BCM2053
ASSIGNMENT REPORT

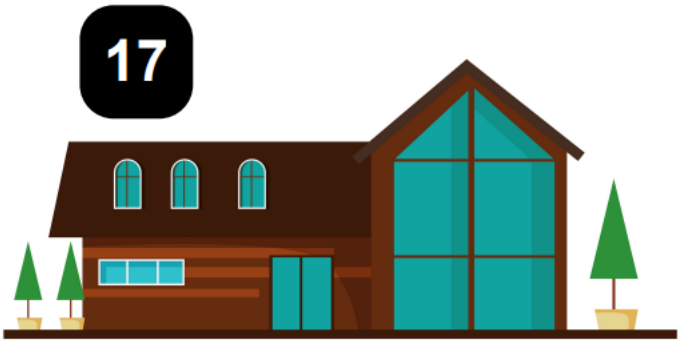
GROUP NAME: WOODY

MEMBERS:

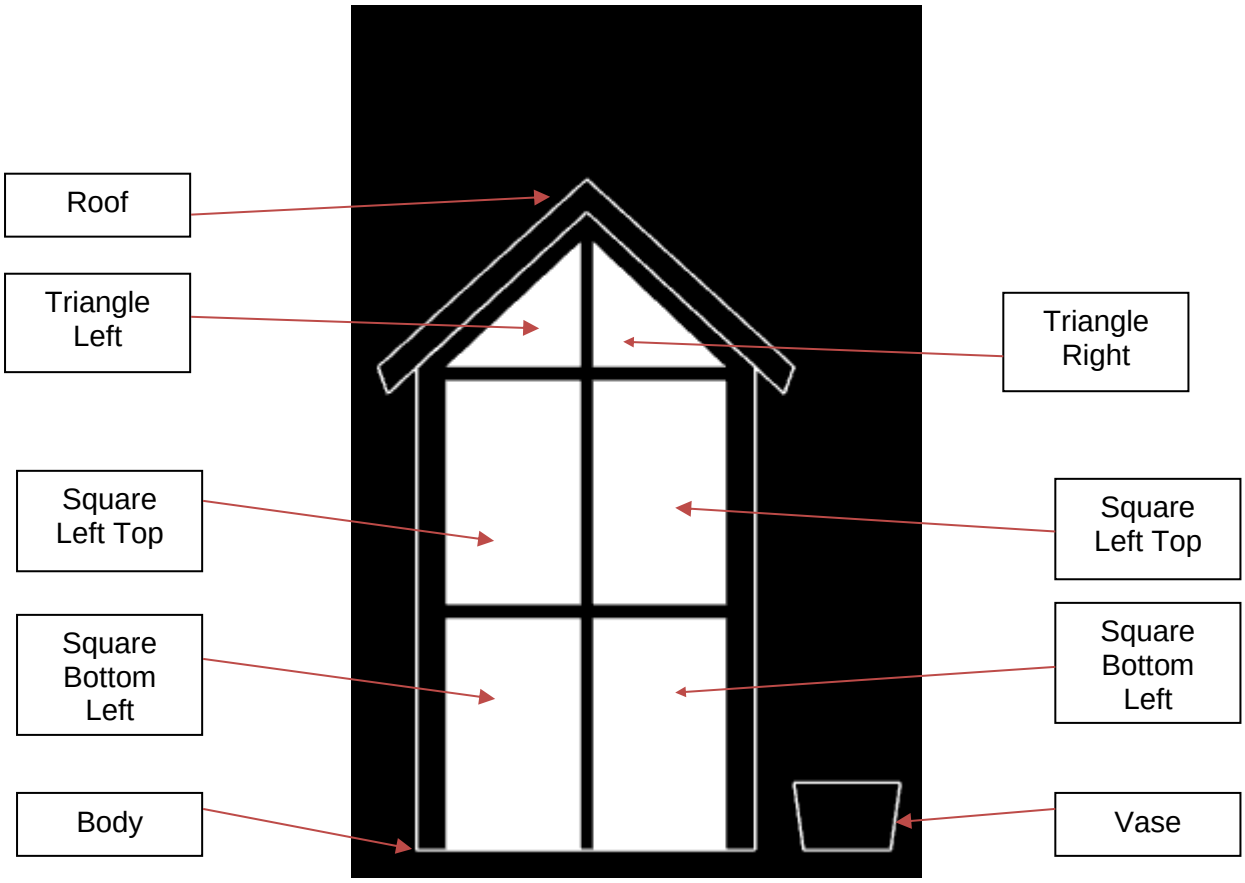
CB21159 MUHAMMAD HAFIZUDDIN BIN HAMSAH
CD19102 MUHAMMAD KHAIRUL ANAM BIN MOHD KHAIRI
CB21144 WAN KHAIRUNNISA BINTI WAN KHAIRUDIN
CB21151 FATIN FAIQAH BINTI YUSOF
CB22072 NUR AMRINA RASYADA BINTI MOHD KHAIRILLAH

SECTION: 01C

SELECTED IMAGE: No. 17



CB21129 MUHAMMAD HAFIZUDDIN BIN HAMSAH



1. Justification of line primitive selection

1.1 Body

I select GL_LINE_LOOP with a vertex count of 5, indicating that mean is required, Therefore, 5 vertices are needed to produce the main body of this house, which is enough because if I use GL_LINE_STRIP then the main body frame will not be completely completed because the coordinates for the last offset (4) will not connect with the first offset (0), and this does not happen on GL_LINE_LOOP because this primitive has the advantage of connecting the last offset to the first.

1.2 Roof Body

I select GL_LINE_LOOP with a vertex count of 6, indicating that mean is required, Therefore, 6 vertices are needed to produce the roof body of this house, which is enough because if I use GL_LINE_STRIP then the roof body frame will not be completely completed because the coordinates for the last offset (32) will not connect with the first offset (27), and this does not happen on GL_LINE_LOOP because this primitive has the advantage of connecting the last offset to the first.

1.3 Tree Vase

I select GL_LINE_LOOP with a vertex count of 4, indicating that mean is required, Therefore, 4 vertices are needed to produce the tree vase of this house, which is enough because if I use GL_LINE_STRIP then the vase tree frame will not be completely completed because the coordinates for the last offset (36) will not connect with the first offset (33), and this does not happen on GL_LINE_LOOP because this primitive has the advantage of connecting the last offset to the first.

2. Justification of face primitive selection

2.1 Triangle Left

I chose GL_TRIANGLE because to generate the triangle that is on the top left of the main body using 3 vertices, it is good to use this primitive from GL_TRIANGLE_STRIP and GL_TRIANGLE_FAN.

2.2 Triangle Right

I chose GL_TRIANGLE because to generate a triangle that is on the top right of the main body using 3 vertices, it is good to use this primitive from GL_TRIANGLE_STRIP and GL_TRIANGLE_FAN.

2.3 Square Left Top

I chose GL_TRIANGLE_STRIP because to create a square on the top left side of the main body that has a surface that requires two triangles that are joined using 4 vertex count, then the most appropriate primitive to use is GL_TRIANGLE_STRIP.

2.4 Square Right Top

I chose GL_TRIANGLE_STRIP because to create a square on the top right side of the main body that has a surface that requires two triangles that are joined using 4 vertex count, then the most appropriate primitive to use is GL_TRIANGLE_STRIP.

2.5 Square Left Bottom

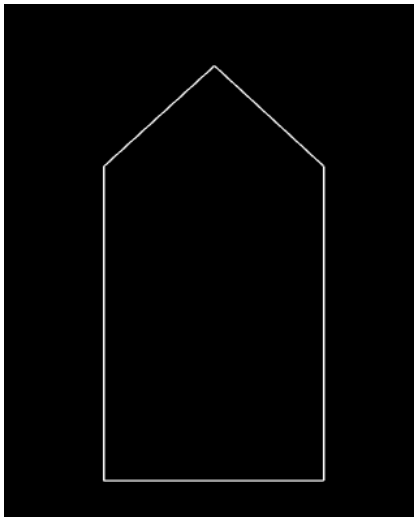
I chose GL_TRIANGLE_STRIP because to create a square on the bottom left side of the main body that has a surface that requires two triangles that are joined using 4 vertex count, then the most appropriate primitive to use is GL_TRIANGLE_STRIP.

2.6 Square Right Bottom

I chose GL_TRIANGLE_STRIP because to create a square on the bottom right side of the main body that has a surface that requires two triangles that are joined using 4 vertex count, then the most appropriate primitive to use is GL_TRIANGLE_STRIP.

3. Explanation of construction process

3.1 Body



For the main body, I started using a line primitive, which is GL_LINE_LOOP with 5 vertices and coordinates:

1. 0.65, -2.0 (lower left)

2. 0.65, 0.5 (upper left)

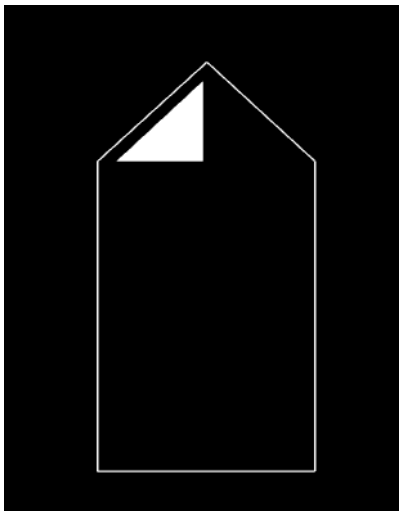
3. 1.53, 1. (middle)

4. 2.4, 0.5 (top right)

5. 2.4, -2.0 (lower right)

Because of the use of GL_LINE_LOOP, automatically the final coordinate (2.4, -2.0) and the initial coordinate (0.65, -2.0) will connect to produce the desired house body in the picture.

3.2 Triangle Left



For the upper left triangle, I used a line primitive, which is GL_TRIANGLE with 3 vertices and coordinates:

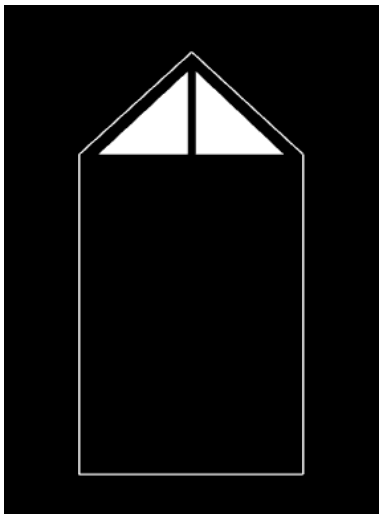
1. 0.8, 0.5 (bottom left)

2. 1.5, 0.5 (lower right)

3. 1.5, 1.15 (top right)

If these three coordinates are connected, then the top left triangle for the body of the house is produced.

3.3 Triangle Right



For the upper right triangle, I used a line primitive, which is GL_TRIANGLE with 3 vertices and coordinates:

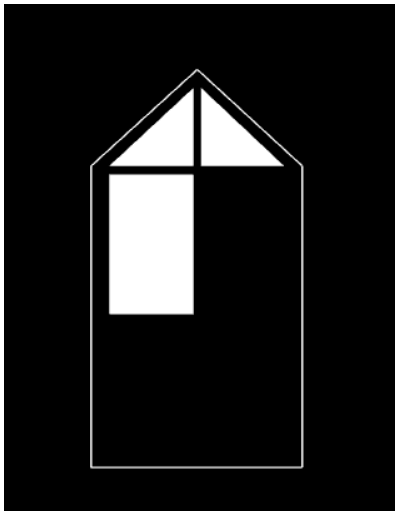
1. 2.25, 0.5 (bottom left)

2. 1.56, 0.5 (lower right)

3. 1.56, 1.15 (top right)

If these three coordinates are connected, then the top right triangle for the body of the house is produced.

3.4 Square Left Top



For the top left square, I use a primitive face, which is GL_TRIANGLE_STRIP with 4 vertices and coordinates:

1. 0.8, 0.43 (top left)

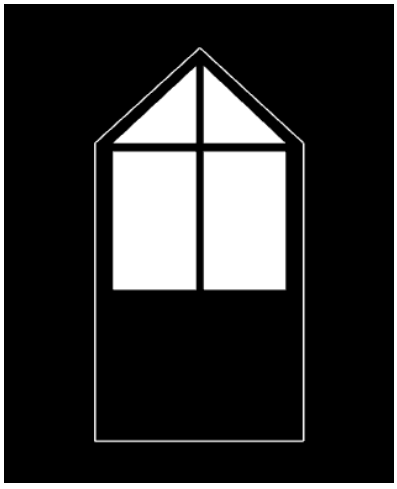
2. 1.5, 0.43 (top right)

3. 0.8, -0.73 (bottom left)

4. 1.5, -0.73 (bottom right)

From these four coordinates, two triangles that form a square will be formed, followed by the top left square for the body house.

3.5 Square Top Right



For the top right square, I used a face primitive, which is GL_TRIANGLE_STRIP

with 4 vertices and coordinates:

1. 1.56, 0.43 (top left)

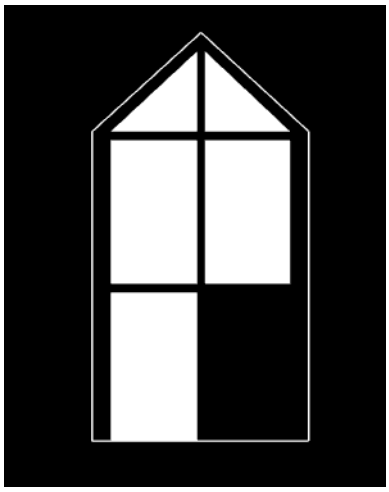
2. 2.25, 0.43 (top right)

3. 1.56, -0.73 (bottom left)

4. 2.25, -0.73 (bottom right)

From these four coordinates, two triangles forming a square will be formed, followed by the top right rectangle for the body house.

3.6 Square Left Bottom



For the lower left square, I used a face primitive, which is GL_TRIANGLE_STRIP with 4 vertices and coordinates:

1. 0.8, -2.0 (bottom left)

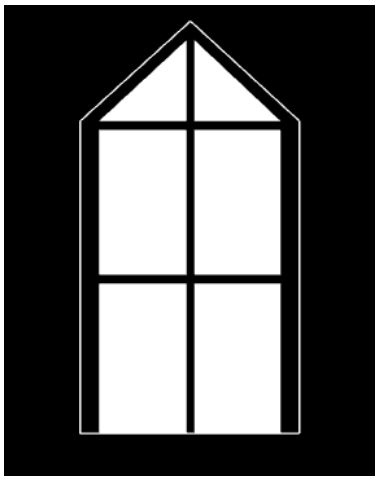
2. 1.5, -2.0 (bottom right)

3. 0.8, -0.8 (top left)

4. 1.5, -0.8 (top right)

From these four coordinates, two triangles forming a square will be formed, followed by a lower left rectangle for the body house.

3.7 Square Right Bottom



For the lower right square, I used a face primitive, which is GL_TRIANGLE_STRIP with 4 vertices and coordinates:

1. 1.56, -2.0 (bottom left)

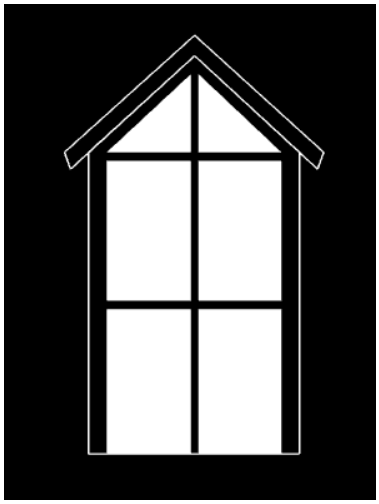
2. 2.25, -2.0 (bottom right)

3. 1.56, -0.8 (top left)

4. 2.25, -0.8 (top right)

From these four coordinates, two triangles forming a square will be formed, followed by a lower right rectangle for the body house.

3.8 Roof Body



For the vase tree, I use a line primitive, which is GL_LINE_LOOP with 6 vertices and coordinates:

1. 0.5, 0.36 (bottom left)

2. 0.45, 0.5 (top left)

3. 1.53, 1.47 (mid top)

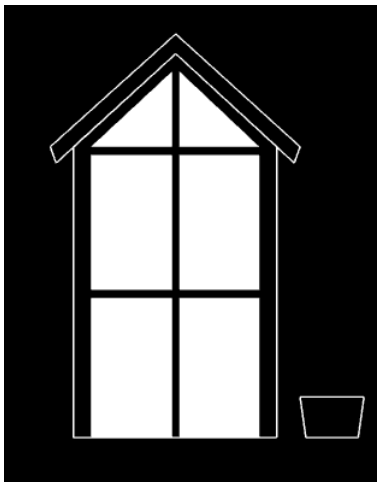
4. 2.6, 0.5 (top right)

5. 2.55, 0.36 (bottom right)

6.1.53, 1.3 (mid bottom)

Because of the use of GL_LINE_LOOP, automatically the final coordinate (1.53, 1.3) and the initial coordinate (0.5, 0.36) will connect to produce the desired roof for the body house, like in the picture.

3.9 Tree Vase



For the vase tree, I use a line primitive, which is GL_LINE_LOOP with 4 vertices and coordinates:

1. 2.65, -2.0 (bottom left)

2. 2.6, -1.65 (top left)

3. 3.15, -1.65 (top right)

4. 3.1, -2.0 (bottom right)

Because of the use of GL_LINE_LOOP, automatically the final coordinate (3.1, -2.0) and the initial coordinate (2.65, -2.0) will connect to produce the desired vase tree in the picture.

4. Justification of vertex counts implementation

```
//Body
primitive = gl.LINE_LOOP;
offset = 0;
vertexCount = 5;
gl.drawArrays(primitive, offset, vertexCount);
```

```
//Roof Body
primitive = gl.LINE_LOOP;
offset = 27;
vertexCount = 6;
gl.drawArrays(primitive, offset, vertexCount);

//Tree Vase - 1
primitive = gl.LINE_LOOP;
offset = 33;
vertexCount = 4;
gl.drawArrays(primitive, offset, vertexCount);
```

For line primitives on the part, I only use GL_LINE_LOOP because this primitive simplifies the work and reduces the number of vertices because the last vertex of each line will connect to the first vertex automatically without vertex count. Total vertex for the primitive line is 15. This is because the shape of my part does not complicate why I use less vertices.

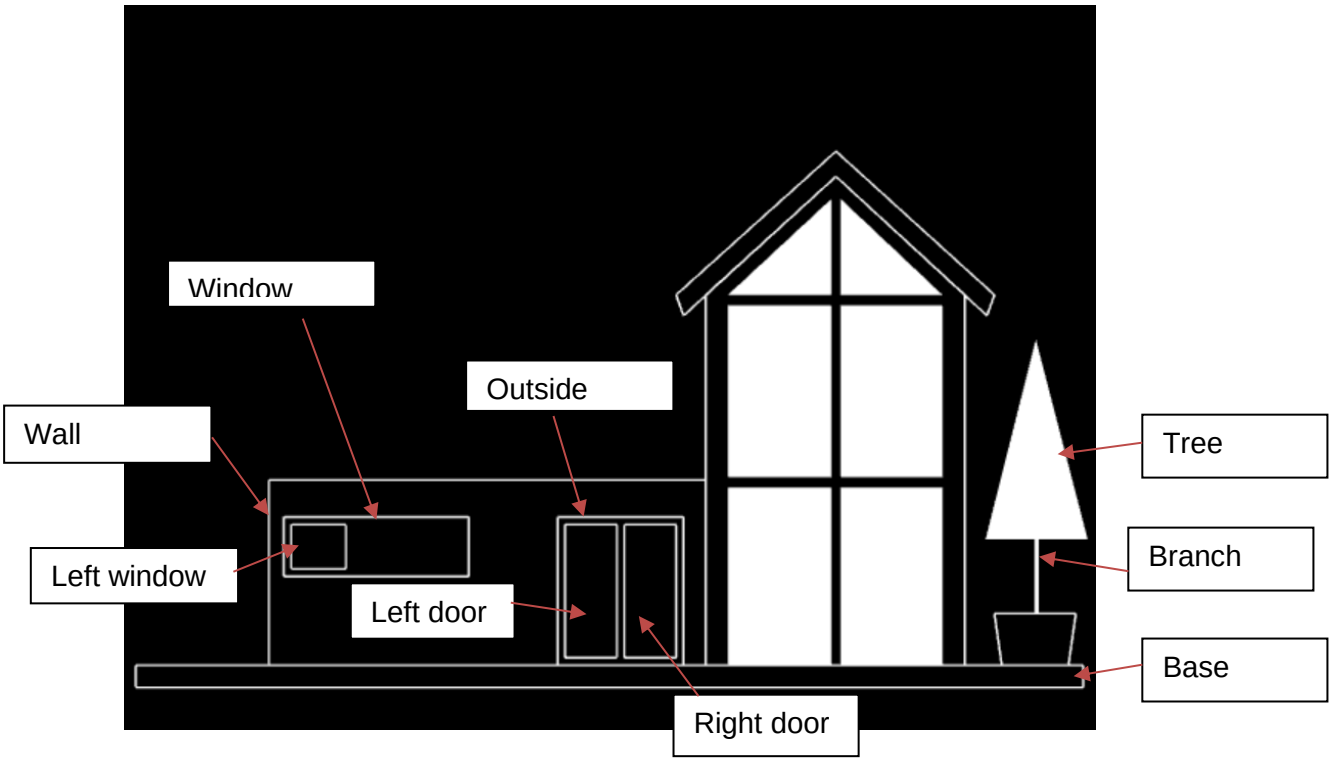
```
//Triangle Left
primitive = gl.TRIANGLES;
offset = 5;
vertexCount = 3;
gl.drawArrays(primitive, offset, vertexCount);

//Triangle Right
primitive = gl.TRIANGLES;
offset = 8;
vertexCount = 3;
gl.drawArrays(primitive, offset, vertexCount);
```

For the face primitive on the part I use GL_TRIANGLE because this primitive is easy to produce a triangle by using only 3 vertices. The number of vertices for a primitive line is 3 vertices. This is because the shape on this part only needs 2 triangles.

```
//Square Left Top
primitive = gl.TRIANGLE_STRIP;
offset = 11;
vertexCount = 4;
gl.drawArrays(primitive, offset, vertexCount);

//Square Right Top
primitive = gl.TRIANGLE_STRIP;
offset = 15;
vertexCount = 4;
gl.drawArrays(primitive, offset, vertexCount);

//Square Left Bottom
primitive = gl.TRIANGLE_STRIP;
offset = 19;
vertexCount = 4;
gl.drawArrays(primitive, offset, vertexCount);

//Square Right Bottom
primitive = gl.TRIANGLE_STRIP;
offset = 23;
vertexCount = 4;
gl.drawArrays(primitive, offset, vertexCount);
```

For the face primitives on this part use GL_TRIANGLE_STRIP because this primitive is easy to produce a rectangle using 2 triangles with only 4 vertices. The number of vertices for a primitive line is 16 vertices. This is because the shape in this section only requires a simple rectangle.

CB21144 WAN KHAIRUNNISA BINTI WAN KHAIRUDIN (01C)



1. Justification of line primitive selection

1.1 Base

I chose GL_LINE_LOOP with only 4 vertices for the base of this house because the last vertex (bottom-right) can connect with the first vertex (bottom-left) automatically. If I use GL_LINE_STRIP, I need to add one more vertex to connect the bottom-right vertex with the bottom-left vertex.

1.2 Wall

I chose GL_LINE_LOOP with only 4 vertices for the wall of this house because the last vertex (bottom-right) can connect with the first vertex (bottom-left) automatically. In this case, GL_LINE_STRIP can also be used, as the bottom-right vertex do not need to connect with the bottom-left vertex as the top-left vertex and top-right vertex of the base have already been connected as a straight line to fulfill the wall shape.

1.3 Outside door

I chose GL_LINE_LOOP with only 4 vertices for the outside door of this house because the last vertex (bottom-right) can connect with the first vertex (bottom-left) automatically. In this case, GL_LINE_STRIP can also be used, as the bottom-right vertex do not need to connect with the bottom-left vertex as the top-left vertex and top-right vertex for the base have already been connected as a straight line to fulfill the wall shape.

1.4 Left door

I chose GL_LINE_LOOP with only 4 vertices for the left door of this house because the last vertex (bottom-right) can connect with the first vertex (bottom-left) automatically. If I use GL_LINE_STRIP, I need to add one more vertex to connect the bottom-right vertex with the bottom-left vertex.

1.5 Right door

I chose GL_LINE_LOOP with only 4 vertices for the right door of this house because the last vertex (bottom-right) can connect with the first vertex (bottom-left) automatically. If I use GL_LINE_STRIP, I need to add one more vertex to connect the bottom-right vertex with the bottom-left vertex.

1.6 Window frame

I chose GL_LINE_LOOP with only 4 vertices for the window frame of this house because the last vertex (bottom-right) can connect with the first vertex (bottom-left) automatically. If I use GL_LINE_STRIP, I need to add one more vertex to connect the bottom-right vertex with the bottom-left vertex.

1.7 Left window

I chose GL_LINE_LOOP with only 4 vertices for the left window of this house because the last vertex (bottom-right) can connect with the first vertex (bottom-left) automatically. If I use GL_LINE_STRIP, I need to add one more vertex to connect the bottom-right vertex with the bottom-left vertex.

2. Justification of face primitive selection

2.1 Branch

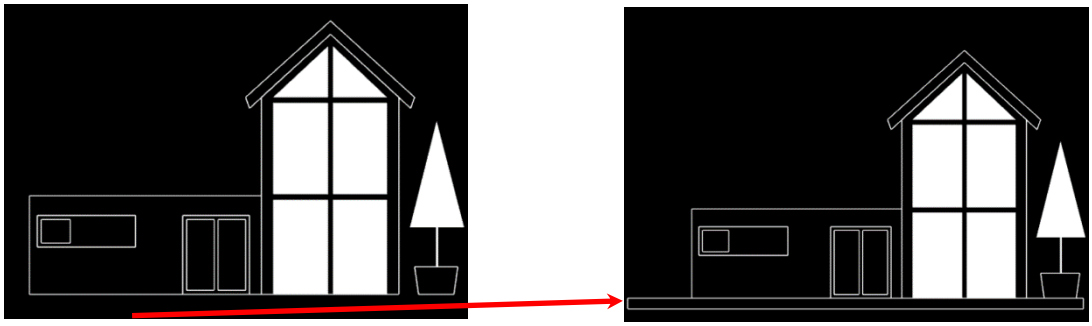
I chose GL_TRIANGLE_STRIP with only 4 vertices for the branch of this tree rather than using GL_LINE_STRIP because I wanted to make the branch look thicker to fit the size of the tree shape. So, when using GL_TRIANGLE_STRIP with 4 vertices, it will produce a rectangle with the bottom-left and bottom-right vertices connected with the top-left and top-right vertices to make two connected triangles to display the thicker branch.

2.2 Tree

I chose GL_TRIANGLES with only 3 vertices for the tree because the shape of the tree is already a triangle. So, I just specified the bottom-left, bottom-right and top vertices to compose a triangle for the tree. In this case, GL_TRIANGLE_STRIP or GL_TRIANGLE_FAN is not suitable as the tree requires only 1 connected triangle.

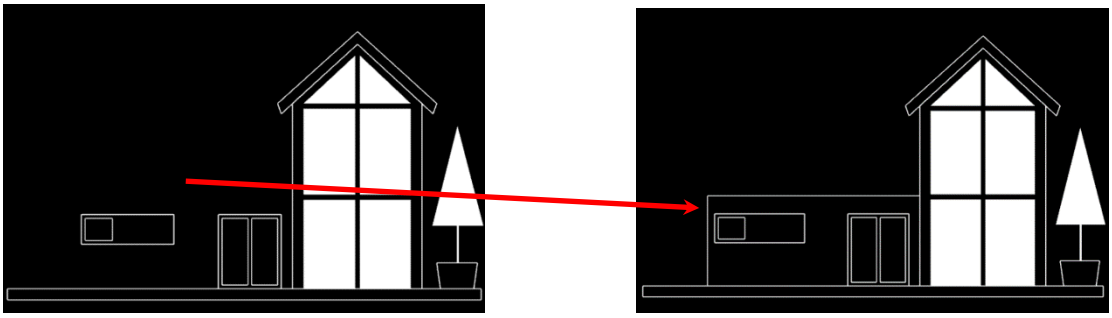
3. Explanation of construction process

3.1 Base



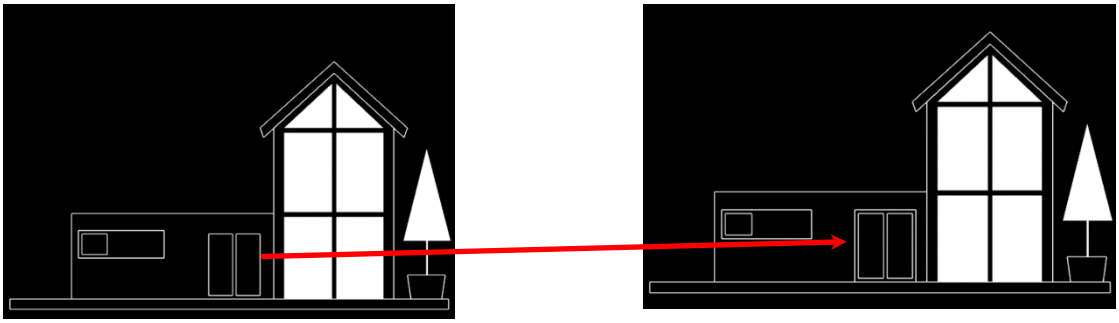
For the line primitive using GL_LINE_LOOP with 4 vertices, I started from bottom-left vertex (-3.2, -2.15), then go to the top-left vertex (-3.2, -2.0) and connected with the top-right vertex (3.2, -2.0) and finally the bottom-right vertex (3.2, -2.15) will connect automatically with bottom-left vertex as I use the GL_LINE_LOOP.

3.2 Wall



For the line primitive using GL_LINE_LOOP with 4 vertices, I started from bottom-left vertex (-2.3, -2.0), then go to the top-left vertex (-2.3, -0.75,) and connected with the top-right vertex (0.65, -0.75) and finally the bottom-right vertex (0.65, -2.0,) will connect automatically with bottom-left vertex as I use the GL_LINE_LOOP.

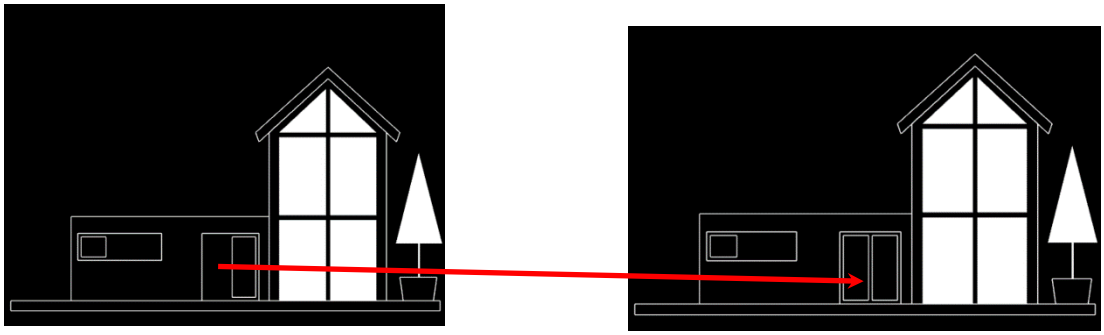
3.3 Outside door



For the line primitive using GL_LINE_LOOP with 4 vertices, I started from

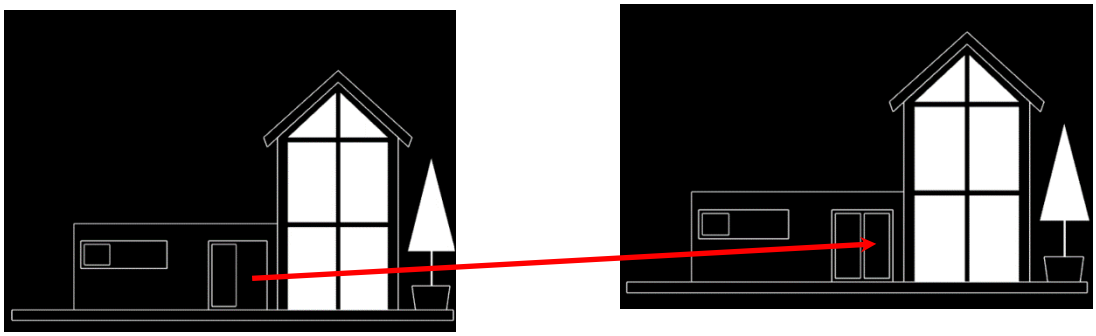
bottom-left vertex (-0.35, -2.0), then go to the top-left vertex (-0.35, -1.0,) and connected with the top-right vertex (0.5, -1.0) and finally the bottom-right vertex (0.5, -2.0) will connect automatically with bottom-left vertex as I use the GL_LINE_LOOP.

3.4 Left door



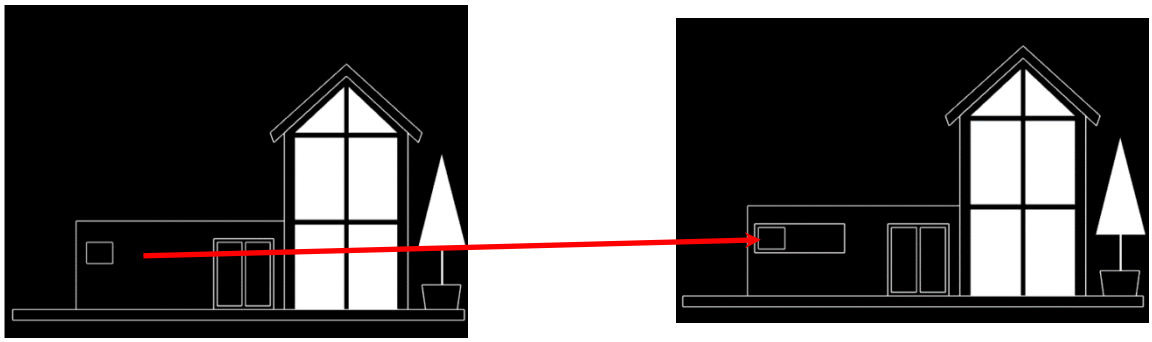
For the line primitive using GL_LINE_LOOP with 4 vertices, I started from bottom-left vertex (-0.3, -1.95), then go to the top-left vertex (-0.3, -1.05,) and connected with the top-right vertex (0.05, -1.05) and finally the bottom-right vertex (0.05, -1.95) will connect automatically with bottom-left vertex as I use the GL_LINE_LOOP.

3.5 Right door



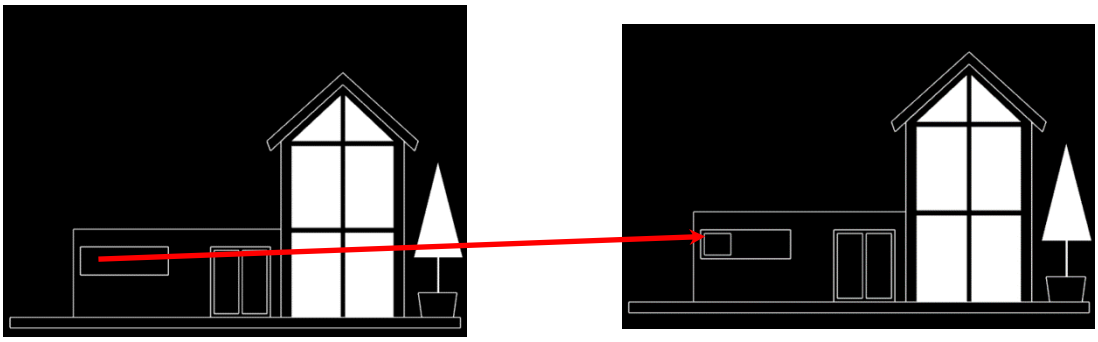
For the line primitive using GL_LINE_LOOP with 4 vertices, I started from bottom-left vertex (0.1, -1.95), then go to the top-left vertex (0.1, -1.05,) and connected with the top-right vertex (0.45, -1.05) and finally the bottom-right vertex (0.45, -1.95) will connect automatically with bottom-left vertex as I use the GL_LINE_LOOP.

3.6 Window frame



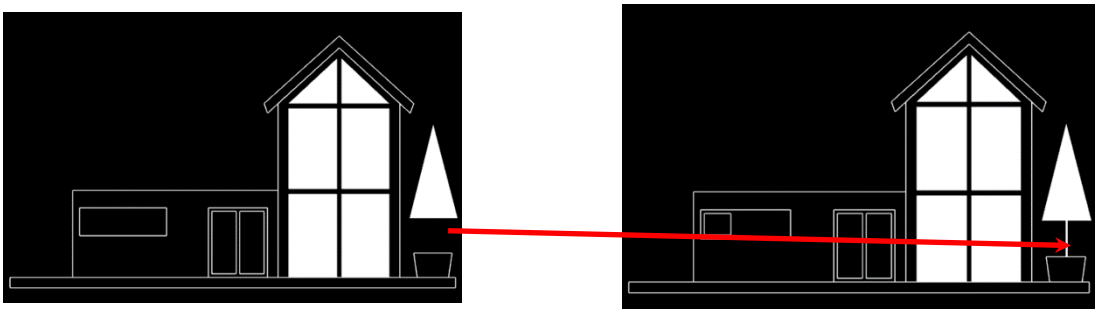
For the line primitive using GL_LINE_LOOP with 4 vertices, I started from bottom-left vertex (-2.2, -1.4), then go to the top-left vertex (-2.2, -1.0,) and connected with the top-right vertex (-0.95, -1.0) and finally the bottom-right vertex (-0.95, -1.4) will connect automatically with bottom-left vertex as I use the GL_LINE_LOOP.

3.7 Left window



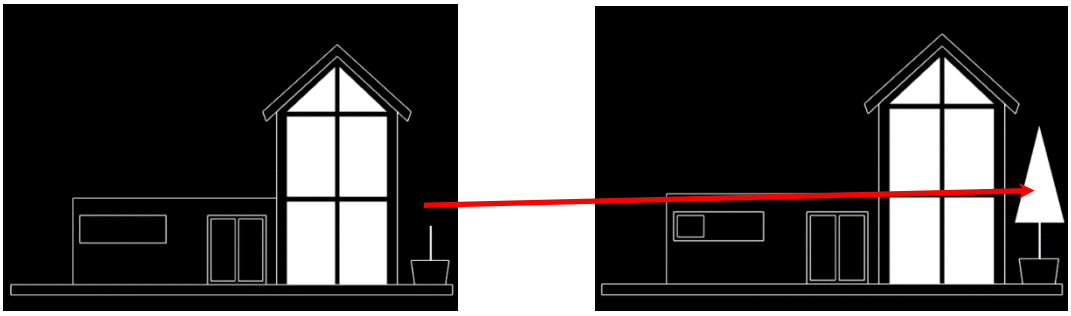
For the line primitive using GL_LINE_LOOP with 4 vertices, I started from bottom-left vertex (-2.15, -1.35), then go to the top-left vertex (-2.15, -1.05,) and connected with the top-right vertex (-1.78, -1.05) and finally the bottom-right vertex -1.78, -1.35) will connect automatically with bottom-left vertex as I use the GL_LINE_LOOP.

3.8 Branch



For the face primitive using GL_TRIANGLE_STRIP with only 4 vertices, I am using bottom-left vertex (2.87, -1.65) as a focal point, and from there I plot the next coordinate at bottom-right vertex (2.9, -1.65) along with the top-left vertex (2.87, -1.15) and lastly the top-right vertex (2.9, -1.15) to make two connected triangles.

3.9 Tree



For the face primitive using GL_TRIANGLE with only 3 vertices, I am using the bottom-left vertex (2.54, -1.15) as a focal point, and from there I plot the next coordinate at the bottom-right vertex (3.23, -1.15) and lastly the top vertex (2.88, 0.2) to compose a triangle for the tree.

4. Justification of vertex counts implementation

```
//Base
primitive = gl.LINE_LOOP;
offset = 44;
vertexCount = 4;
gl.drawArrays(primitive, offset, vertexCount);

//Wall (bottom left)
primitive = gl.LINE_LOOP;
offset = 48;
vertexCount = 4;
gl.drawArrays(primitive, offset, vertexCount);

//Outside Door (bottom left)
primitive = gl.LINE_LOOP;
offset = 52;
vertexCount = 4;
gl.drawArrays(primitive, offset, vertexCount);

//door - 1 (left)
primitive = gl.LINE_LOOP;
offset = 56;
vertexCount = 4;
gl.drawArrays(primitive, offset, vertexCount);
```

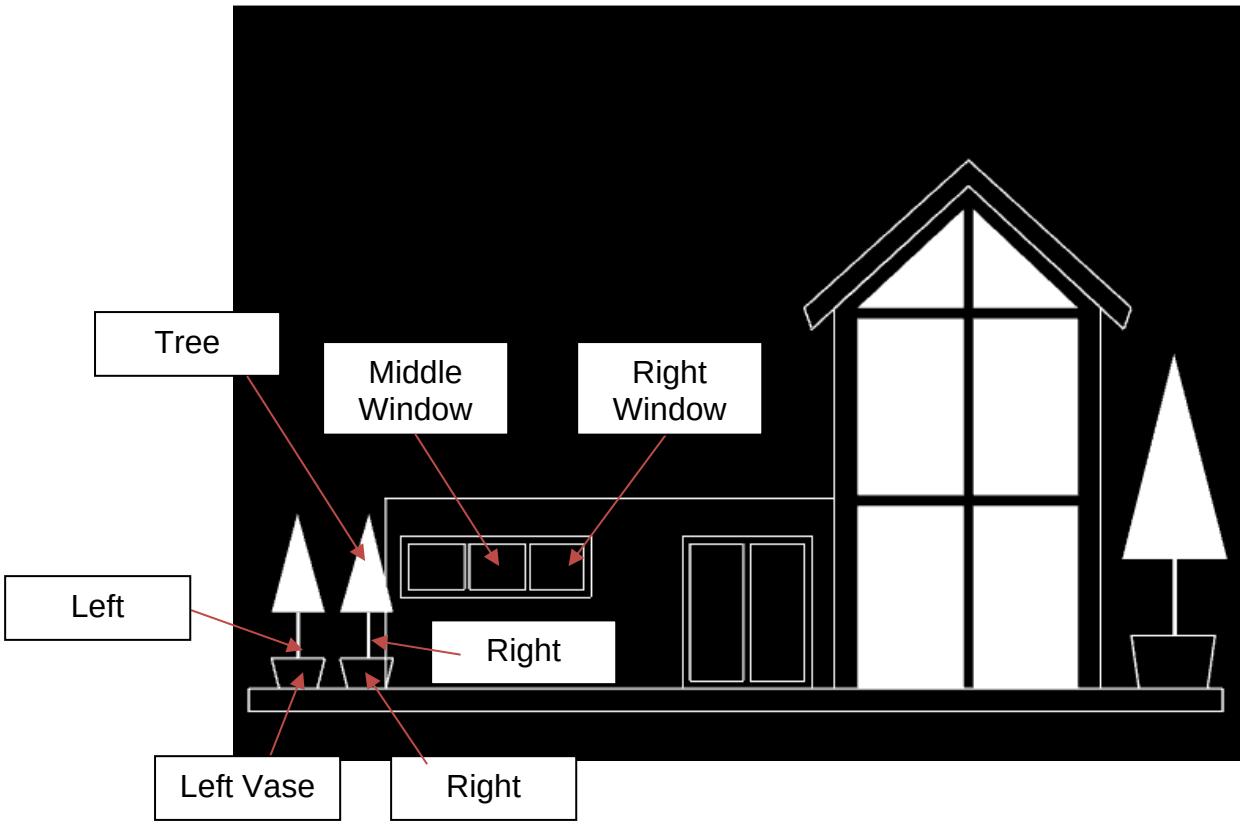
```
//Door - 2 (right)
primitive = gl.LINE_LOOP;
offset = 60;
vertexCount = 4;
gl.drawArrays(primitive, offset, vertexCount);

//Windows frame (bottom left)
primitive = gl.LINE_LOOP;
offset = 64;
vertexCount = 4;
gl.drawArrays(primitive, offset, vertexCount);

//Windows - 1 (left)
primitive = gl.LINE_LOOP;
offset = 68;
vertexCount = 4;
gl.drawArrays(primitive, offset, vertexCount);
```

Total vertex counts for line primitive using GL_LINE_LOOP is 28 vertices. This is because if I use fewer vertices, the line would not be connected to the last vertex to complete the shape for each part of the house. Also, using the GL_LINE_LOOP primitive has already reduced the number of vertices as the last vertex for each line will connect with the first vertex automatically without the vertex count.

```
//Branch
primitive = gl.TRIANGLE_STRIP;
offset = 37;
vertexCount = 4;
gl.drawArrays(primitive, offset, vertexCount);

//Tree
primitive = gl.TRIANGLES;
offset = 41;
vertexCount = 3;
gl.drawArrays(primitive, offset, vertexCount);
```

Total vertex counts for face primitives using GL_TRIANGLE_STRIP and GL_TRIANGLE is 7 vertices. If I use fewer than four vertices on the GL_TRIANGLE_STRIP primitive for the branch, it cannot produce a rectangle to create two connected triangles to display the tree's thicker branch. While using lesser vertices (less than 3) on GL_TRIANGLES, it cannot compose a triangle for the tree as the triangle shape is required to have 3 vertices.

CB21151 FATIN FAIQAH BINTI YUSOF (01C)



1. Justification of line primitive selection

1.1 Middle Window

I chose GL_LINE_LOOP because the middle window have 4 vertex and wanted to connect the start point to the last point easily to create the shape of a square rather than use GL_LINE_STRIP that will not connect the lines except add one more vertex to connecting the point.

1.2 Right Window

I chose GL_LINE_LOOP because the right window has 4 vertices and wanted to connect the start point to the last point easily to create the shape of a square rather than use GL_LINE_STRIP that will not connect the lines except add one more vertex to connect the point.

1.3 Left Vase & Right Vase

I chose GL_LINE_LOOP because it is easy to use lines to create the shape of the vase and use the loop to connect the first point to the last point. Need 4 vertices to create the vase.

1.4 Left & Right Branch

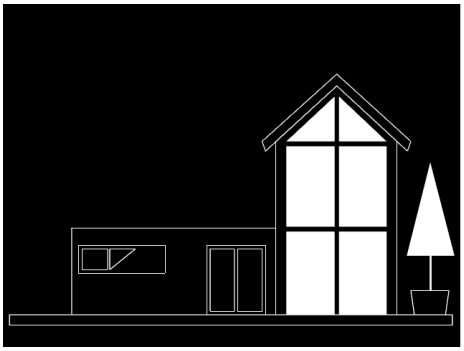
I choose GL_LINE_STRIP because to create the double straight line by using 2

vertices for both branches but with different numbers of positions.

2. **Justification of face primitive selection**

   I choose GL_TRIANGLES for the tree which is in the shape of triangles and just need to set the position using only 3 vertices.

3. **Explanation of construction process**



For the line primitive using GL_LINE_LOOP, I started from bottom-left vertex, then go to the top-left continue with straight line to top-right and connect the loop on bottom-right vertex to create the window. Ensure the using 4 vertices to have a perfect shape of window.

For the line primitive using GL_LINE_STRIP, to create double straight line and have thick branch I started from bottom left vertex, then go to top-left and one straight

will be created by using 2 vertices and do the same process with a different position which is next to each other to create the thick straight line with started from bottom-right vertex, then go to top-right and using 2 vertices

For the face primitive using GL_TRIANGLES, to create the shape of the tree the shape of a triangle by using this face primitive is the best choice. I started from bottom-left then went to bottom-right and set the position of the top vertex to create the perfect shape of the tree. The vertex count for the tree is 3.

4. Justification of vertex counts implementation

```
//Windows - 2 (middle)
primitive = gl.LINE_LOOP;
offset = 72;
vertexCount = 4;
gl.drawArrays(primitive, offset, vertexCount);

//Windows - 3 (right)
primitive = gl.LINE_LOOP;
offset = 76;
vertexCount = 4;
gl.drawArrays(primitive, offset, vertexCount);

//Tree vase
primitive = gl.LINE_LOOP;
offset = 80;
vertexCount = 4;
gl.drawArrays(primitive, offset, vertexCount);

//Tree vase 2
primitive = gl.LINE_LOOP;
offset = 91;
vertexCount = 4;
gl.drawArrays(primitive, offset, vertexCount);
```

- Total vertex counts for line primitive using GL_LINE_LOOP is 16 vertices. This is because if I use fewer vertices, the line would be in unwanted shape because I choose the line loop which will connect the first vertex to the last vertex after deciding the position of the vertex.

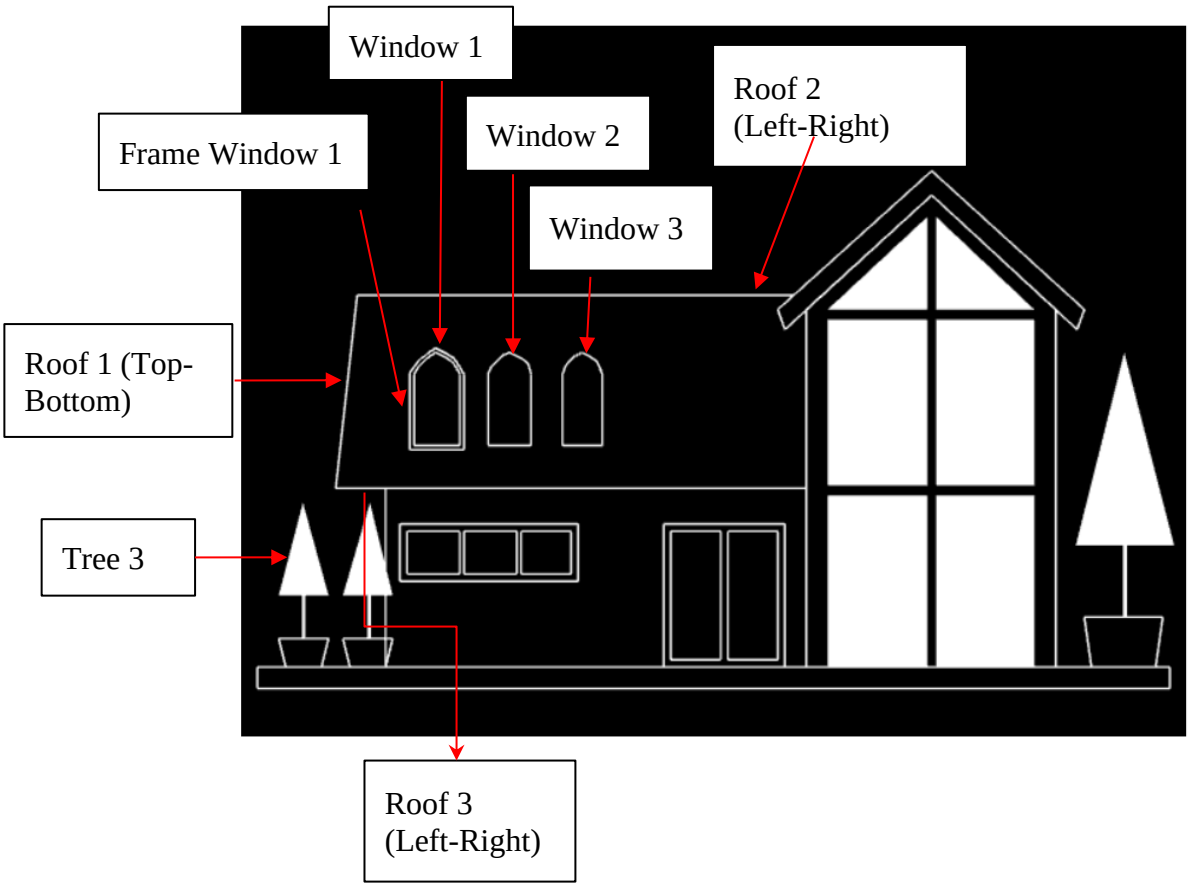
```
//Branch - 1
 primitive = gl.LINE_STRIP;
 offset = 84;
 vertexCount = 2;
 gl.drawArrays(primitive, offset, vertexCount);

 //Branch - 2
 primitive = gl.LINE_STRIP;
 offset = 86;
 vertexCount = 2;
 gl.drawArrays(primitive, offset, vertexCount);

 //Branch - 1
 primitive = gl.LINE_STRIP;
 offset = 95;
 vertexCount = 2;
 gl.drawArrays(primitive, offset, vertexCount);

 //Branch - 2
 primitive = gl.LINE_STRIP;
 offset = 97;
 vertexCount = 2;
 gl.drawArrays(primitive, offset, vertexCount);
```

- Total vertex counts for line primitive using GL_LINE_STRIP is 8 vertices. This is because if I use fewer vertices, the line would not be shown or can't see the line because it needs to have 2 vertices to have a straight line to create the branch. It has 2 lines for each branch to create the thick line which is placed side by side.

```
        //Tree
        primitive = gl.TRIANGLES;
        offset = 88;
        vertexCount = 3;
        gl.drawArrays(primitive, offset, vertexCount);
```

- Total vertex counts for face primitives using GL_TRIANGLES are 3 vertices. Because I used GL_TRIANGLES primitive, it only needed 3 vertices to design the shape of the tree perfectly.

CB22072 NUR AMRINA RASYADA BINTI MOHD KHAIRILLAH (01C)



1. Justification of line primitive selection

1.1 Roof 1 (Top-Bottom)

I choose GL_LINES to draw a line from top to bottom of the roof because it draws disconnected line segments which is only need 2 vertices to consider a line. The first and the last vertex are not connected to any other vertex. In this case, the roof is not connected with the wall part. Besides, if a non-even number of vertices is specified, then the extra vertex is ignored.

1.2 Roof 2 (Left- Right)

I choose GL_LINES to draw a line from left to right of the roof because it draws disconnected line segments. I only need 2 vertices because the first and the last vertex are not connected to any other vertex. Between 2 vertex is considered a line. Besides, if a non-even number of vertices is specified, then the extra vertex is ignored.

1.3 Roof 3 (Left-Right)

I choose GL_LINES to draw a line from left to right of the roof because it draws disconnected line segments. I only need 2 vertices because the first and the last vertex are not connected to any other vertex. Between 2 vertex is considered a line. Besides, if a non-even number of vertices is specified, then

the extra vertex is ignored.

1.4 Window 1

I choose GL_LINE_STRIP because it is easier to make the windows curve form well. It needs to plot one by one of the vertices to create a connected line. Even though the first vertex is not connected to the last vertex but one more line is needed to complete it as a window shape. The adjacent vertices are considered lines. Thus, if I pass n vertices, I will get n-1 lines. If I only specify 1 vertex, the drawing command is ignored.

1.5 Window 2

I choose GL_LINE_STRIP because it is easier to make the windows curve form well. It needs to plot one by one of the vertices to create a connected line. Even though the first vertex is not connected to the last vertex but one more line is needed to complete it as a window shape. The adjacent vertices are considered lines. Thus, if I pass n vertices, I will get n-1 lines. If I only specify 1 vertex, the drawing command is ignored.

1.6 Window 3

I choose GL_LINE_STRIP because it is easier to make the windows curve form well. It needs to plot one by one of the vertices to create a connected line. Even though the first vertex is not connected to the last vertex but one more line is needed to complete it as a window shape. The adjacent vertices are considered lines. Thus, if I pass n vertices, I will get n-1 lines. If I only specify 1 vertex, the drawing command is ignored.

1.7 Frame Window 1

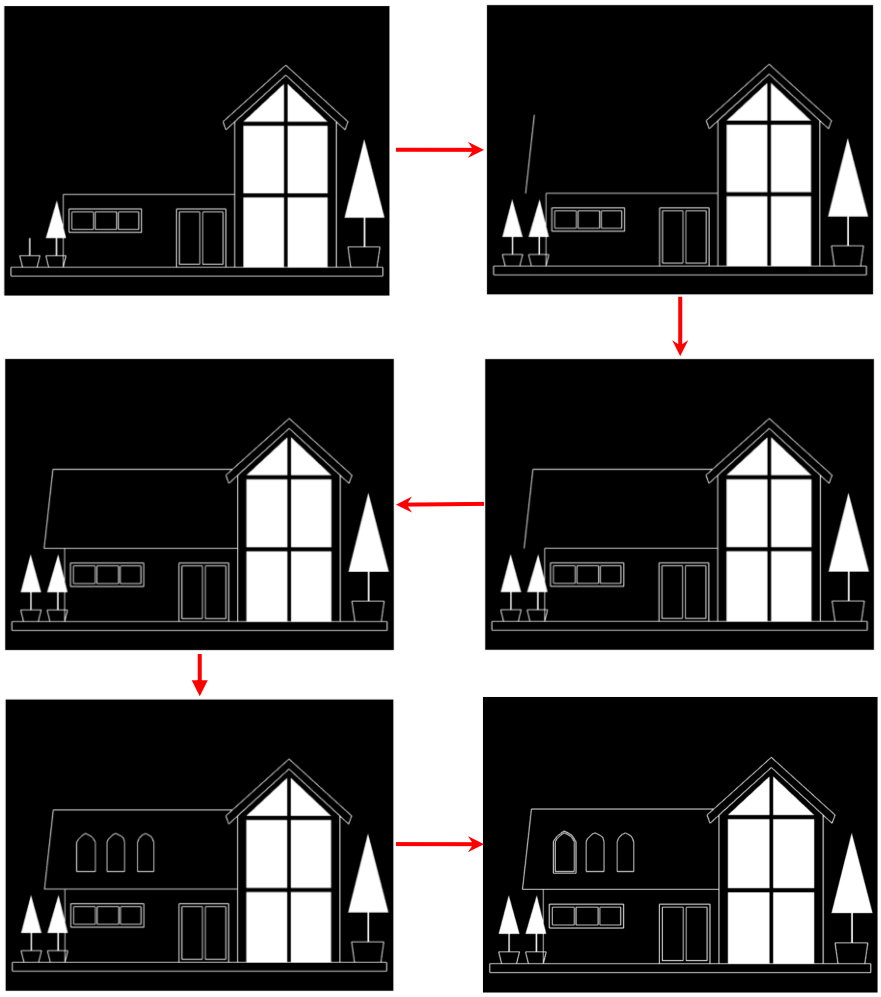
I choose GL_LINE_STRIP because it is easier to make the windows curve form well. It needs to plot one by one of the vertices to create a connected line. Even though the first vertex is not connected to the last vertex but one more line is needed to complete it as a window shape. The adjacent vertices are considered lines. Thus, if I pass n vertices, I will get n-1 lines. If I only specify 1 vertex, the drawing command is ignored.

2. Justification of face primitive selection

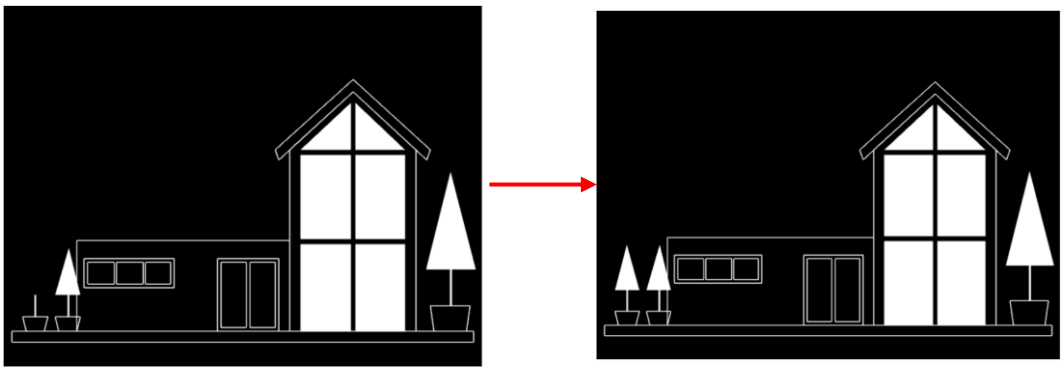
2.1 Tree

I choose GL_TRIANGLES because the tree only needs 3 vertices to make one triangle. I just need to specify the bottom-left, bottom-right and top vertices.

3. Explanation of construction process



For the line primitive using GL_LINES for the roof while GL_LINE_STRIP for windows, I started from bottom-left vertex (-2.65, -0.75) to top-left vertex (-2.5, 0.6), then the other two lines I started from left to right which are the first top line is (-2.5, 0.6) to (0.56, 0.6) and (-2.65, -0.75) to (-2.3, -0.75). For the windows, I started the first vertex from bottom-right (-1.78, -0.45) until top-right (-1.78, -0.45) vertex which is connected with the first vertex, same way for the frame window 1.



For the face primitive using GL_TRIANGLES, I am using bottom-left (-3.05, -1.5) vertex as a focal point, and from there I plot the next coordinate at bottom-right (-2.7, -1.5) and finally, the last one is at top (-2.88, -0.85) vertex to make a perfect triangle for tree.

4. Justification of vertex counts implementation

```
//Roof
primitive = gl.LINES;
offset = 102;
vertexCount = 2;
gl.drawArrays(primitive, offset, vertexCount);

//Roof 1
primitive = gl.LINES;
offset = 104;
vertexCount = 2;
gl.drawArrays(primitive, offset, vertexCount);

//Roof 2
primitive = gl.LINES;
offset = 106;
vertexCount = 2;
gl.drawArrays(primitive, offset, vertexCount);
```

Total vertex counts for line primitive using GL_LINES is 8 vertices. This is because 2 vertexes are already enough to produce a line and much easier to draw a roof which only needs 3 lines that are not connected with the wall lines. If I use more vertices, it couldn't make a perfectly straight line for the roof.

```
//Window 1
primitive = gl.LINE_STRIP;
offset = 108;
vertexCount = 15;
gl.drawArrays(primitive, offset, vertexCount);

//Window 2
primitive = gl.LINE_STRIP;
offset = 123;
vertexCount = 15;
gl.drawArrays(primitive, offset, vertexCount);

//Window 3
primitive = gl.LINE_STRIP;
offset = 138;
vertexCount = 13;
gl.drawArrays(primitive, offset, vertexCount);

//Frame window 1
primitive = gl.LINE_STRIP;
offset = 151;
vertexCount = 15;
gl.drawArrays(primitive, offset, vertexCount);
```

Total vertex counts for line primitive using GL_LINE_STRIP is 58 vertices. This is because if I use fewer vertices, each curve for the window cannot be formed

properly since framing each curve requires a different number of vertices according to the appropriate situations.

```
//Tree
primitive = gl.TRIANGLES;
offset = 99;
vertexCount = 3;
gl.drawArrays(primitive, offset, vertexCount);
```

Total vertex counts for face primitive using GL_TRIANGLES is 3 vertices. Only 3 coordinates are needed to form a perfect triangle for the tree. If I use more than 3, the other vertices will be ignored.

CD19102 MUHAMMAD KHAIRUL ANAM BIN MOHD KHAIRI (01C)

1. Justification of line primitive selection

- I choose GL_LINE_STRIP for frame window 2 & 3. This is because it is easier to make the curve at the top of the window.
- I choose GL_LINES for the branch inside the windows because it is just a simple line which is a vertical and horizontal line.

2. Justification of face primitive selection

- No face primitive for my part.

3. Explanation of construction process

- For the line primitive of frame windows, I started from the glass left window. I started from bottom left, then went up to the top left and plotted a few coordinates to make a curved line at the top and then went back down as the same y-coordinates bottom left. I used the same method to plot all the other 2 windows.
- For the line primitive of the branch inside the window, I use the same line primitive, and I started from the bottom to the top for the vertical branch and for the horizontal line, I started from the left to the right. I used the same method to plot all the other 2 windows.

4. Justification of vertex counts implementation

```
576         //Muhammad Khairul Anam Bin Mohd Khairi CD19102
577         //Frame window 2
578         primitive = gl.LINE_STRIP;
579         offset = 166;
580         vertexCount = 15;
581         gl.drawArrays(primitive, offset, vertexCount);
582
583         //Frame window 3
584         primitive = gl.LINE_STRIP;
585         offset = 181;
586         vertexCount = 13;
587         gl.drawArrays(primitive, offset, vertexCount);
588
589         //Inside window 1 (vertical)
590         primitive = gl.LINES;
591         offset = 194;
592         vertexCount = 2;
593         gl.drawArrays(primitive, offset, vertexCount);
594
595         //Inside window 1 (horizontal)
596         primitive = gl.LINES;
597         offset = 196;
598         vertexCount = 2;
599         gl.drawArrays(primitive, offset, vertexCount);
600
601         //Inside window 2 (vertical)
602         primitive = gl.LINES;
603         offset = 198;
604         vertexCount = 2;
605         gl.drawArrays(primitive, offset, vertexCount);
606
```

```
607         //Inside window 2 (horizontal)
608         primitive = gl.LINES;
609         offset = 200;
610         vertexCount = 2;
611         gl.drawArrays(primitive, offset, vertexCount);
612
613         //Inside window 3 (vertical)
614         primitive = gl.LINES;
615         offset = 202;
616         vertexCount = 2;
617         gl.drawArrays(primitive, offset, vertexCount);
618
619         //Inside window 3 (horizontal)
620         primitive = gl.LINES;
621         offset = 204;
622         vertexCount = 2;
623         gl.drawArrays(primitive, offset, vertexCount);
624
625     }
```

- Total vertex count for the line primitive is 40 vertices. All the vertices are used when plotting the windows curve since it requires a lot of coordinates to make a well-shaped curve.