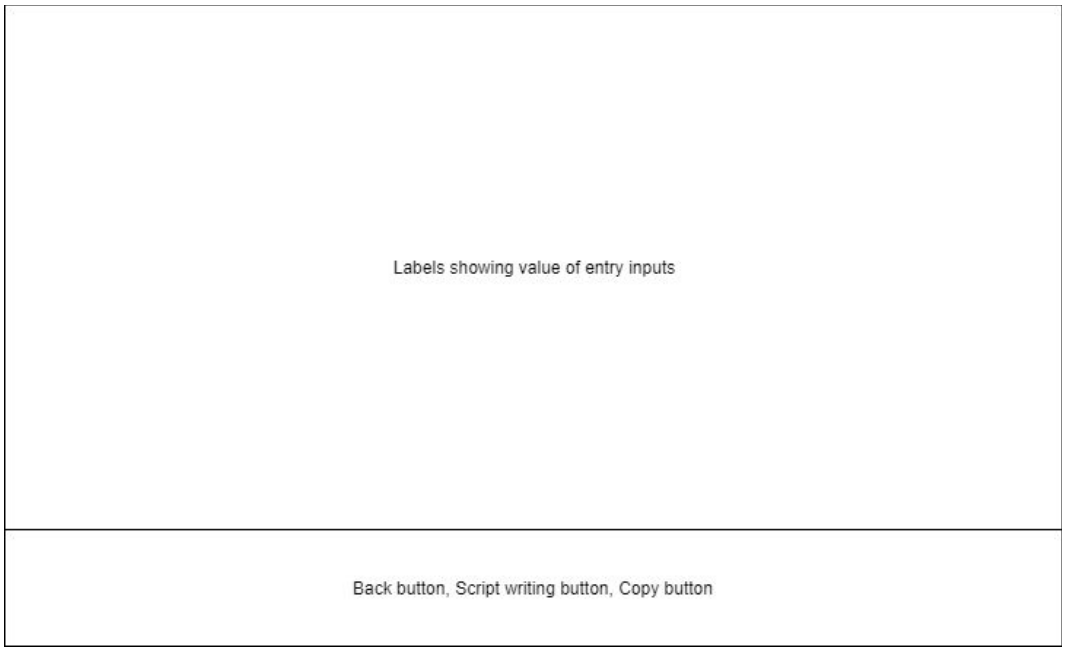


Inputs

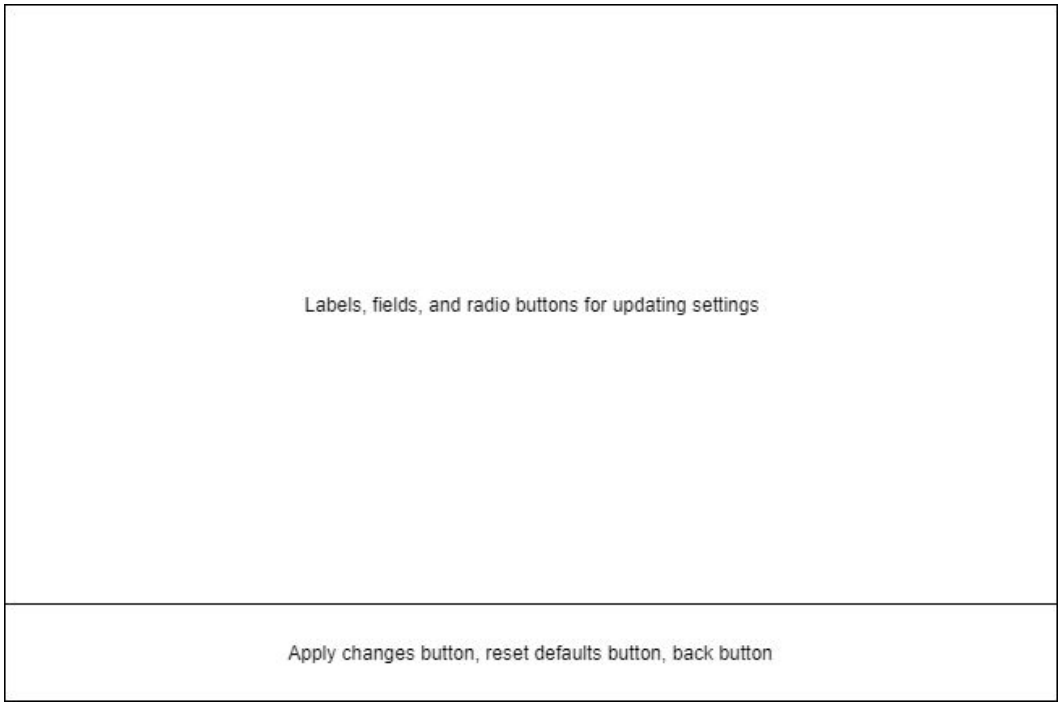


Database

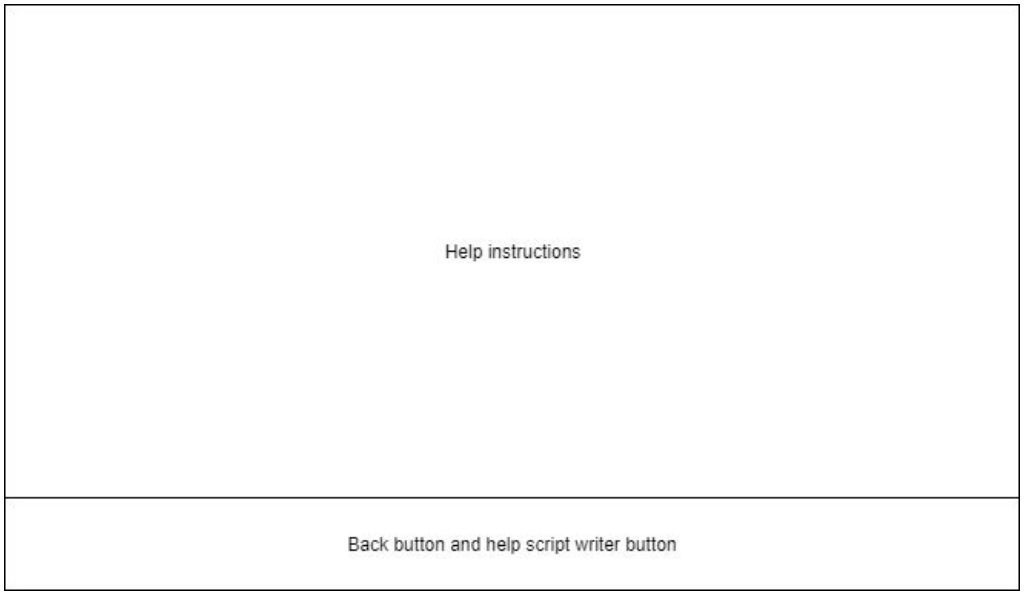
Entry Viewer



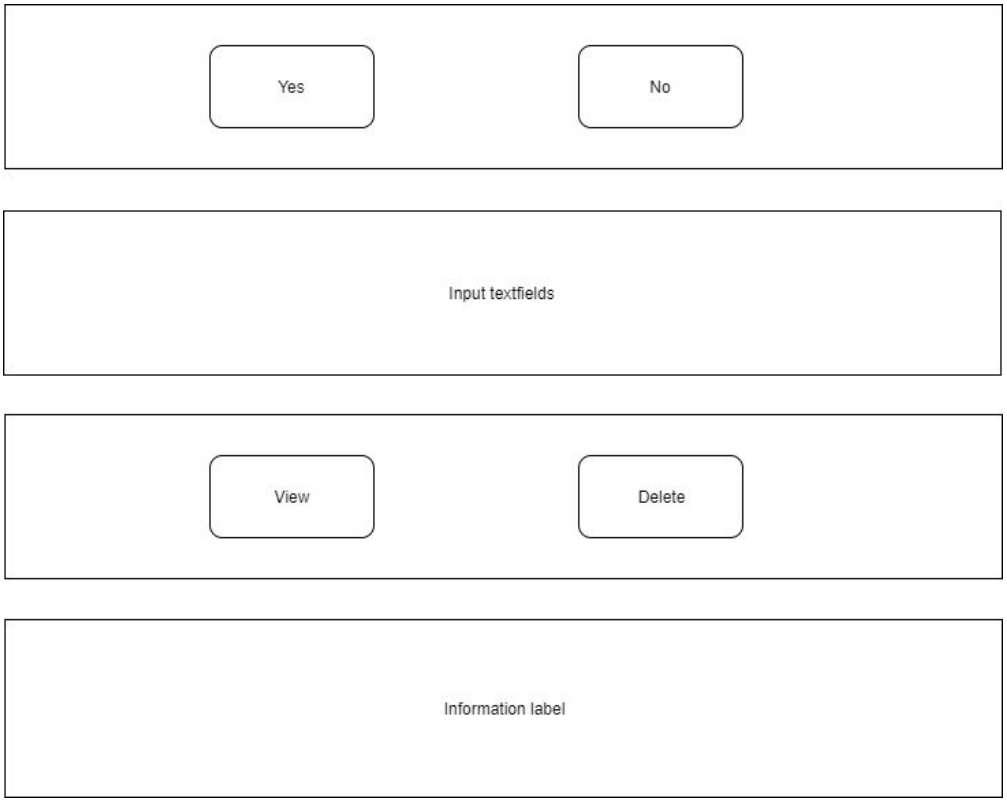
Preferences Change



Help

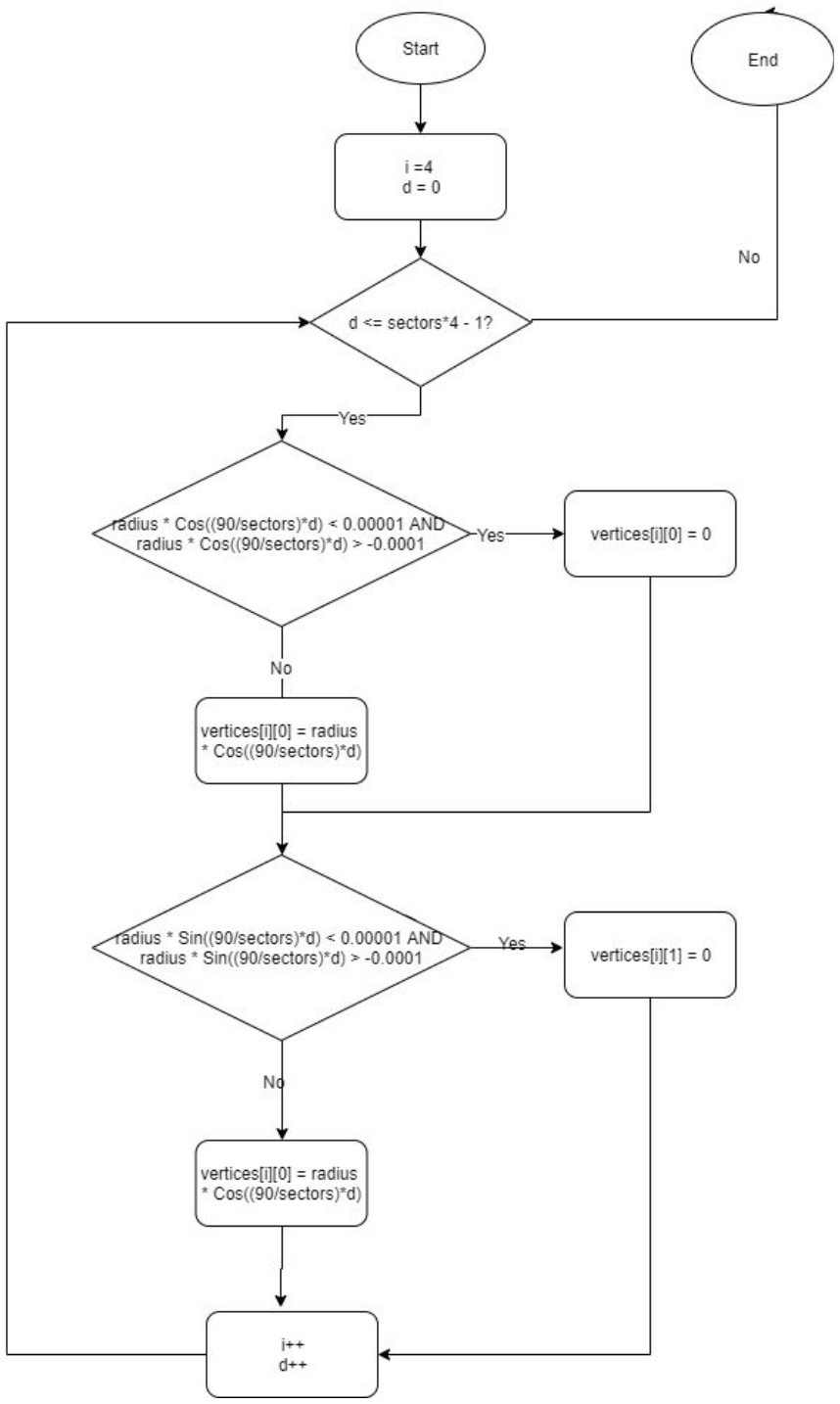


Dialog Boxes



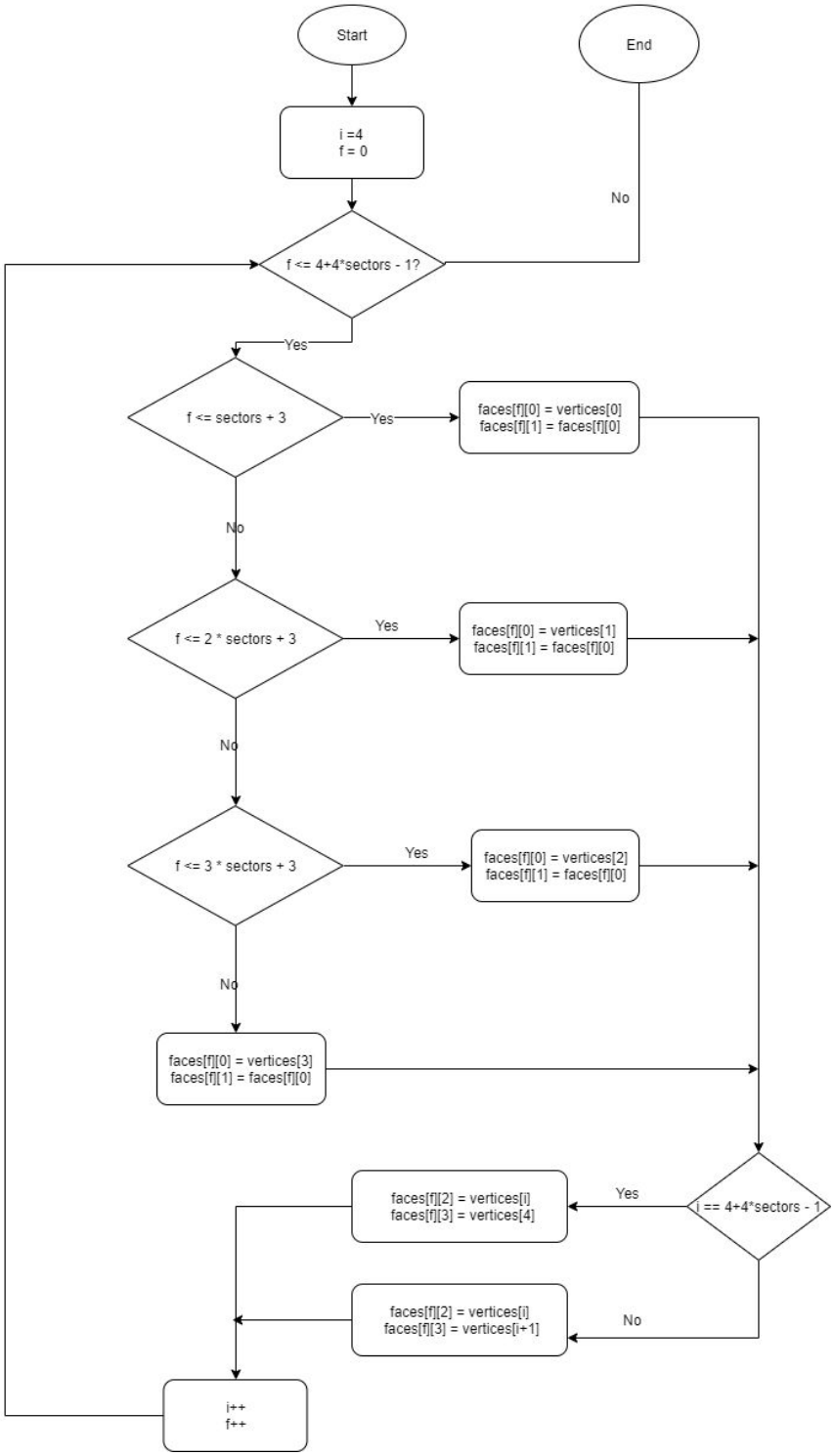
Circle Vertex Loop

The following flowchart assigns points to vertices[][] and is contained within the verticesCreate algorithm. “sectors” is a pre initialized input. “vertices[][]” is a predefined array.



Sector Face Loop

The following flowchart assigns vertices to faces[][] and is contained within the facesCreate algorithm. “sectors” is a pre initialized input. “vertices[][]” is a pre initialized array. “Faces[][]” is a predefined array.



Input Variables

Radius: double

Sectors: double

Side: double

Height: double

Angle: double = 90/sectors
Script: String
Array vertices[#][3] (# represents a variable number of arrays)
Array faces [#][4][3] (# represents a variable number of 2d arrays)

Program Data Structures and Data Types

Name	Data structures and Data Types
Square To Square	2d array[][], 3d array[][][], String, double
Circle To Circle	2d array[][], 3d array[][][], String, double
Square To Circle	2d array[][], 3d array[][][], String, double
vertices	2d array[][]
faces	3d array[][][]
JavaDb	ArrayList

Pseudocode

assigns vertices to vertices array by assigning 3 numbers to a vertex
verticiesCreate()

```
    assign square vertices by index
    vertices[0][0]= (side)/2
    vertices[0][1]= (side)/2
    vertices[0][2]= height

    vertices[1][0]= - (side)/2
    vertices[1][1]= (side)/2
    vertices[1][2]= height

    vertices[2][0]= - (side)/2
    vertices[2][1]= - (side)/2
    vertices[2][2]= height

    vertices[3][0]= (side)/2
    vertices[3][1]= - (side)/2
    vertices[3][2]= height

    count through circle with i and divisions with d
    i = 4
```

```
        d = 0
        loop for through all divisions (d)
                vertices[i][0] = radius * cos (angle * d)
                vertices[i][1] = radius * sin (angle * d)
                vertices[i][2] = 0
                i++
                d++
        end loop
end method

assigns faces to faces array by assigning 4 vertices to a face
facesCreate()
        assign square vertices by index

        faces[0][0] = vertices[0]
        faces[0][1] = vertices[1]
        faces[0][2] = vertices[4 + sectors]
        faces[0][3] = vertices[4 + sectors]

        faces[1][0] = vertices[1]
        faces[1][1] = vertices[2]
        faces[1][2] = vertices[4 + (2 * sectors)]
        faces[1][3] = vertices[4 + (2 * sectors)]

        faces[2][0] = vertices[2]
        faces[2][1] = vertices[3]
        faces[2][2] = vertices[4 + (3 * sectors)]
        faces[2][3] = vertices[4 + (3 * sectors)]

        faces[3][0] = vertices[3]
        faces[3][1] = vertices[0]
        faces[3][2] = vertices[4]
        faces[3][3] = vertices[4]

        count through faces with f and vertex index with i
        i = 4
        f = 4
        loop for through all faces
                assigns first two (identical) vertices, depending on quadrant
                if f is on first quadrant
                        faces[f][0] = square vertex vertices[0]
                        faces[f][1] = square vertex vertices[0]
                else if is on second quadrant
```

```
                faces[f][0] = square vertex vertices[1]
            faces[f][1] = square vertex vertices[1]
        else if is on third quadrant
                faces[f][0] = square vertex vertices[2]
                faces[f][1] = square vertex vertices[2]
        else
                faces[f][0] = square vertex vertices[3]
                faces[f][1] = square vertex vertices[3]
            assigns the final two vertices by looping around
            if NOT last face
            faces[f][2] = circle vertex vertices[i]
                faces[f][3] = circle vertex vertices[i+1]
            else
                faces[f][2] = circle vertex vertices[i]
                faces[f][3] = circle vertex vertices[4]
            f++
            i++
        end loop
end method

returns the generated script
String scriptCreate()
        S = “”
        writes the script, \n adds a new line
        f = 0
        loop for through all faces
            S = S + "3dface\n"
            S = S + faces[f][0][0] + "," + faces[f][0][1] + "," + faces[f][0][2] + "\n"
            S = S + faces[f][1][0] + "," + faces[f][1][1] + "," + faces[f][1][2] + "\n"
            S = S + faces[f][2][0] + "," + faces[f][2][1] + "," + faces[f][2][2] + "\n"
            S = S + faces[f][3][0] + "," + faces[f][3][1] + "," + faces[f][3][2] + "\n"
            S = S+ "\n"
            f++
        end loop
        return S
```

Test Plan

Success Criteria	Test Type	Nature of Test	Example
Script files which accurately contain faces are created	Check if the script generated works in CAD	Make sure the script can be run	Run the script, and make sure it doesn’t return an error and

			shows the model
User can select from three funnel profiles (Sts, Ctc, Stc)	Check that program can differentiate between different selected types	Check that the requested funnel inputs match the selected funnel type	Compare the requested funnel inputs from the input frame to the selected funnel type from the shape select frame
Funnel vertices are accurately calculated	Check all vertices are correct	Run generated script in CAD and check 3d model's properties	Check the number of vertices is correct, the spacing between them is correct, and the values are accurate
Funnel vertices are accurately mapped to faces	Check all faces are correct	Run generated script in CAD and check 3d model's properties	Check the number of faces is correct, each face is a fully formed shape, and that they each contain correct vertices
Database can insert funnel inputs	Check if database can insert inputs	Compare inputs of database to inputs I sent	Send inputs to database, recall them and compare them to the original inputs
Database can delete funnel inputs	Check delete function of database client	Delete a funnel, make sure it doesn't show up in the database table anymore, and make sure it doesn't ruin ID	Create numerous funnels, delete numerous funnels, and create numerous funnels. This will check to make sure it can delete, and that the delete function doesn't break database
Database can update script settings	Update settings in database	Check for database errors	Insert settings into the database, check if the database catches any errors
Database can retrieve script settings (which are added to scripts)	Retrieve Settings from database	Check script for errors	Make sure the script is saved in the right location, and contains the proper settings

Database can retrieve funnel inputs	Retrieve funnel inputs for a specific ID from database	Check that funnel inputs are accurate for specific IDs	Insert funnel inputs into the database, then retrieve inputs and check that they match
Program is Robust	Check to make sure program is error proof	Program will return an output or an error message for any action. No java non caught error messages.	Check every element of user input for failings, such as text length, text type, or content of text

Amendments

Capping Algorithm and Thickness

//to assign the second funnel, my program will add half of size of F and half of size of V to each face index and vertex index, respectively

//capping algorithm connect the two layers
F[2 * (N*4 + 4)][0] = V[0]
F[2 * (N*4 + 4)][1] = V[1]
F[2 * (N*4 + 4)][2] = V[(N*4 + 4) + 1]
F[2 * (N*4 + 4)][3] = V[t(N*4 + 4)]

F[(2 * (N*4 + 4)) + 1][0] = V1]
F[(2 * (N*4 + 4)) + 1][1] = V[2]
F[(2 * (N*4 + 4)) + 1][2] = V[(N*4 + 4) + 2]
F[(2 * (N*4 + 4)) + 1][3] = V[(N*4 + 4) + 1]

F[(2 * (N*4 + 4)) + 2][0] = V[2]
F[(2 * (N*4 + 4)) + 2][1] = V[3]
F[(2 * (N*4 + 4)) + 2][2] = V[(N*4 + 4) + 3]
F[(2 * (N*4 + 4)) + 2][3] = V[(N*4 + 4) + 2]

F[(2 * (N*4 + 4)) + 3][0] = V[3]
F[(2 * (N*4 + 4)) + 3][1] = V[0]
F[(2 * (N*4 + 4)) + 3][2] = V[(N*4 + 4)]
F[(2 * (N*4 + 4)) + 3][3] = V[(N*4 + 4) + 3]

//capping

```
k = 4
l = 4 + (N*4 + 4)
foop for f = (2 * (N*4 + 4)) + 4 to f <= (3 * (N*4 + 4)) - 1
    if (f == ((N*4 + 4) * 3) - 1)
            F[f][0] = V[k]
            F[f][1] = V[4]
            F[f][2] = V[4 + (N*4 + 4)]
            F[f][3] = V[l]
    else
            F[f][0] = V[k]
            F[f][1] = V[k+1]
            F[f][2] = V[l+1]
            F[f][3] = V[l]

        k++
        l++
end loop
```