Approach:

① Reconstruct given window object

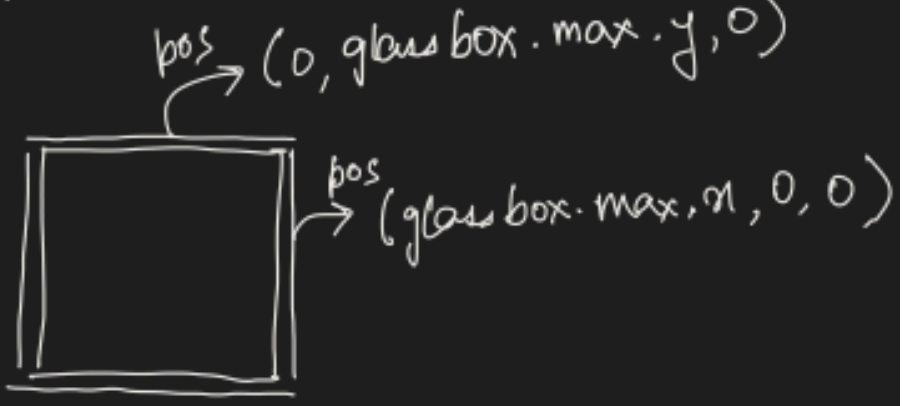
(*) Glass → scale (1, 1, 1)
pos (0, 0, 0)

(*) Frame

- Top
- Bottom → Frame, Space
- Left
- Right

✓ scale to glass width & height

✓ pos @ glassbox min & max



② Width & Height I/P for window

$$\text{widthScale} = \frac{\text{I/P width}}{\text{original width}}$$

glass scaled to (widthScale, HeightScale)

— scaled to widthScale
— positioned to glassbox.min/max.y

| scaled to heightScale
| positioned to glassbox.min/max.x

(3) Corner Type : Chamfer

(a)

scale = $x - w$ / original size.
(x)

pos = $x - w/2$
(x)

Similarly scaled & shifted.

(b)

new frame is spawned

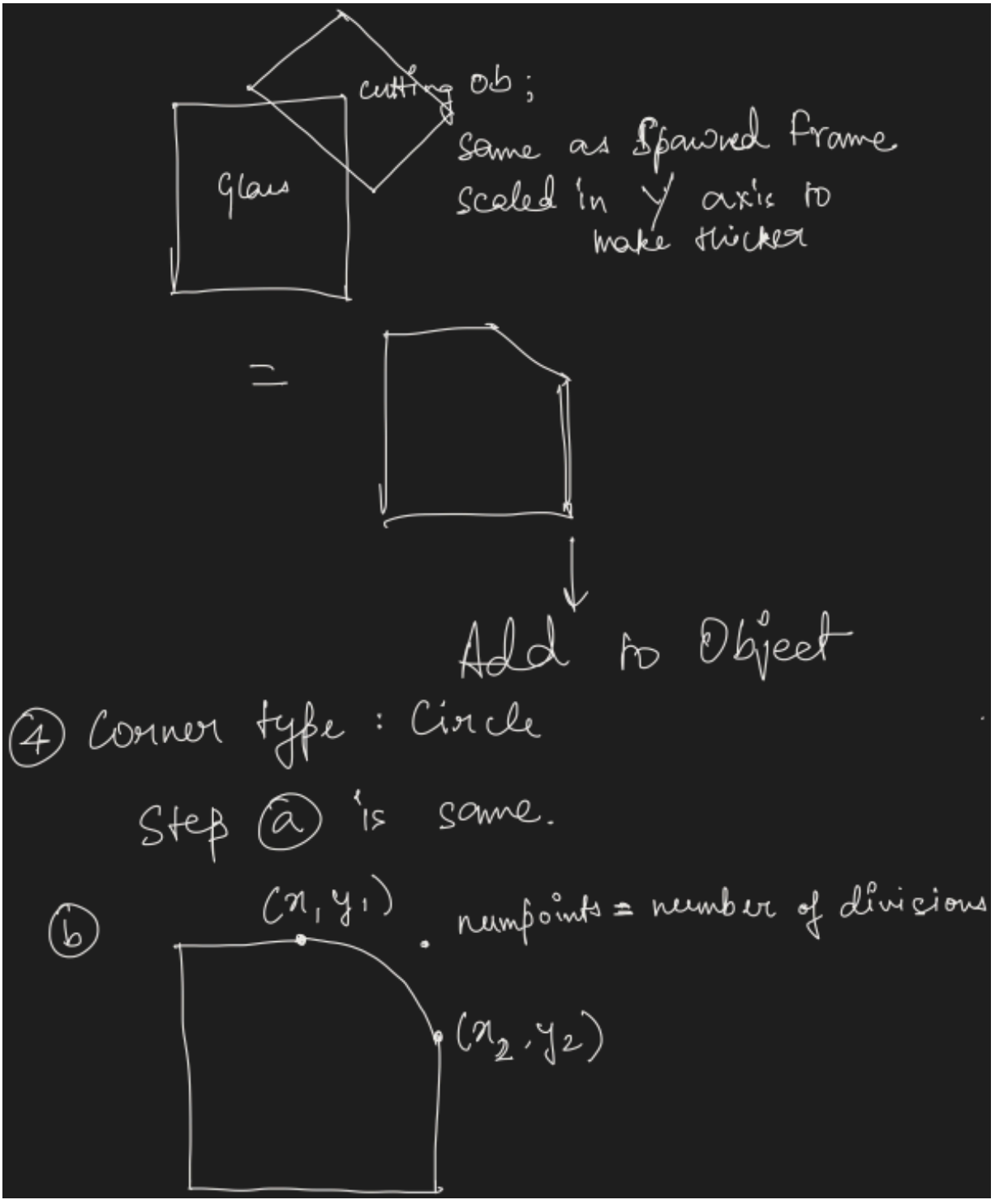
length = $\sqrt{w^2 + h^2}$

position = glassbox.max.x − $w/2$
(x)

(y) glassbox.max.y − $h/2$

rotation = $90^\circ + \left(90^\circ - \cos^{-1}\left(\frac{w/2}{\sqrt{w^2+h^2}/2}\right)\right)$

(c) After frame is spawned
Glass is cut using three-bvh-csg




```
const startPoint = new THREE.Vector2(x1, y1);
const endPoint = new THREE.Vector2(x2, y2);
// Calculate the center point (midpoint of the line segment connecting the two points)
const centerPoint = new THREE.Vector2( (x1 + x2) / 2, (y1 + y2) / 2 );
// Calculate the radius (distance from the center point to one of the end points)
const radius = startPoint.distanceTo(centerPoint);
// Calculate the start and end angles
const startAngle = Math.atan2(y1 - centerPoint.y, x1 - centerPoint.x);
const endAngle = Math.atan2(y2 - centerPoint.y, x2 - centerPoint.x);
```

```
// Generate points along the arc const points = [];
for (let i = 0; i < numPoints; i++) {
        const angle = startAngle + (endAngle - startAngle) * (i / (numPoints - 1));
        const x = centerPoint.x + radius * Math.cos(angle);
        const y = centerPoint.y + radius * Math.sin(angle);
        points.push(new THREE.Vector2(x, y));
}
```