1. make Translation, rotation, scale.

I try to implement but I failed, the slider will freeze and I try many ways to fix it, but still not work. Here's the process I've done in these weeks.

> code snippets

I followed the reasoning I learned in class. What I learned in class was a 3x3 translation matrix in the x and y directions, but now the implementation involves a 4x4 matrix in the x, y, and z directions. So I wrote the following code.

```
(1) makeTrans()
m[0] = 1.0f ;  m[1]  = 0.0f ; m[2]  = 0.0f;  m[3]  = s.x;
m[4] = 0.0f ;  m[5]  = 1.0f ; m[6]  = 0.0f;  m[7]  = s.y;
m[8] = 0.0f ;  m[9]  = 0.0f ; m[10] = 1.0f;  m[11] = s.z;
m[12] = 0.0f ; m[13] = 0.0f ; m[14] = 0.0f;  m[15] = 1.0f;

(2) makeRotZ()
m[0] = cos(a) ; m[1]  = -sin(a) ; m[2]  = 0.0f;  m[3]  = 0.0f;
m[4] = sin(a) ; m[5]  = cos(a) ;  m[6]  = 0.0f;  m[7]  = 0.0f;
m[8] = 0.0f ;   m[9]  = 0.0f ;    m[10] = 1.0f;  m[11] = 0.0f;
m[12] = 0.0f ;  m[13] = 0.0f ;    m[14] = 0.0f;  m[15] = 1.0f;

(3) makeScale()
m[0] = s.x ;   m[1]  = 0.0f ; m[2]  = 0.0f;  m[3]  = 0.0f;
m[4] = 0.0f ;  m[5]  = s.y ;  m[6]  = 0.0f;  m[7]  = 0.0f;
m[8] = 0.0f ;  m[9]  = 0.0f ; m[10] = s.z;   m[11] = 0.0f;
m[12] = 0.0f ; m[13] = 0.0f ; m[14] = 0.0f;  m[15] = 1.0f;
```

But I found that the slider freeze, so I ask ChatGPT, he think I can add some debug message.

- add **try** and **catch** in 3 function.
- add message in **Slider.click()** to check the mouse is inside or not.

Here's the change I've done and the console message I get.

- matrix part

```
void makeTrans(Vector3 t) {
  // TODO HW2
  // You need to implement the translate matrix here.
     try {
        makeIdentity();
         m[3] = t.x;
         m[7] = t.y;
         m[11] = t.z;
         m[15] = 1.0f;

        println("Translation matrix created for: " + t.x + ", " + t.y + ", " + t.z);
        println(this.toString());
     } catch (Exception e) {
        println("Error in makeTrans: " + e.getMessage());
     }
}
```

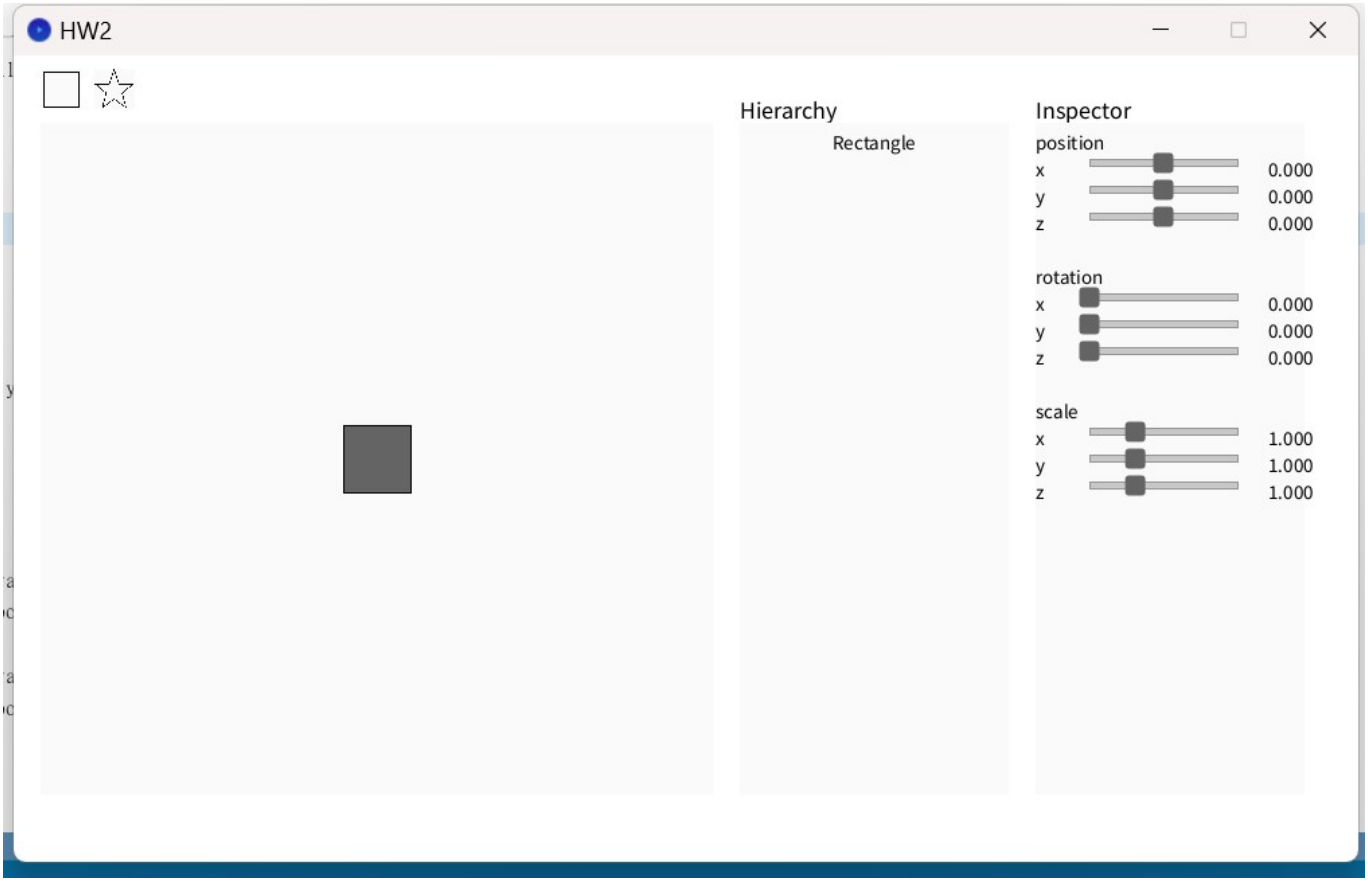
- slider part

```
public void click() {
println("Slider click called");
println("checkInSide(): " + checkInSide());
println("press: " + press);

if (!checkInSide() && !press) {
    println("Early return");
    return;
}

if (mousePressed) {
    println("Mouse is pressed");
    press = true;
    float oldValue = value();

    if (vertical) {
        float newX = constrain(mouseX, start_end_point.x, start_end_point.y);
        println("Vertical slider: trying to move from " + pos.x + " to " + newX);
        pos.x = newX;
    } else {
        float newY = constrain(mouseY, start_end_point.x, start_end_point.y);
        println("Horizontal slider: trying to move from " + pos.y + " to " + newY);
        pos.y = newY;
    }

    println("Value changed from " + oldValue + " to " + value());
} else {
    println("Mouse released");
```

- debug message

```
Slider click called
checkInSide(): false
press: false
Early return
Translation matrix created for: 0.21818185, 0.0, 0.0
1.0 0.0 0.0 0.21818185
0.0 1.0 0.0 0.0
0.0 0.0 1.0 0.0
0.0 0.0 0.0 1.0
```

From the result, I can see that The slider detected the click, but thought that the mouse was not within the range of the slider, so it returned early. And the matrix does change, but the small button inside the slider still can't move. So I try to make the button bigger, and the slider thinner, like this :



Is the same result, it does not work. Then I go **back to the matrix** I wrote because the slider freeze after I implement those 3 function, I decided to write the result of multiplying the three matrices in **localWorld** to test it. Here's the original one :

```
public Matrix4 localToWorld(){
    return Matrix4.Trans(transform.position).mult(Matrix4.RotZ(transform.rotation.z)).mult(Matrix4.Scale(transform.scale));
}
```

I change it like this :

```
Matrix4 finalMatrix = new Matrix4();

finalMatrix.m[0] = s.x * cos(theta);
finalMatrix.m[1] = -s.y * sin(theta);
finalMatrix.m[2] = 0;
finalMatrix.m[3] = t.x;

finalMatrix.m[4] = s.x * sin(theta);
finalMatrix.m[5] = s.y * cos(theta);
finalMatrix.m[6] = 0;
finalMatrix.m[7] = t.y;

finalMatrix.m[8] = 0;
finalMatrix.m[9] = 0;
finalMatrix.m[10] = s.z;
finalMatrix.m[11] = t.z;

finalMatrix.m[12] = 0;
finalMatrix.m[13] = 0;
```

```
    finalMatrix.m[14] = 0;
    finalMatrix.m[15] = 1;
```

Still don't work. Then I ask TA after class, it's clearly that The problem lies in the way I write the matrix, for the **makeatrans** function, the last column should be zero because it is output, so I change it like this:

```
m[0] = 1.0f ;  m[1]  = 0.0f ; m[2]  = 0.0f;  m[3]  = s.x;
m[4] = 0.0f ;  m[5]  = 1.0f ; m[6]  = 0.0f;  m[7]  = s.y;
m[8] = 0.0f ;  m[9]  = 0.0f ; m[10] = 1.0f;  m[11] = s.z;
m[12] = 0.0f ; m[13] = 0.0f ; m[14] = 0.0f;  m[15] = 0.0f;
```

and it freeze as usual, So try to take these three out and **test them separately**, it means that the localWorld will return only one matrix. the code will look below (same as rotation and scale):

```
public Matrix4 localToWorld(){
    Matrix4 result = Matrix4.Trans(transform.position);

    // debug messsage
    if (frameCount % 60 == 0) {
        println("=== Transform Debug ===");
        println("Position: " + transform.position.x() + ", " +
                transform.position.y() + ", " +
                transform.position.z());
        println("Translation Matrix:");
        println(result.toString());
    }

    return result;
}
```

The result doesn't succeed, my debug messages stop the moment I pull the mouse (the slider does not move with my mouse). Any way, I try many method the solve it, but the result still don't work as I expected.

> How I complete this part

I see the note I write in class, with some online information to ensure my note is right, and I ask ChatGPT, Claude to help me with this homework, I also ask TA for help. Although I fail to implement this task, I did my best. I leave my homework like this.

```
void makeRotZ(float a) {
    // TODO HW2
    // You need to implement the rotation of z-axis matrix here. (Yaw)
     makeIdentity();
     float c = cos(a);
     float s = sin(a);
     m[0] = c;
     m[1] = -s;
     m[4] = s;
     m[5] = c;
     m[10] = 1.0f;
     m[15] = 1.0f;
}
```

```
 void makeTrans(Vector3 t) {
    // TODO HW2
    // You need to implement the translate matrix here.
     makeIdentity();
     m[3] = t.x;
     m[7] = t.y;
     m[11] = t.z;
     m[15] = 0.0f;

 }
void makeScale(Vector3 s) {
      makeIdentity();
      m[0] = s.x;
      m[5] = s.y;
      m[10] = s.z;
      m[15] = 1.0f;
}
```

2. Fill the color of polygon.

I've implement two function in this part.

> code snippets & Algorithm

(1) pnpoly function This function uses the **Ray Casting Algorithm** to determine whether a point is inside the polygon:

Main algorithm

1. First I do a basic check to make sure the polygon has at least 3 vertices

```
if (vertexes == null || vertexes.length < 3) {
    return false;
}

int n = vertexes.length;
boolean inside = false;
```

2. Then I emit a horizontal ray to the right from the point to be measured
3. Record the number of intersections of this ray with the polygon edges
    - Use (yi > y) != (yj > y) to determine whether the edge may intersect the ray
    - Calculate the actual intersection position using x < (xj - xi) * (y - yi) / (yj - yi) + xi

```
int n = vertexes.length;
boolean inside = false;

for (int i = 0, j = n - 1; i < n; j = i++) {
    float xi = vertexes[i].x;
    float yi = vertexes[i].y;
    float xj = vertexes[j].x;
    float yj = vertexes[j].y;

    if ((yi > y) != (yj > y) &&
        (x < (xj - xi) * (y - yi) / (yj - yi) + xi)) {
        inside = !inside;
    }
}
```

4. If the number of intersection points is an odd number, the point is inside the polygon; if it is an even number, the point is outside the polygon.

```
if (inside) {
    return true;
}
return false;
```

Key mathematical principles

- Use the concept of vector cross product to determine the relative position of points and edges
- Determine internal and external relationships through even-odd rules

(2) findBoundBox function This function calculates the **Axis-Aligned Bounding Box (AABB)** of the object:

Main algorithm

1. Initialization considers boundary conditions and returns the default value when the vertex array is empty.

```
if (v == null || v.length == 0) {
    Vector3 recordminV = new Vector3(0);
    Vector3 recordmaxV = new Vector3(999);
    Vector3[] result = { recordminV, recordmaxV };
    return result;
}
```

2. Set the coordinates of the first vertex to the initial minimum and maximum values

```
Vector3 recordminV = new Vector3(v[0].x, v[0].y, v[0].z);
Vector3 recordmaxV = new Vector3(v[0].x, v[0].y, v[0].z);
```

3. Traverse all vertices and update the minimum and maximum values of the three axes:
    - x-axis: record minimum x and maximum x
    - y-axis: records minimum y and maximum y
    - z-axis: records minimum z and maximum z

```
for (int i = 1; i < v.length; i++) {

    if (v[i].x < recordminV.x) recordminV.x = v[i].x;
    if (v[i].x > recordmaxV.x) recordmaxV.x = v[i].x;

    if (v[i].y < recordminV.y) recordminV.y = v[i].y;
    if (v[i].y > recordmaxV.y) recordmaxV.y = v[i].y;

    if (v[i].z < recordminV.z) recordminV.z = v[i].z;
    if (v[i].z > recordmaxV.z) recordmaxV.z = v[i].z;
}
```

4. Return an array containing the minimum and maximum points

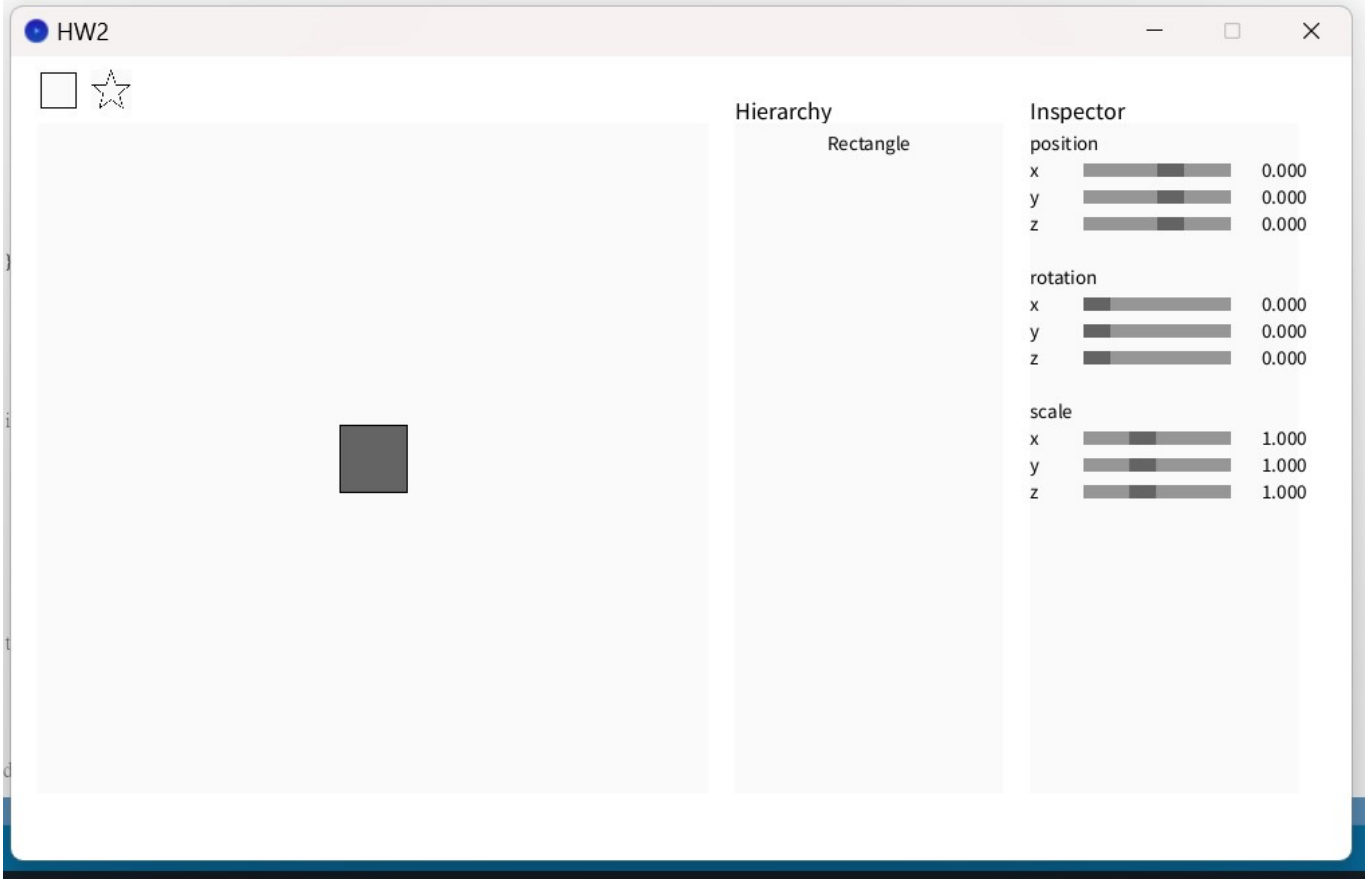
```
Vector3[] result = { recordminV, recordmaxV };
return result;
```

Key technical

- Use Greedy Algorithm to find the extreme value of each axis
- The position and size of the entire bounding box can be determined through two vertices (minimum point and maximum point)

> Demo Screenshot

You can see the polygon is filled with color.



> How I complete this part

I ask ChatGPT to enter the website that given in the HackMD， and according to the architecture in the homework, I ask it to fill the code.

3. Keep the polygon inside the canvas

Since the first task in this homework is failed, I cannot test this algorithm is right or wrong. So I just write it as I thought.

> code snippets & Algorithm

The algorithm works by iteratively processing each edge of the clipping boundary and "clipping" the polygon against each edge. The result is a new polygon that lies inside the clipping boundary.

1. Iterating Over Each Boundary Edge

- For each boundary edge, I clear the output list, since it will hold the clipped polygon for this specific boundary edge.

```
for (int i = 0; i < boundary.length; i++) {
    // Clear the output to store the results of this cropping
    output.clear();

    // Get the two endpoints of the boundary line
    Vector3 clipEdgeStart = boundary[i];
    Vector3 clipEdgeEnd = boundary[(i + 1) % boundary.length];
```

2. Processing Polygon Vertices

- processes each vertex in the current polygon (input). Each vertex is checked relative to the current clipping edge.
- currentPoint refers to the current vertex, and prevPoint is the previous vertex in the polygon.

```
for (int j = 0; j < input.size(); j++) {
    // Get current point and previous point
    Vector3 currentPoint = input.get(j);
    Vector3 prevPoint = input.get((j + input.size() - 1) % input.size());
```

3. Checking Whether a Point is Inside or Outside the Clipping Boundary

- Inside/Outside Determination: This code calculates the "side" of the clipping edge that each point lies on.
- The currentSide and prevSide variables are the results of a 2D cross product that determines if the points are to the left or right of the clipping edge.
- If the result is positive, the point is to the left (inside the clipping region), and if it is negative, the point is to the right (outside the clipping region).
- The algorithm assumes the clipping boundary is convex and that "inside" means to the left of each edge (or the specified inside direction).

```
float currentSide = (clipEdgeEnd.x - clipEdgeStart.x) * (currentPoint.y - clipEdgeStart.y) -
                    (clipEdgeEnd.y - clipEdgeStart.y) * (currentPoint.x - clipEdgeStart.x);
float prevSide = (clipEdgeEnd.x - clipEdgeStart.x) * (prevPoint.y - clipEdgeStart.y) -
                 (clipEdgeEnd.y - clipEdgeStart.y) * (prevPoint.x - clipEdgeStart.x);
```

4. Handling Different Cases (Inside and Outside Points)

- If the current point is inside the boundary (i.e., currentSide <= 0), we simply add it to the output polygon.
- If the previous point was outside the boundary (i.e., prevSide > 0), we compute the intersection between the previous edge and the clipping edge.
- If the current point is outside the boundary (currentSide > 0), but the previous point is inside (prevSide <= 0), we compute the intersection point between the current edge and the clipping boundary and add the intersection to the output polygon.

```
if (currentSide <= 0) {
    // If the previous point is outside, calculate and add the intersection point
    if (prevSide > 0) {
        // Calculate intersection point
        float A1 = currentPoint.y - prevPoint.y;
        float B1 = prevPoint.x - currentPoint.x;
        float C1 = A1 * prevPoint.x + B1 * prevPoint.y;

        float A2 = clipEdgeEnd.y - clipEdgeStart.y;
        float B2 = clipEdgeStart.x - clipEdgeEnd.x;
        float C2 = A2 * clipEdgeStart.x + B2 * clipEdgeStart.y;

        float det = A1 * B2 - A2 * B1;
        if (Math.abs(det) > 1e-7) {          // Avoid dividing by 0
            float intersectX = (B2 * C1 - B1 * C2) / det;
            float intersectY = (A1 * C2 - A2 * C1) / det;
            output.add(new Vector3(intersectX, intersectY, 0));
        }
    }
    // Add current point
    output.add(currentPoint);
}
```

5. Calculating the Intersection Point

- If there is an intersection between the clipping edge and the current polygon edge, this code calculates the intersection point using a system of linear equations derived from the line equations of both the polygon edge and the clipping edge.
- The determinant det is calculated to avoid dividing by zero (which could occur if the two lines are parallel).
- The intersection point is calculated and added to the output.

```
    else if (prevSide <= 0) {
        // Calculate intersection point
        float A1 = currentPoint.y - prevPoint.y;
        float B1 = prevPoint.x - currentPoint.x;
        float C1 = A1 * prevPoint.x + B1 * prevPoint.y;

        float A2 = clipEdgeEnd.y - clipEdgeStart.y;
        float B2 = clipEdgeStart.x - clipEdgeEnd.x;
        float C2 = A2 * clipEdgeStart.x + B2 * clipEdgeStart.y;

        float det = A1 * B2 - A2 * B1;
        if (Math.abs(det) > 1e-7) {     // Avoid dividing by 0
            float intersectX = (B2 * C1 - B1 * C2) / det;
            float intersectY = (A1 * C2 - A2 * C1) / det;
            output.add(new Vector3(intersectX, intersectY, 0));
        }
    }
```

6. Preparing for Next Iteration

- After processing all the vertices against one clipping edge, the output list (which now contains the clipped polygon) becomes the input for the next iteration (for clipping against the next boundary edge).

```
input = new ArrayList<Vector3>(output);
```

7. Returning the Final Clipped Polygon

```
    Vector3[] result = new Vector3[output.size()];
    for (int i = 0; i < result.length; i += 1) {
        result[i] = output.get(i);
    }
    return result;
```

> How I complete this part

I ask Claude and ChatGPT as well, and I also ask it to explain the step he give it to me.