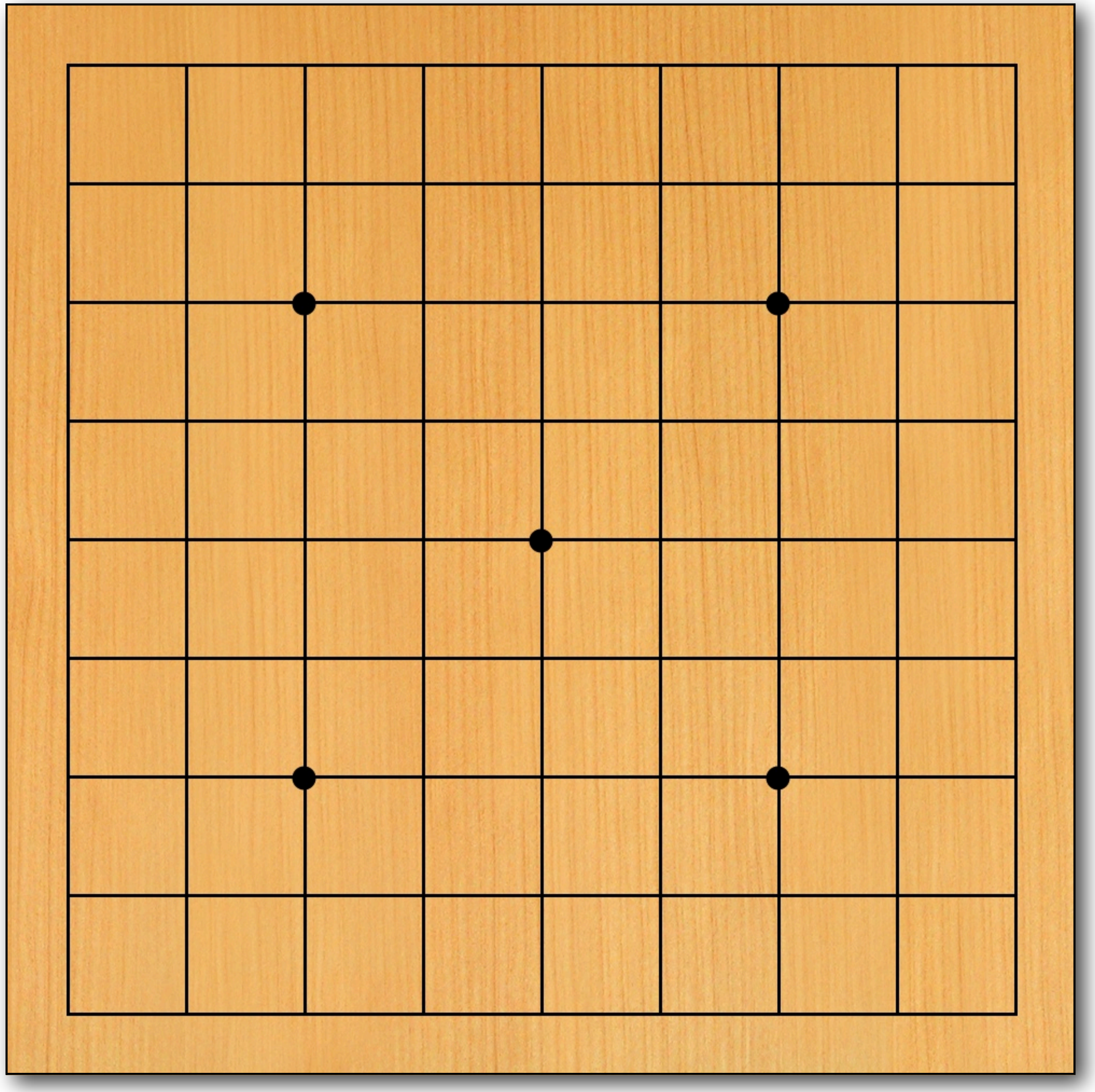
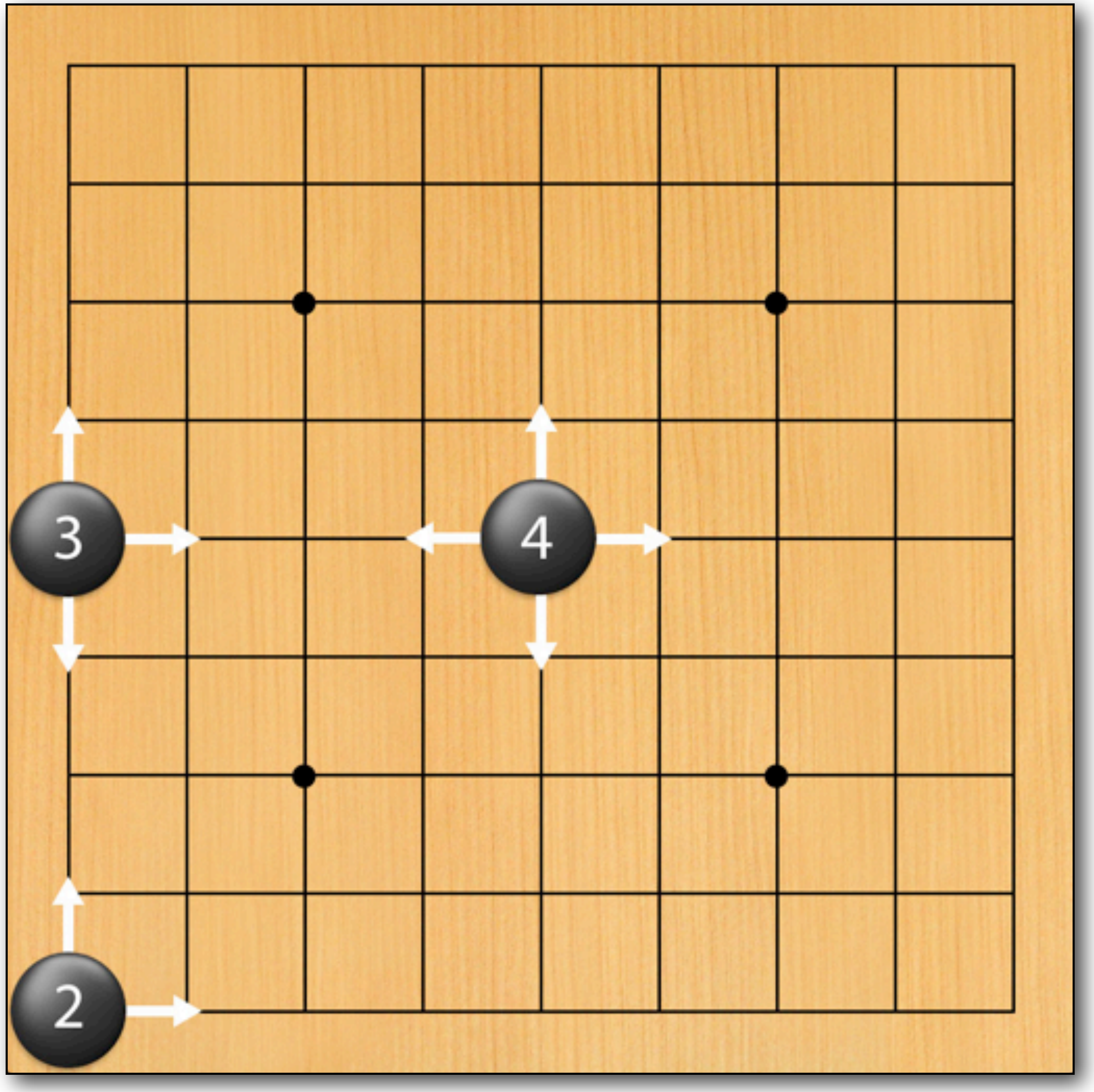
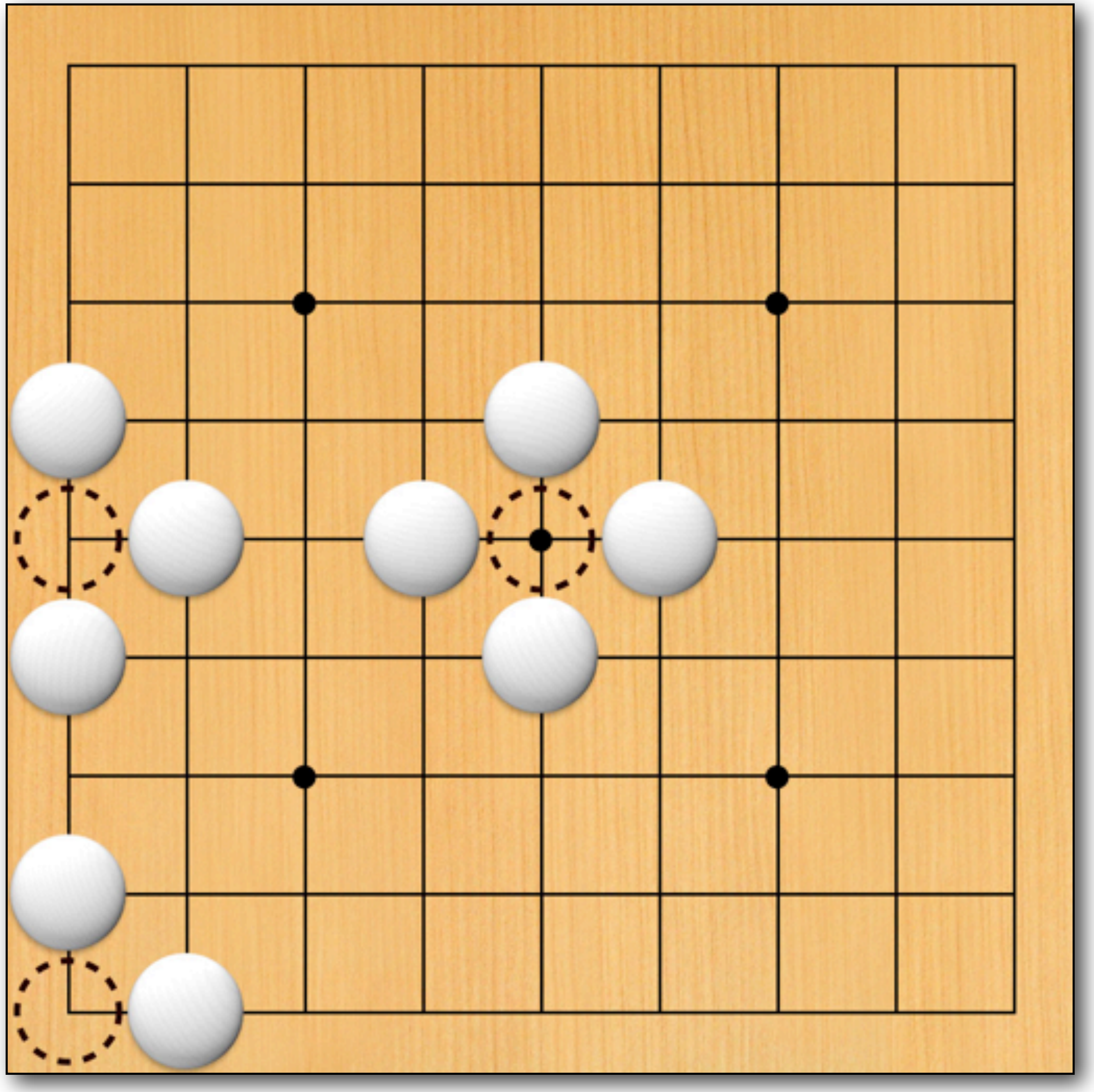
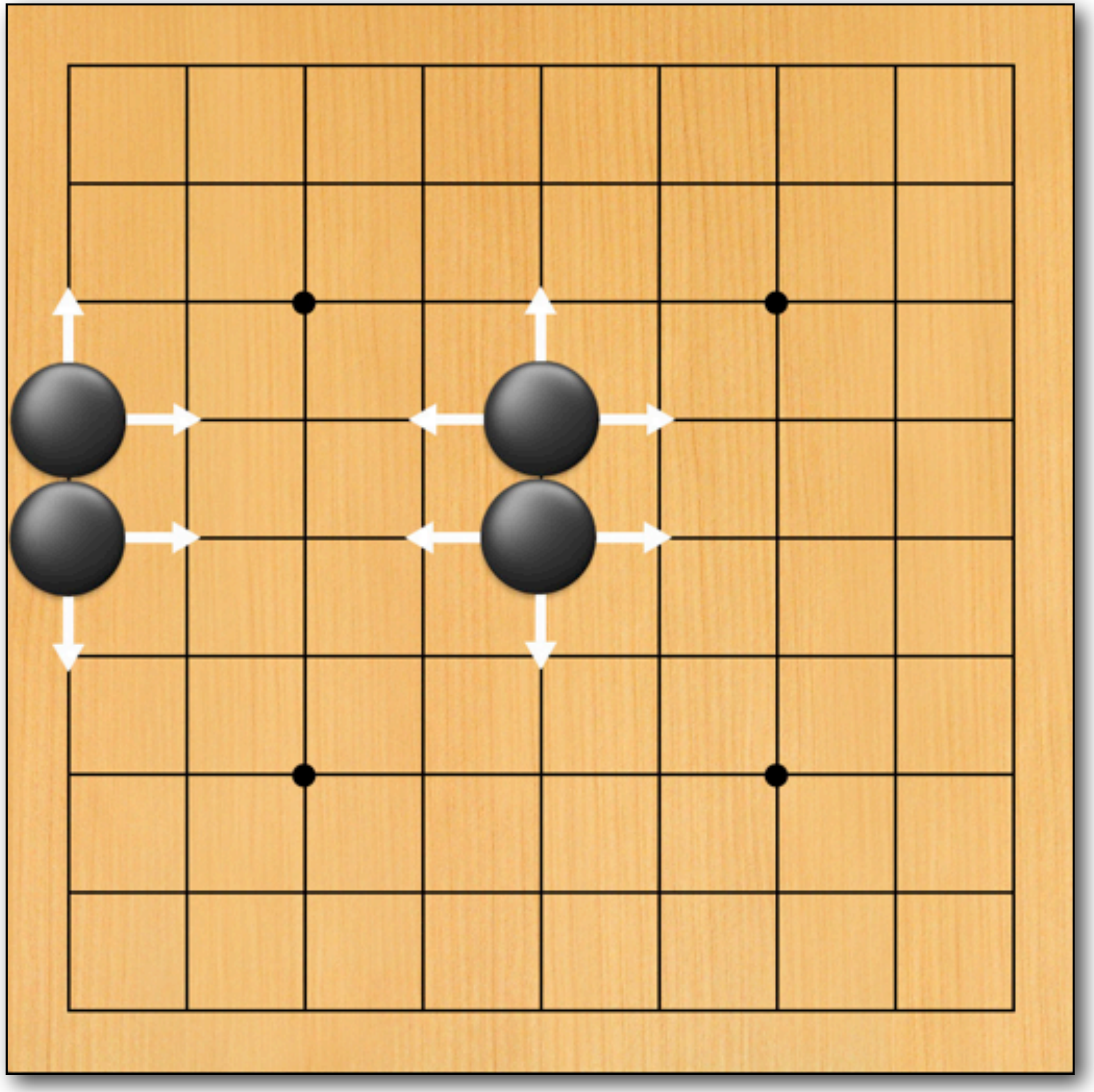
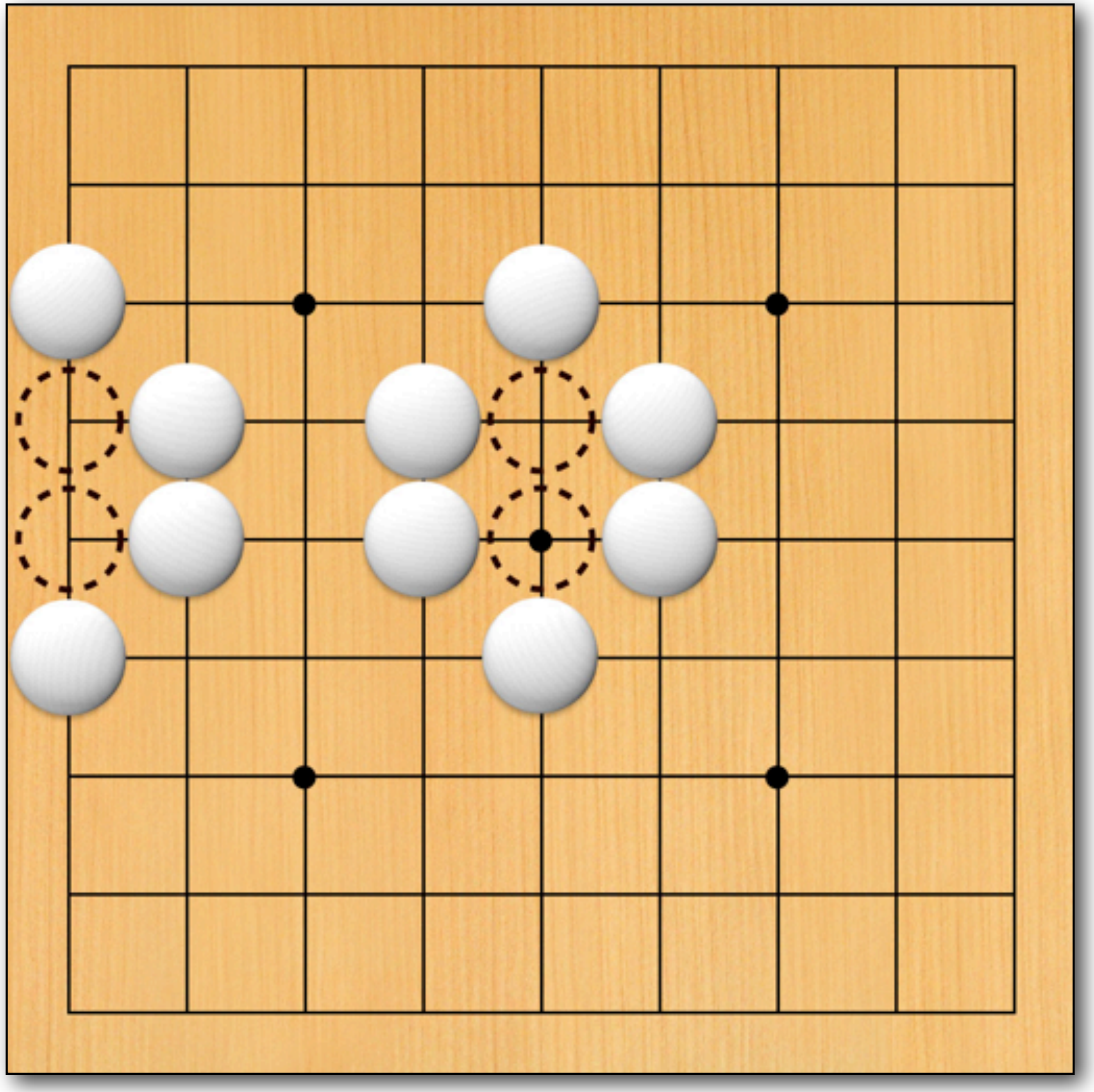
Virtual Go | Simulating a Go Board and Stones

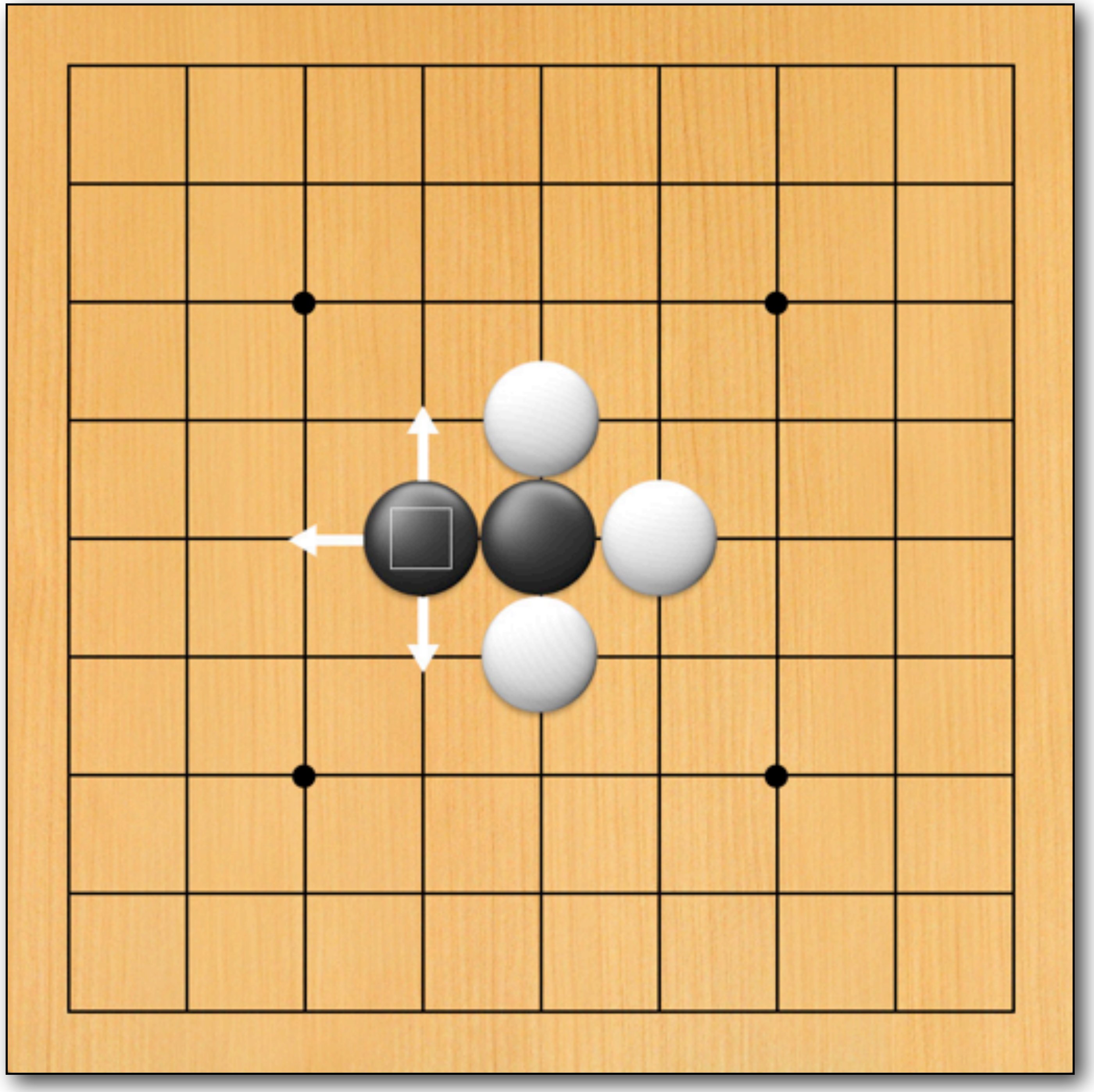


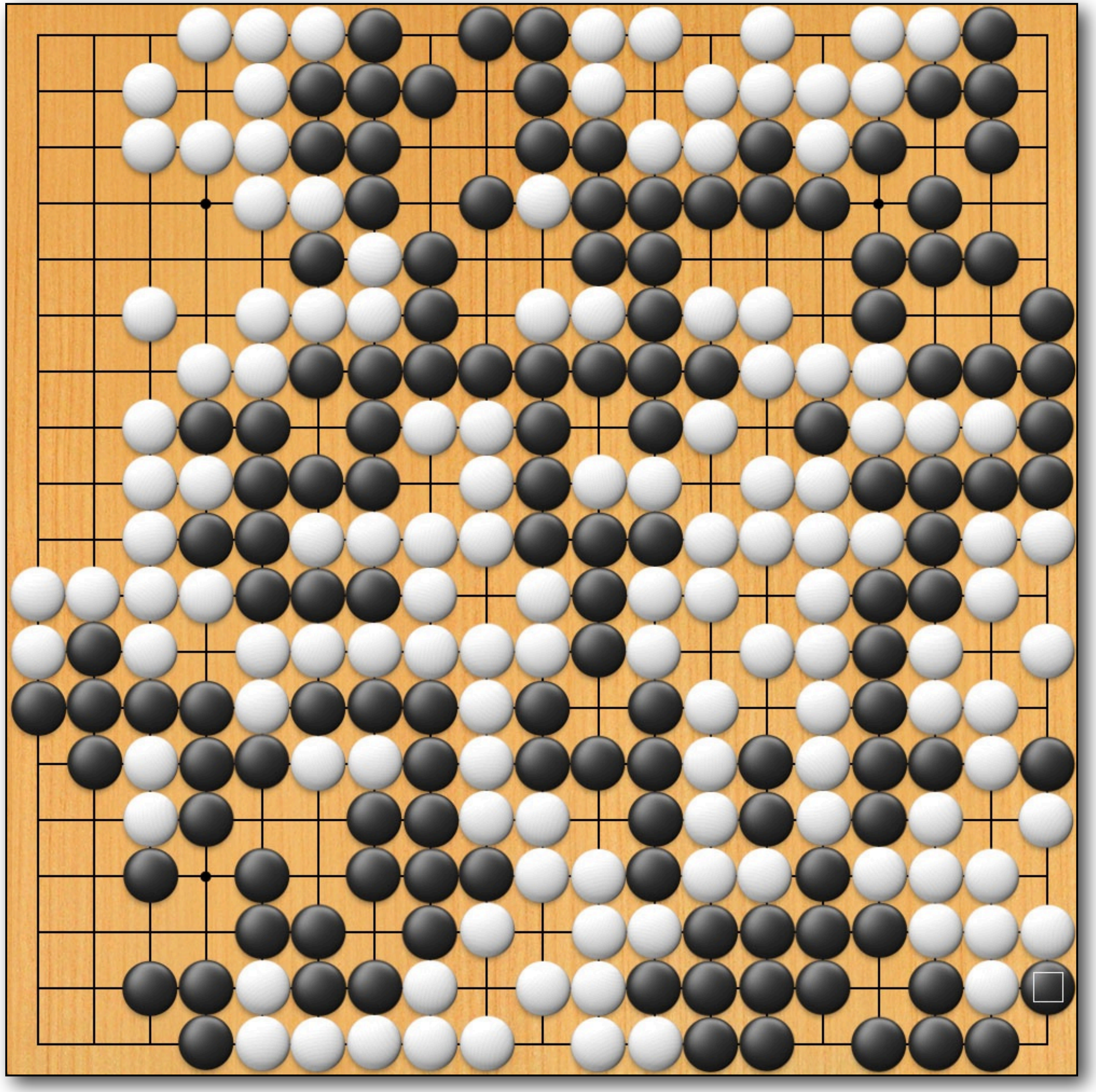

3
4
2



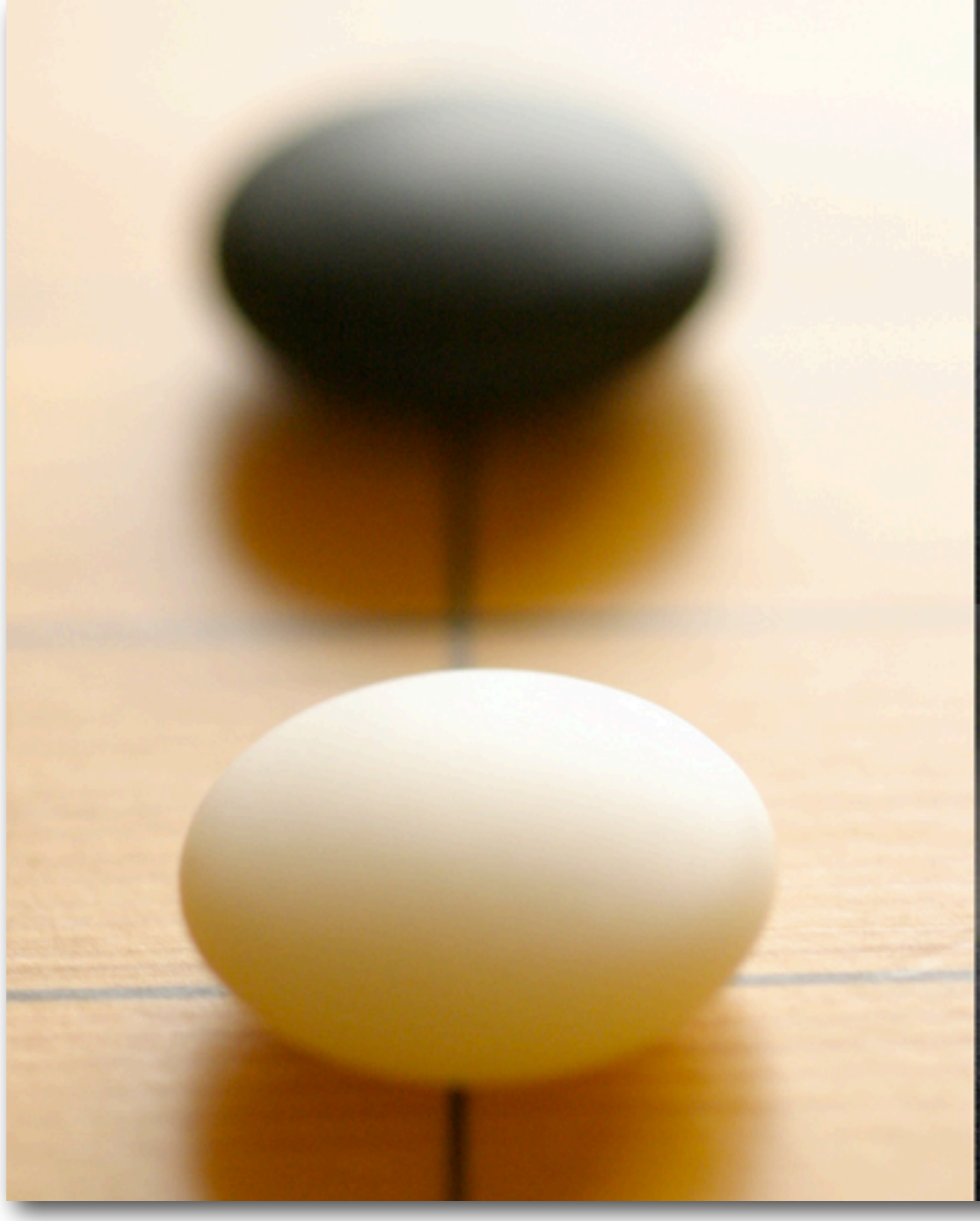


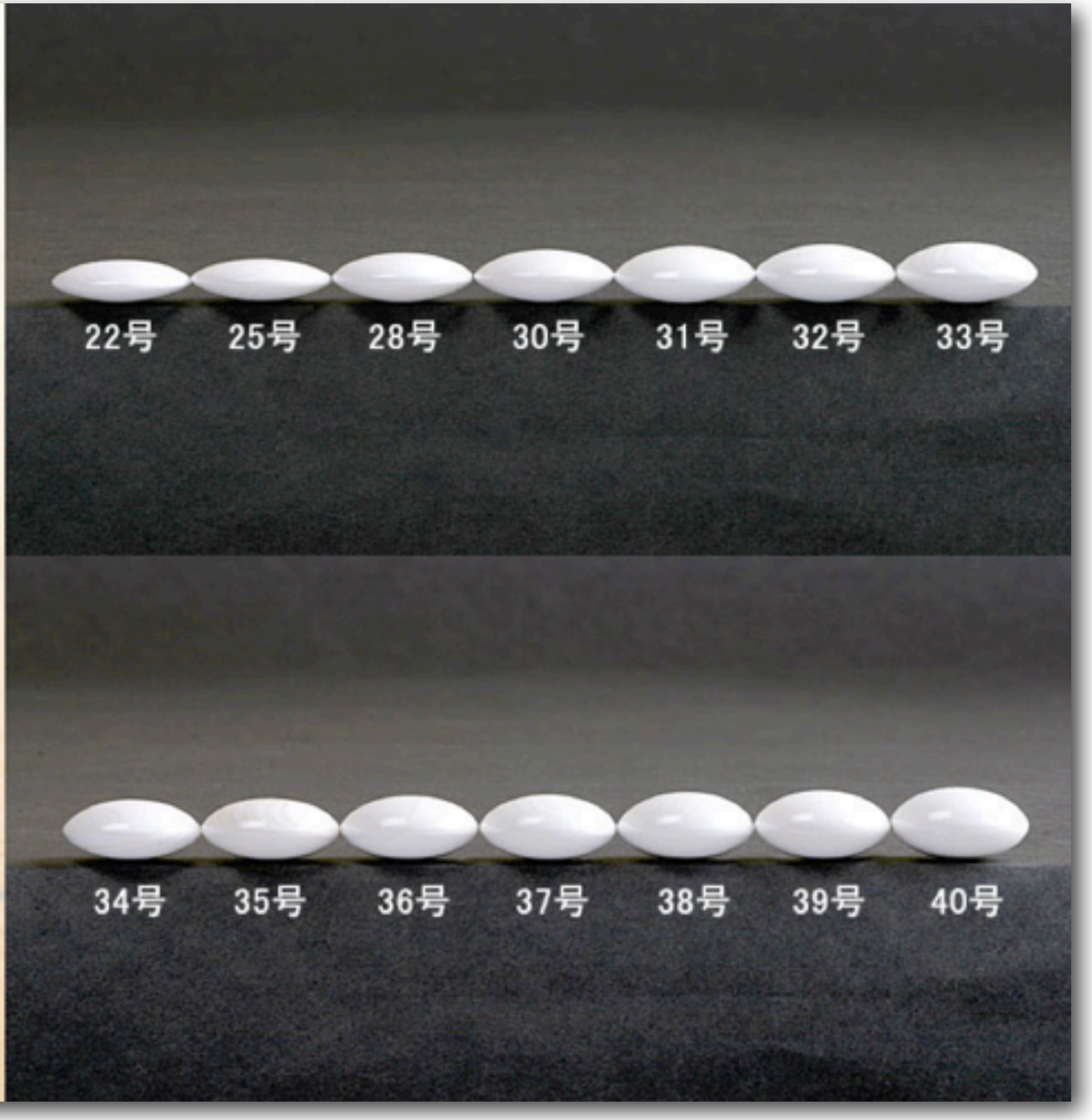


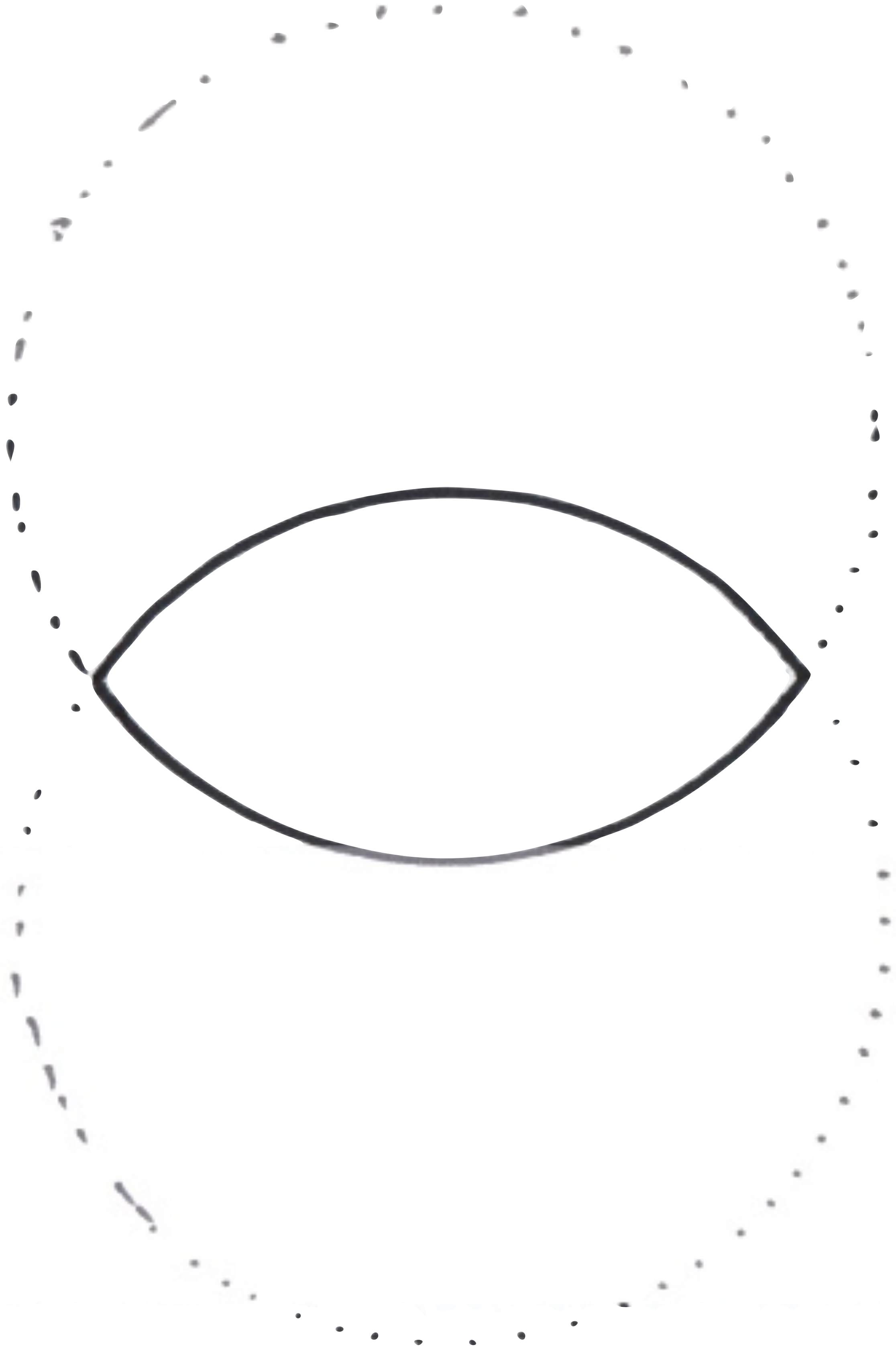


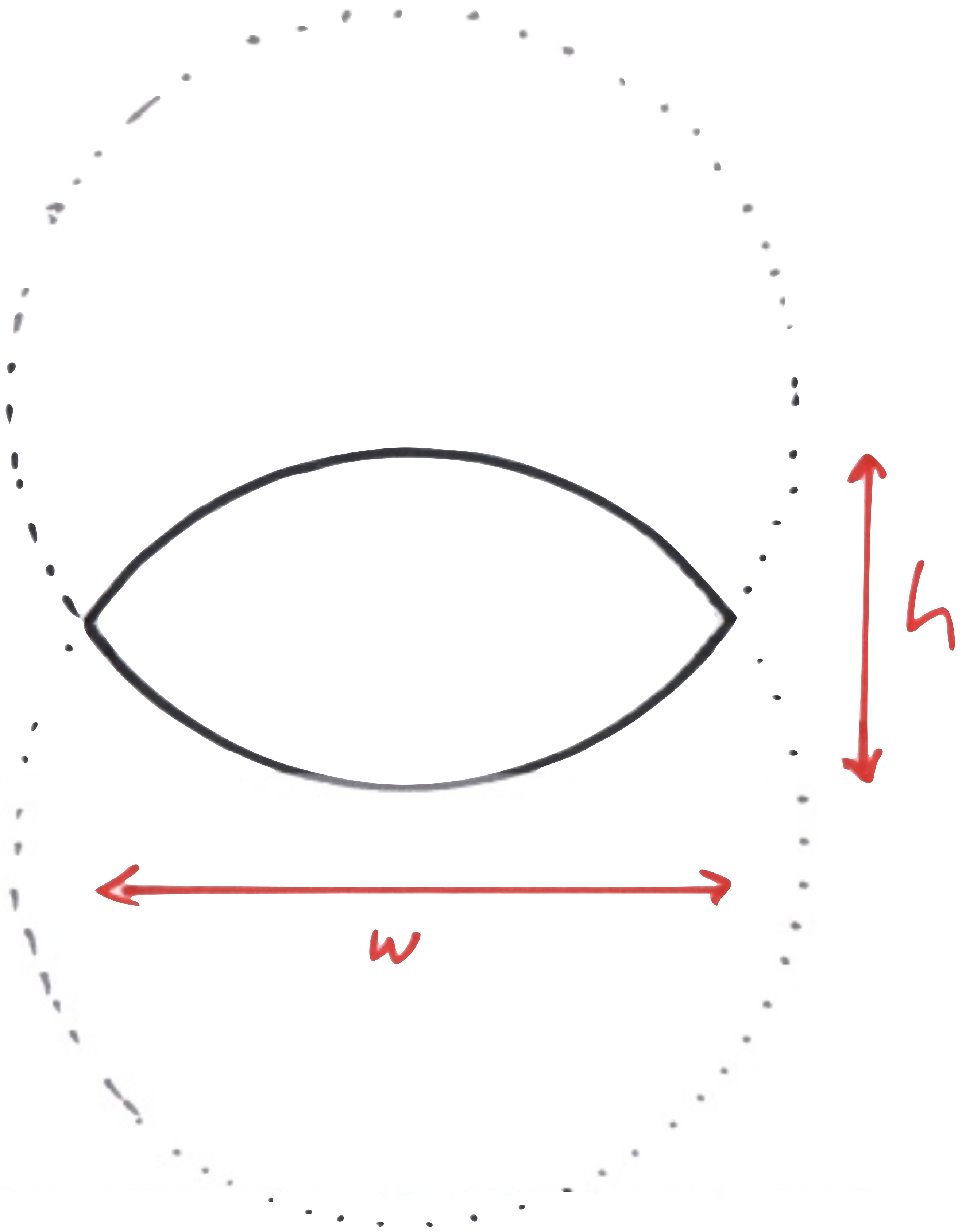


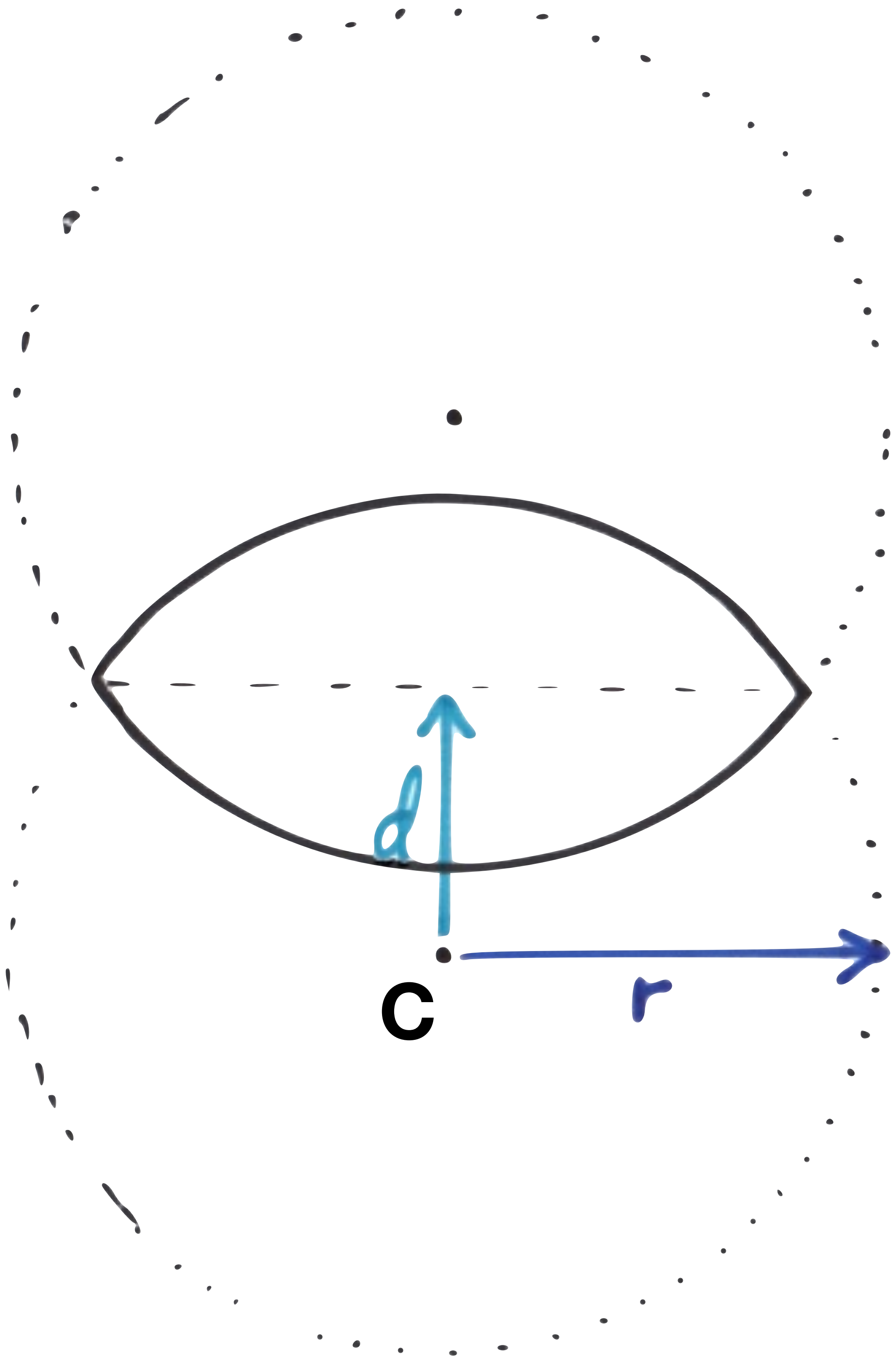
The shape of the go stone

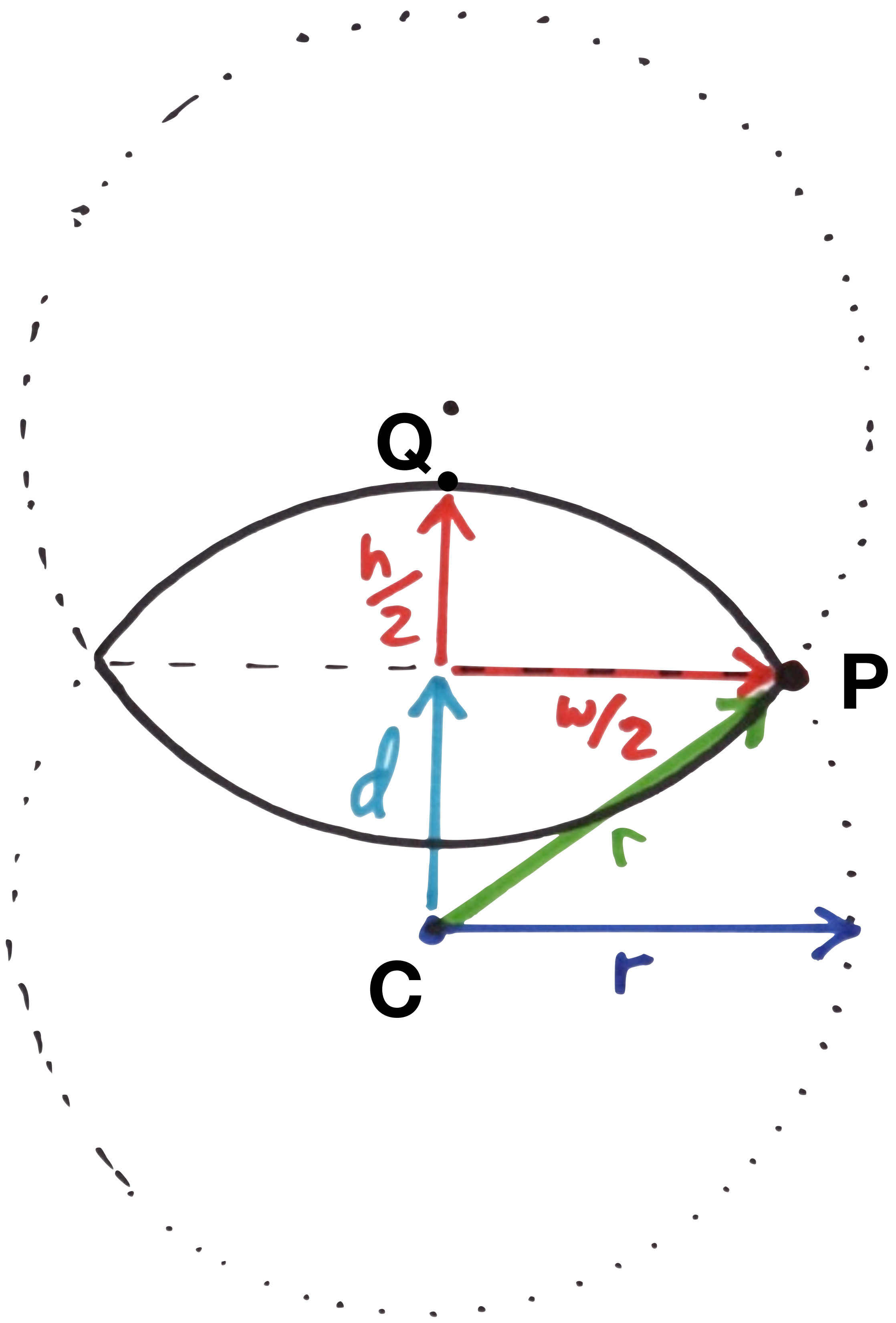


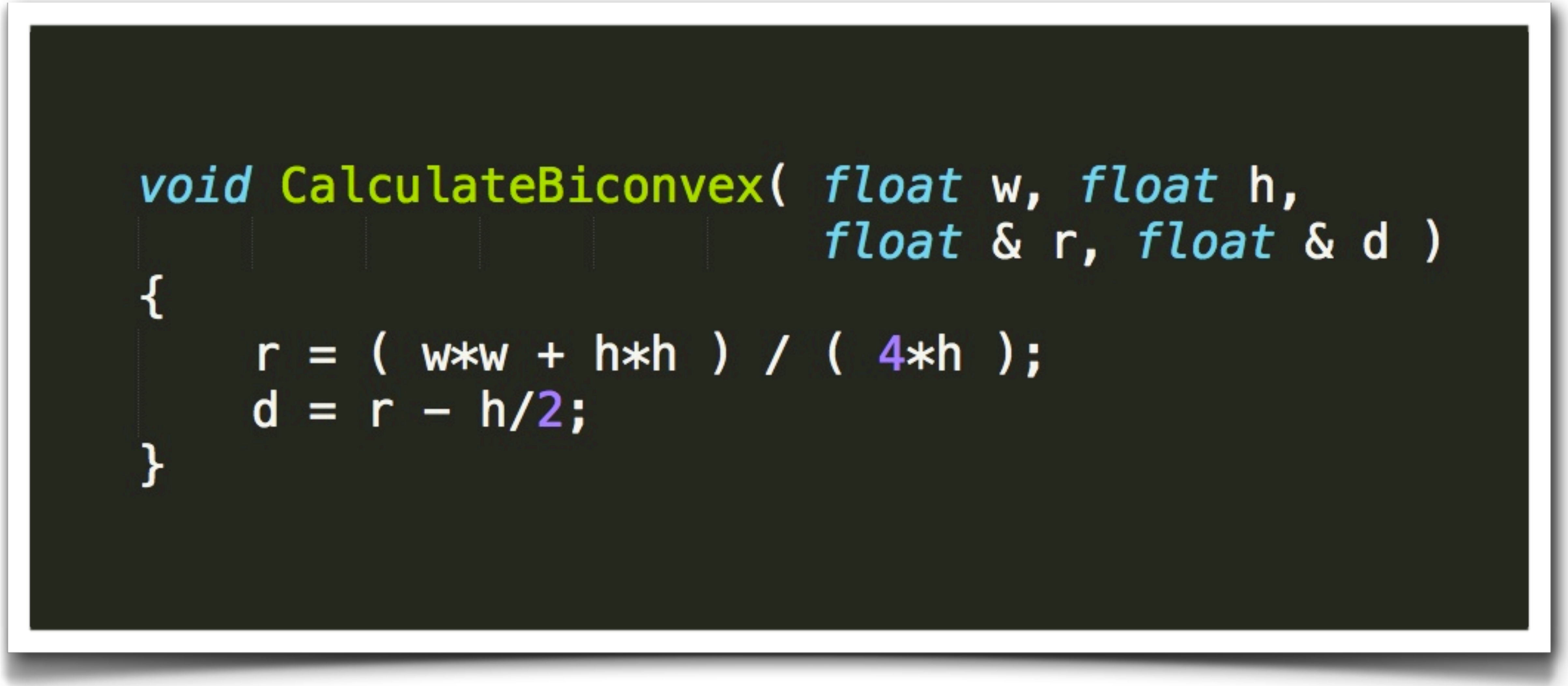

22号
25号
28号
30号
31号
32号
33号
34号
35号
36号
37号
38号
39号
40号



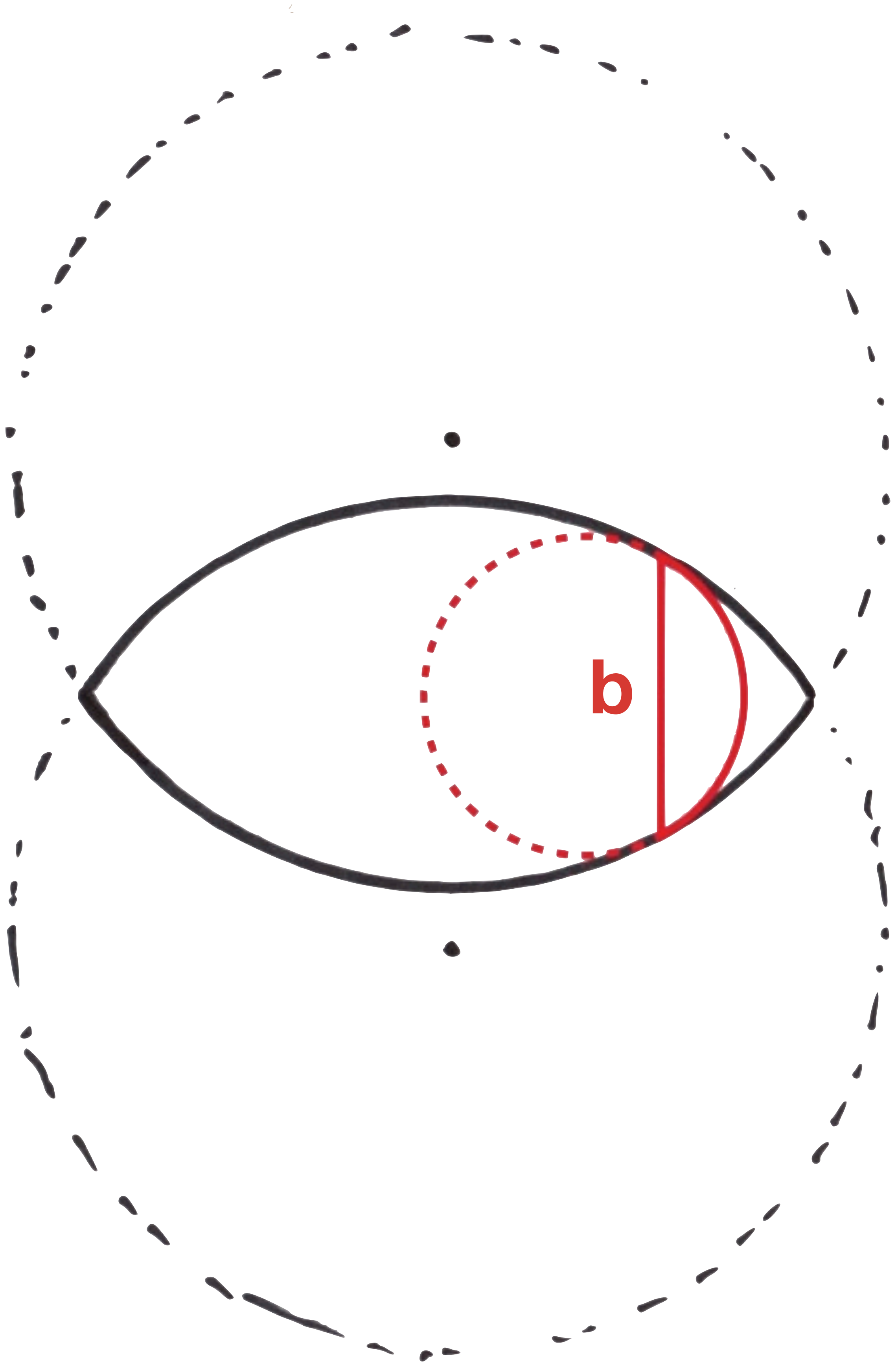

h
w

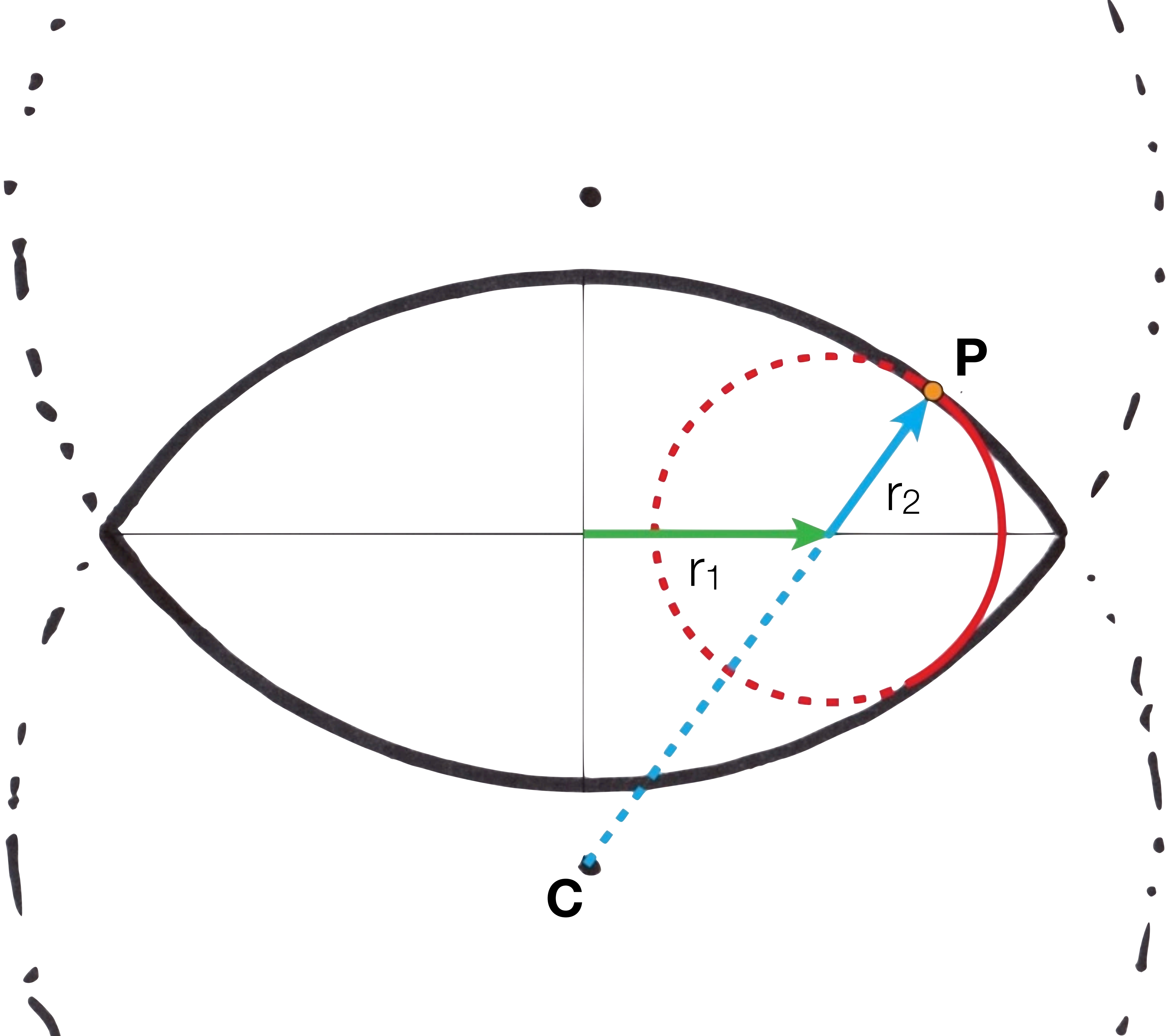

d
c
r

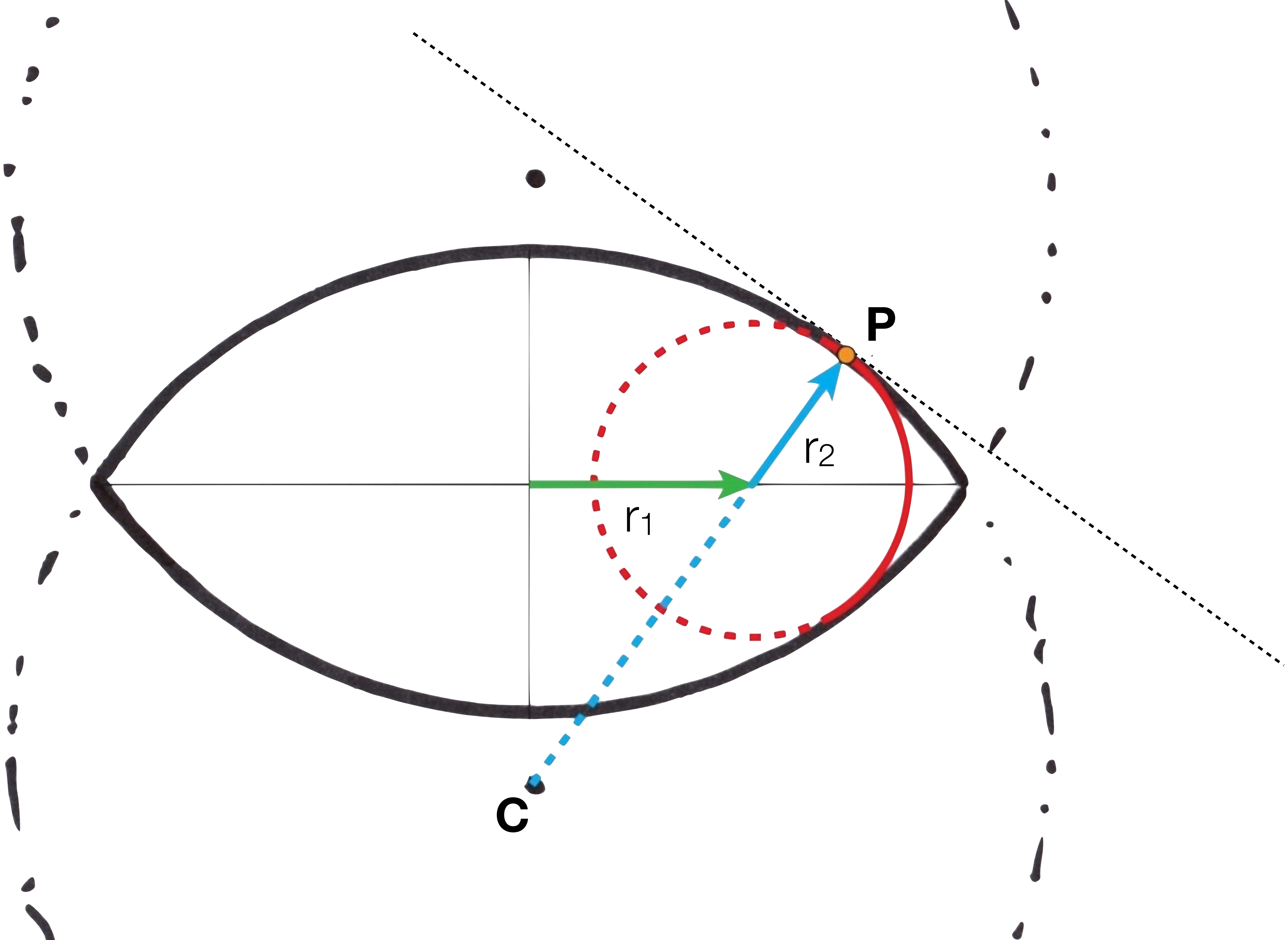

Q
h/2
P
w/2
d
r
C
r

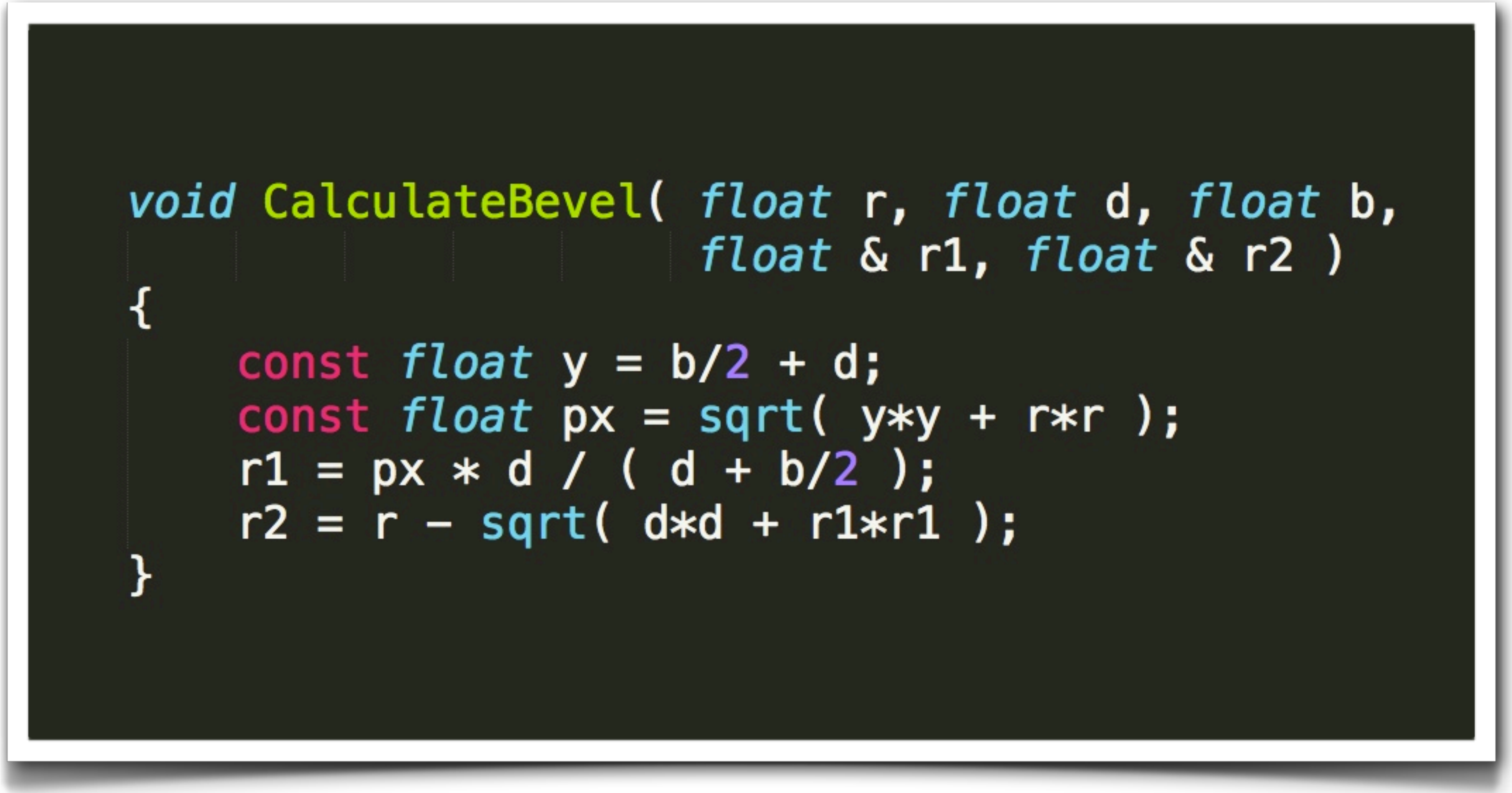


```
void CalculateBiconvex( float w, float h,
                        float & r, float & d )
{
    r = ( w*w + h*h ) / ( 4*h );
    d = r - h/2;
}
```


b


P
r_2
r_1
C

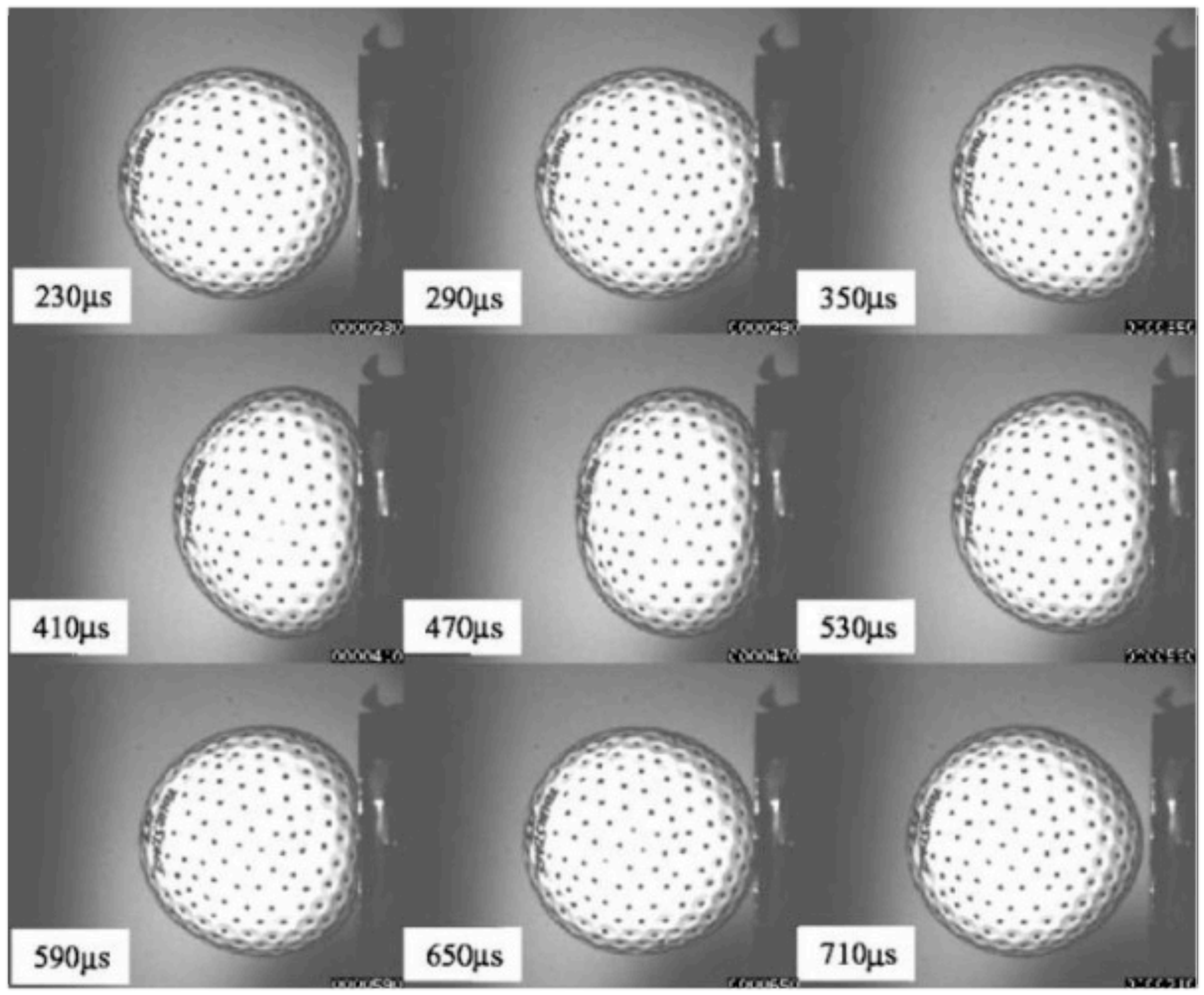

P
r2
r1
C

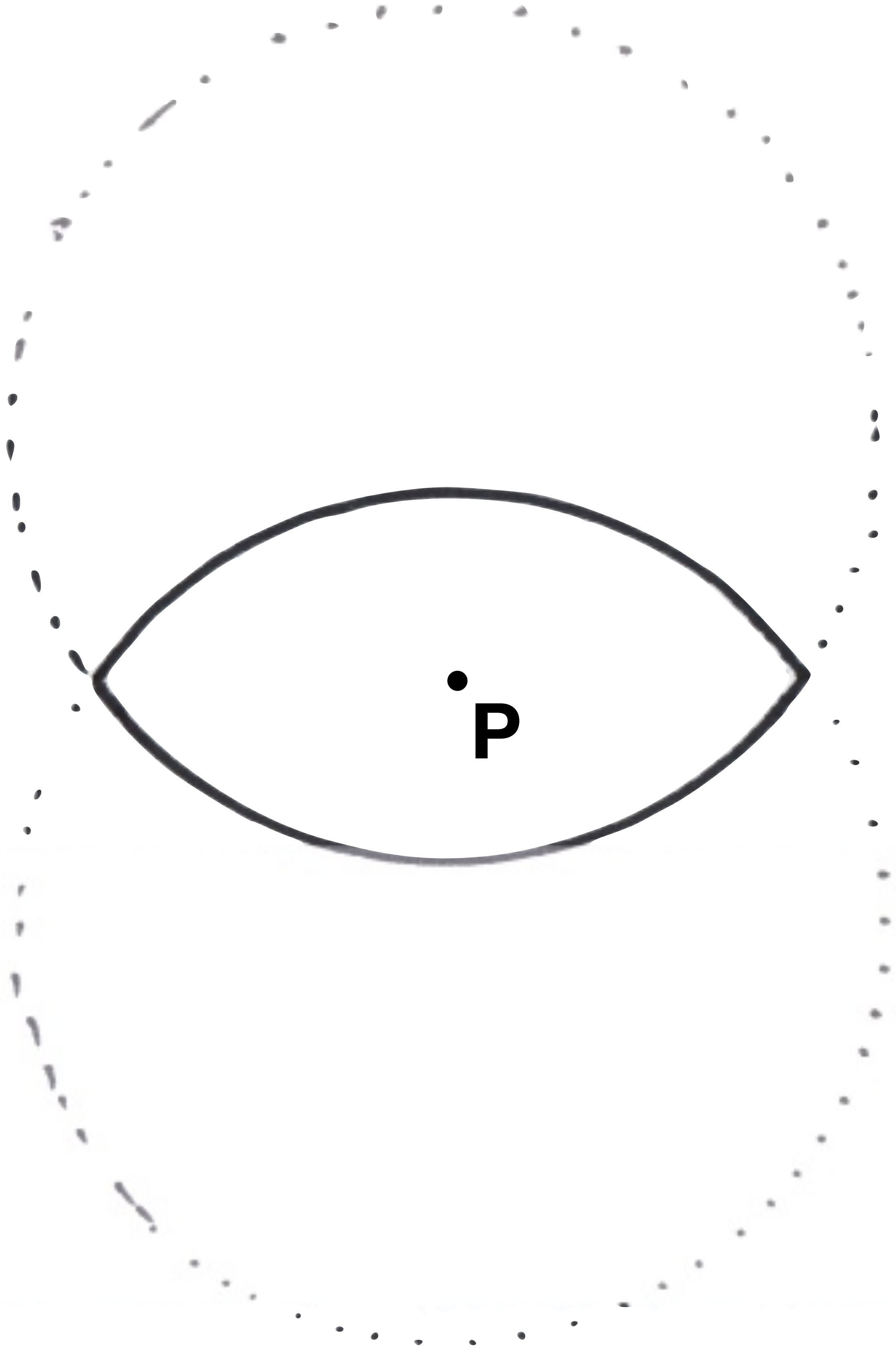


```
void CalculateBevel( float r, float d, float b,
                     float & r1, float & r2 )
{
    const float y = b/2 + d;
    const float px = sqrt( y*y + r*r );
    r1 = px * d / ( d + b/2 );
    r2 = r - sqrt( d*d + r1*r1 );
}
```

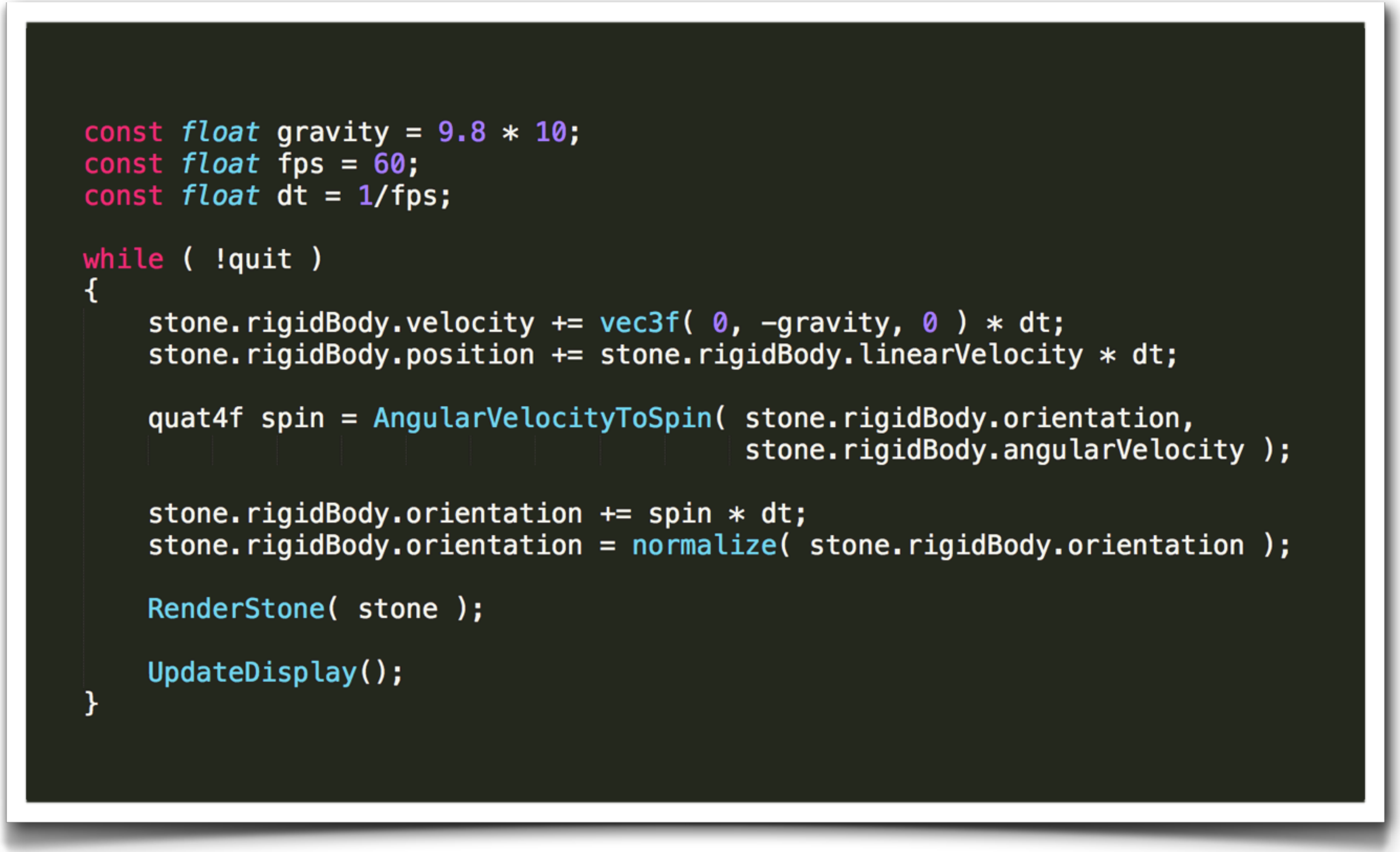
Tessellation demo

Rigid body dynamics


230µs
290µs
350µs
410µs
470µs
530µs
590µs
650µs
710µs

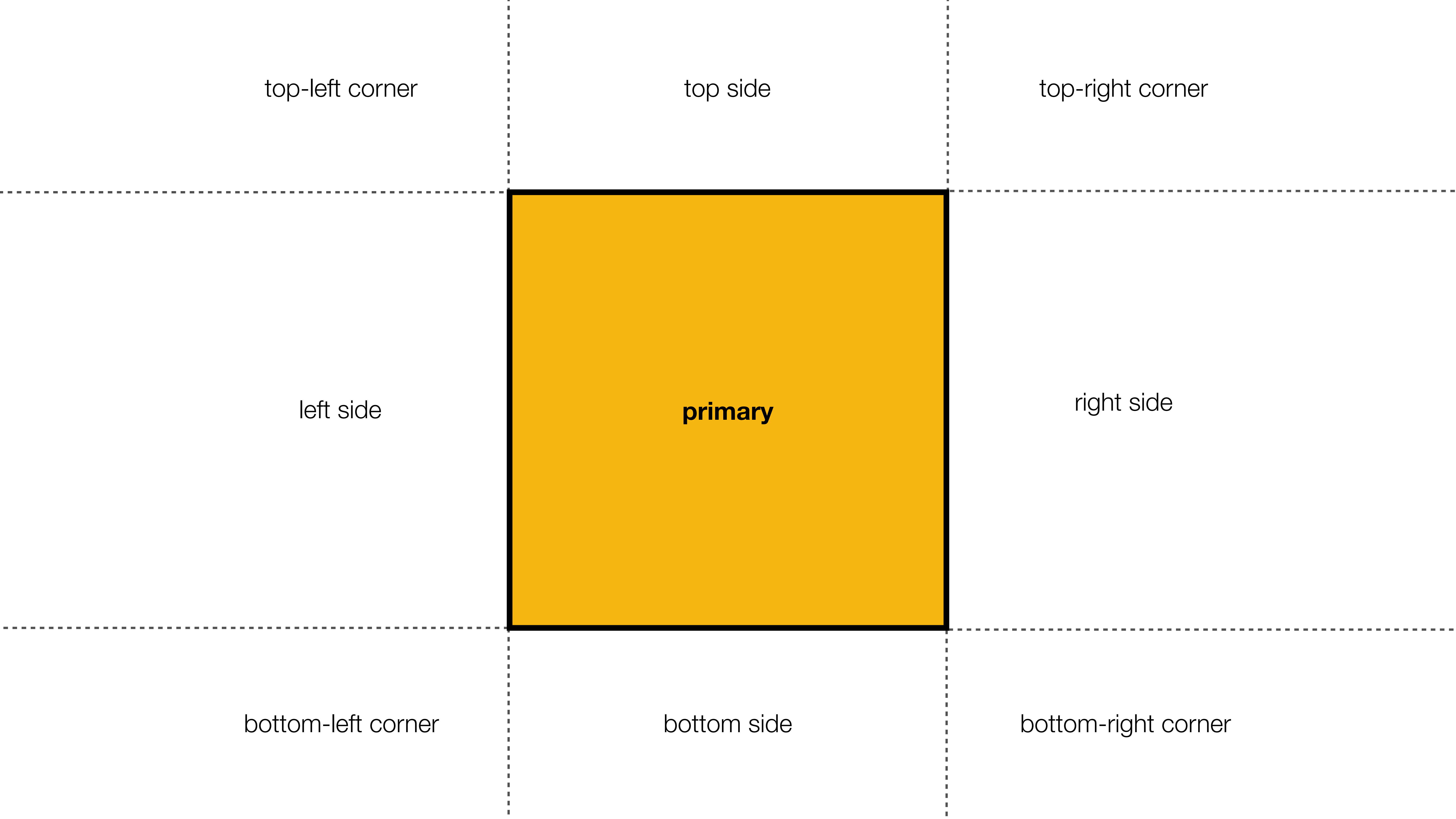

P

Dynamics demo

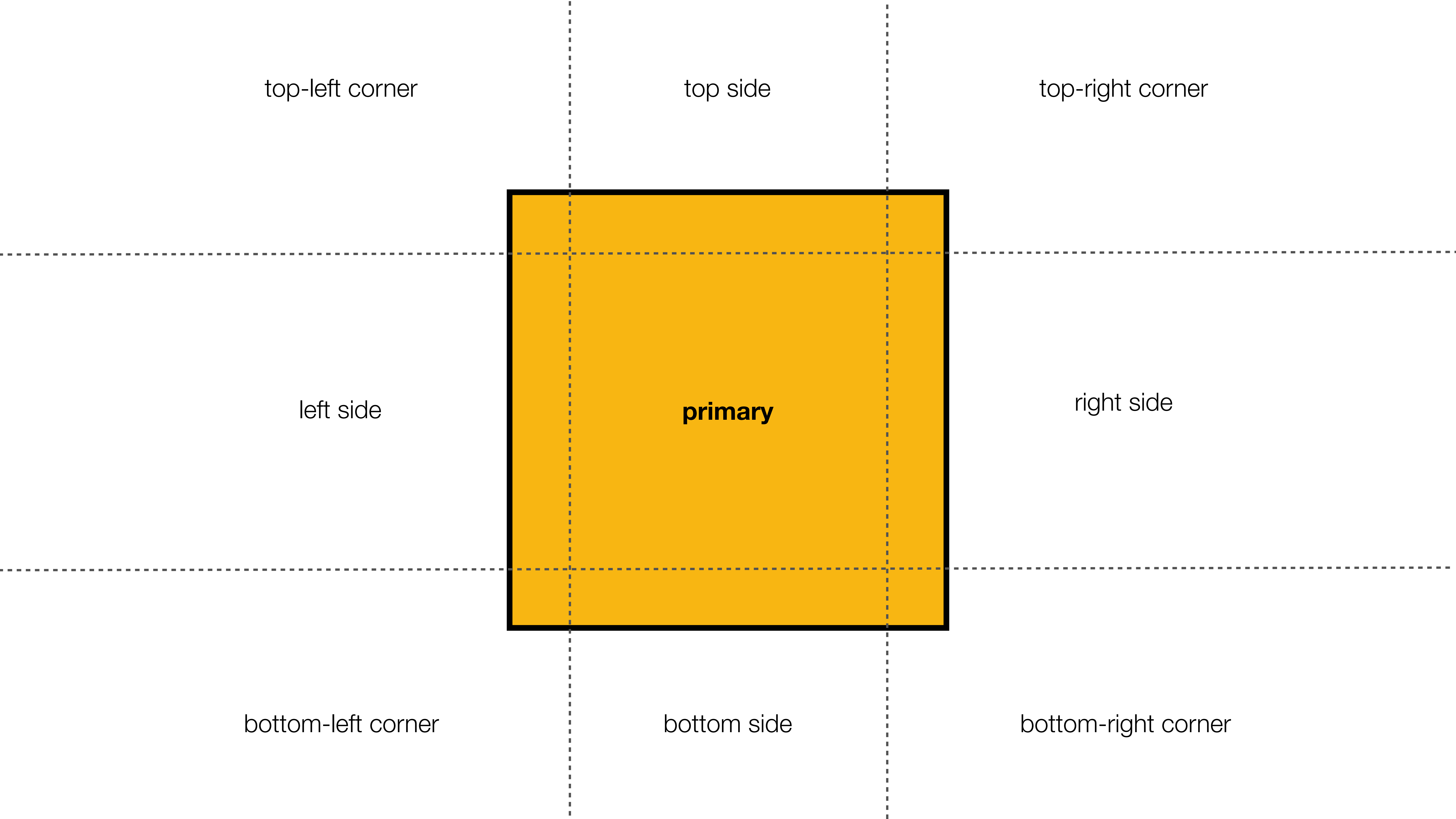


```
const float gravity = 9.8 * 10;
const float fps = 60;
const float dt = 1/fps;

while ( !quit )
{
    stone.rigidBody.velocity += vec3f( 0, -gravity, 0 ) * dt;
    stone.rigidBody.position += stone.rigidBody.linearVelocity * dt;

    quat4f spin = AngularVelocityToSpin( stone.rigidBody.orientation,
                                         stone.rigidBody.angularVelocity );

    stone.rigidBody.orientation += spin * dt;
    stone.rigidBody.orientation = normalize( stone.rigidBody.orientation );

    RenderStone( stone );

    UpdateDisplay();
}
```

Collision detection



top-left corner	top side	top-right corner
left side	**primary**	right side
bottom-left corner	bottom side	bottom-right corner



top-left corner

top side

top-right corner

left side

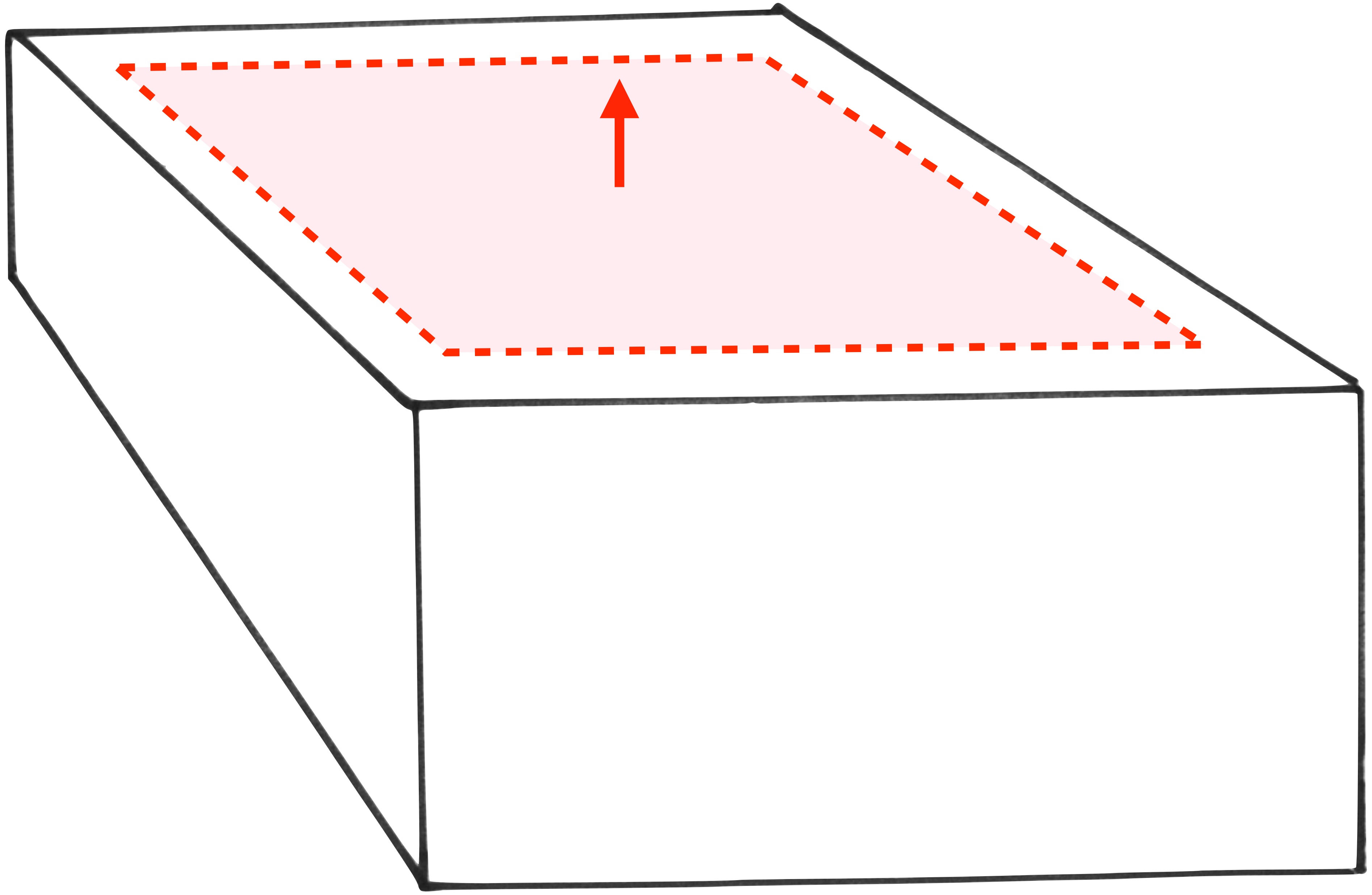
primary

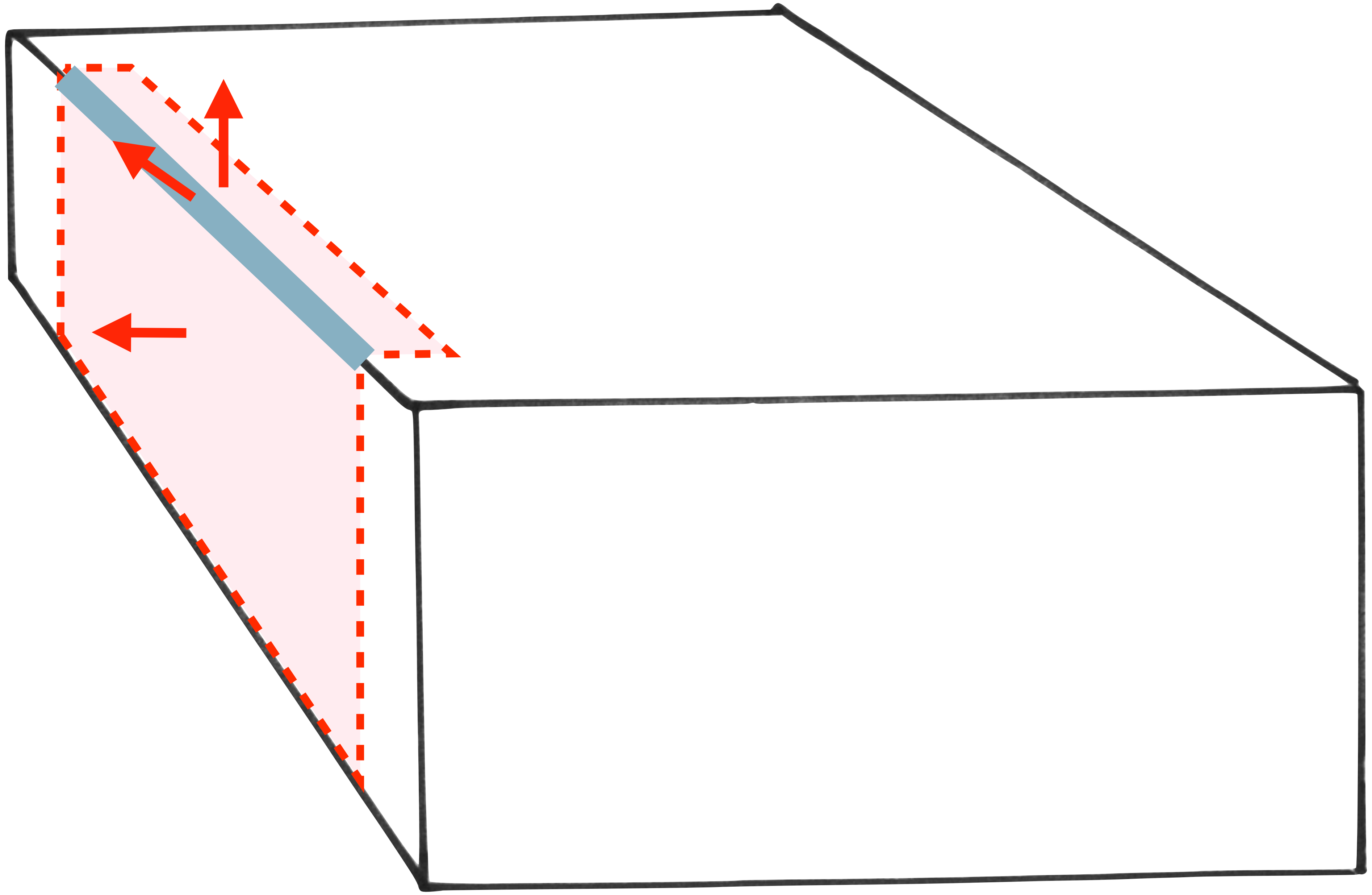
right side

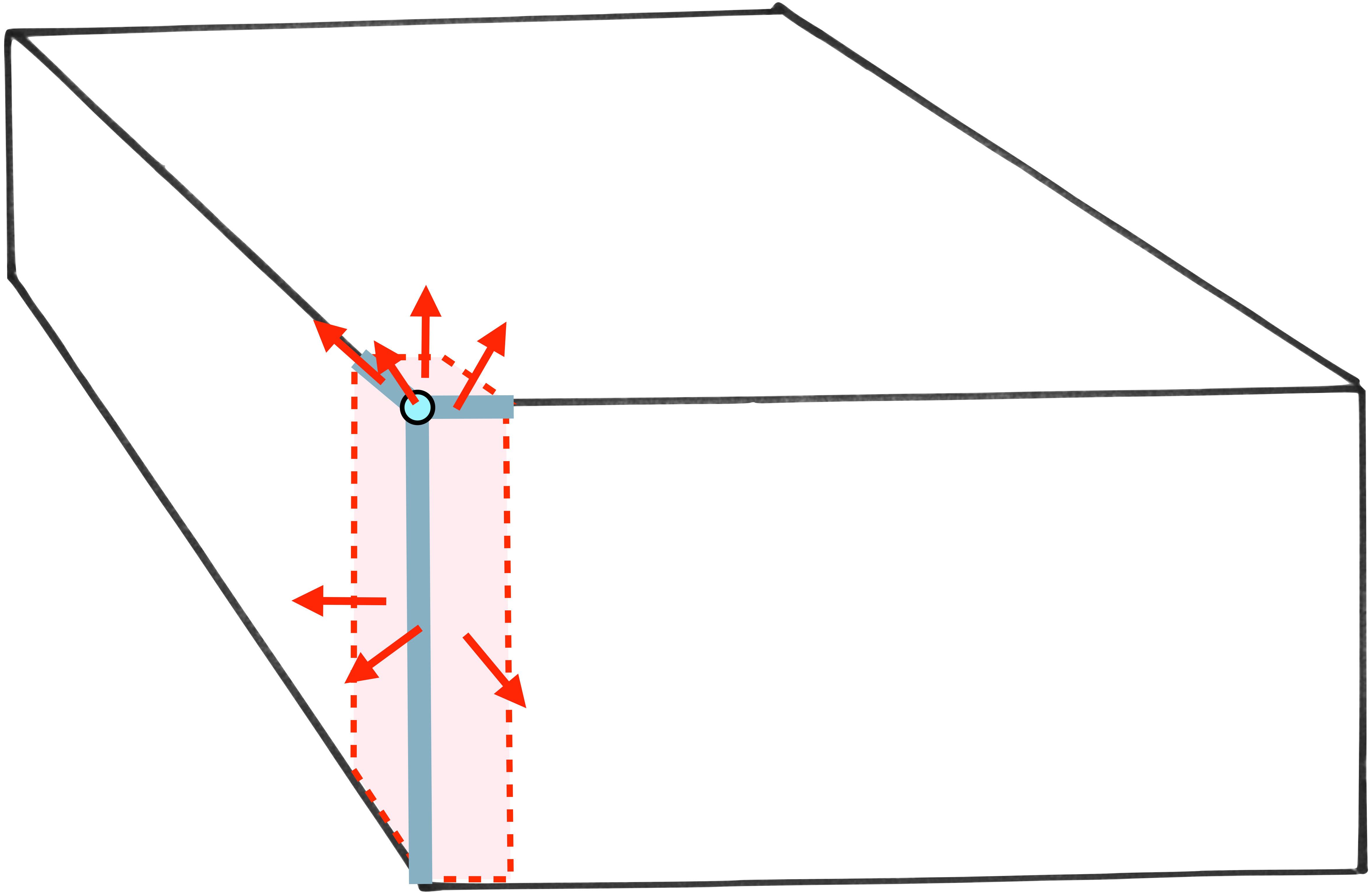
bottom-left corner

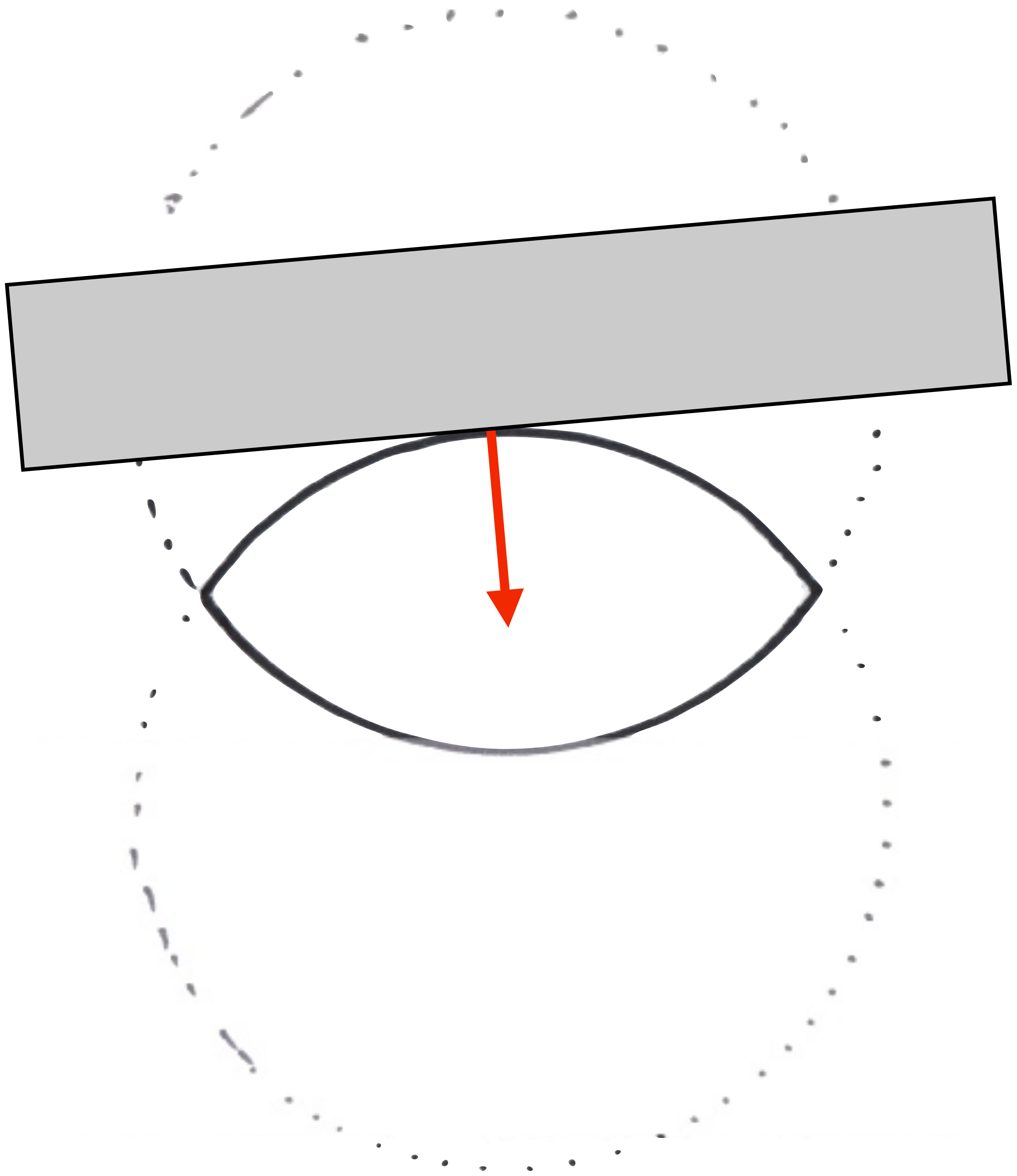
bottom side

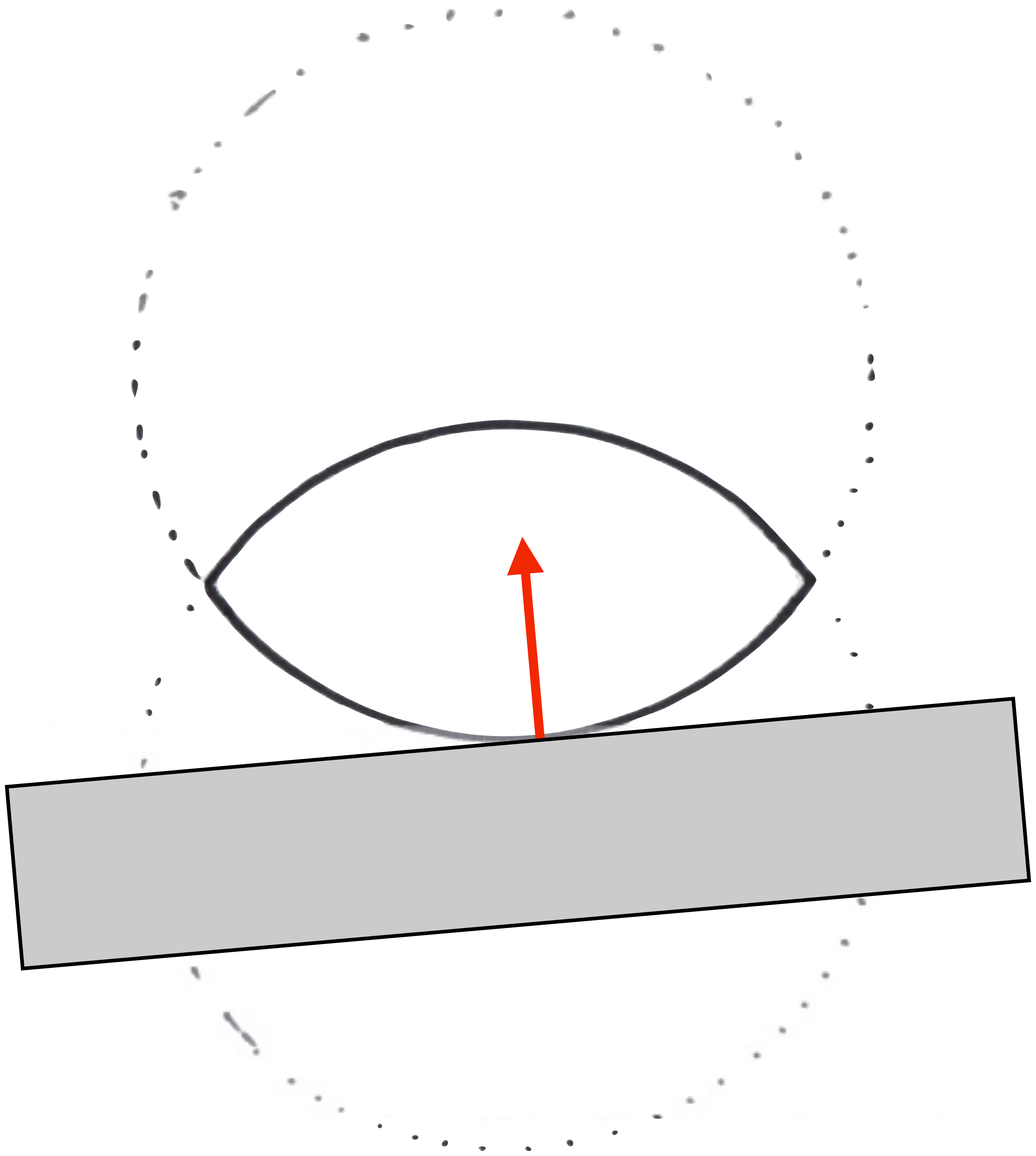
bottom-right corner

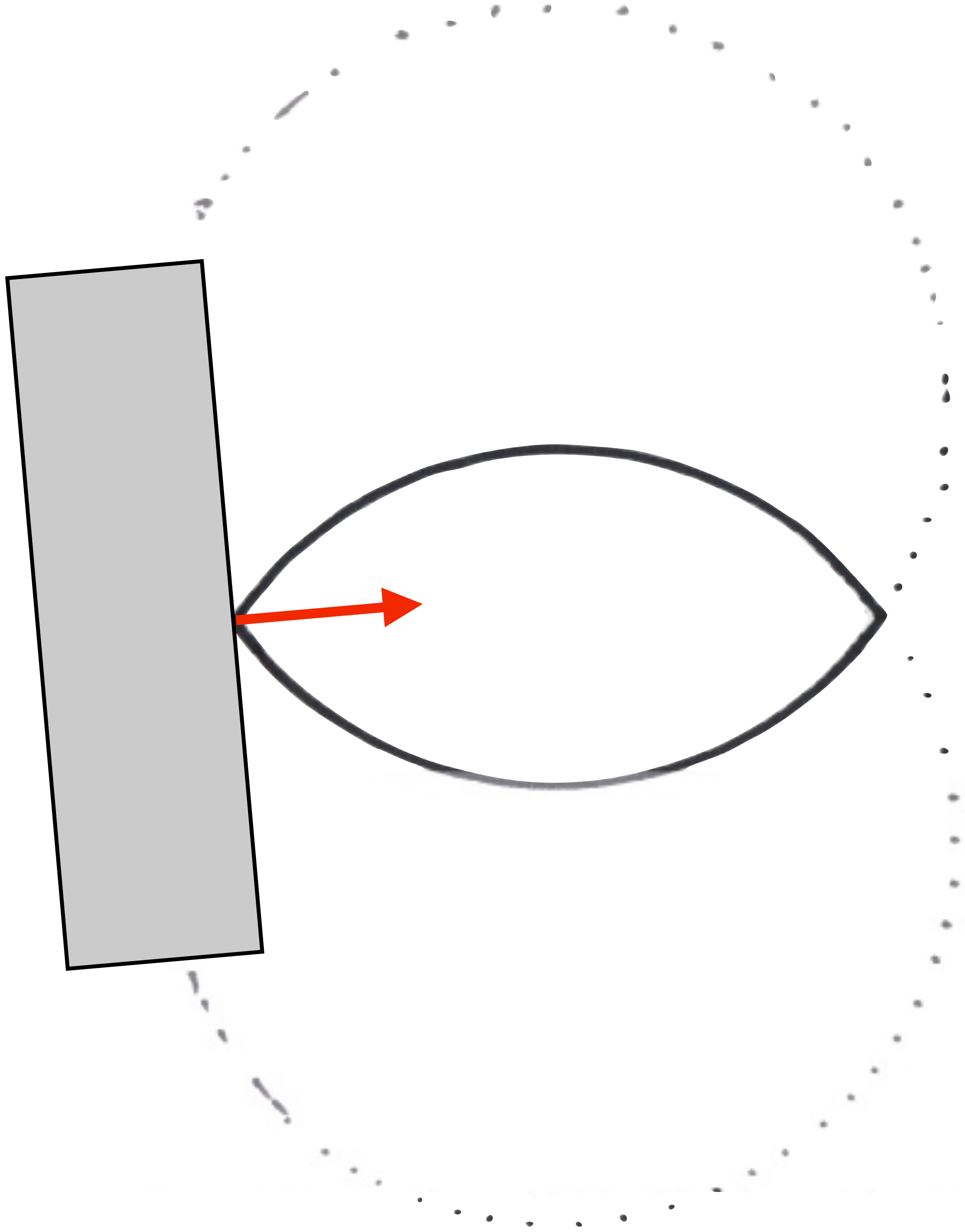




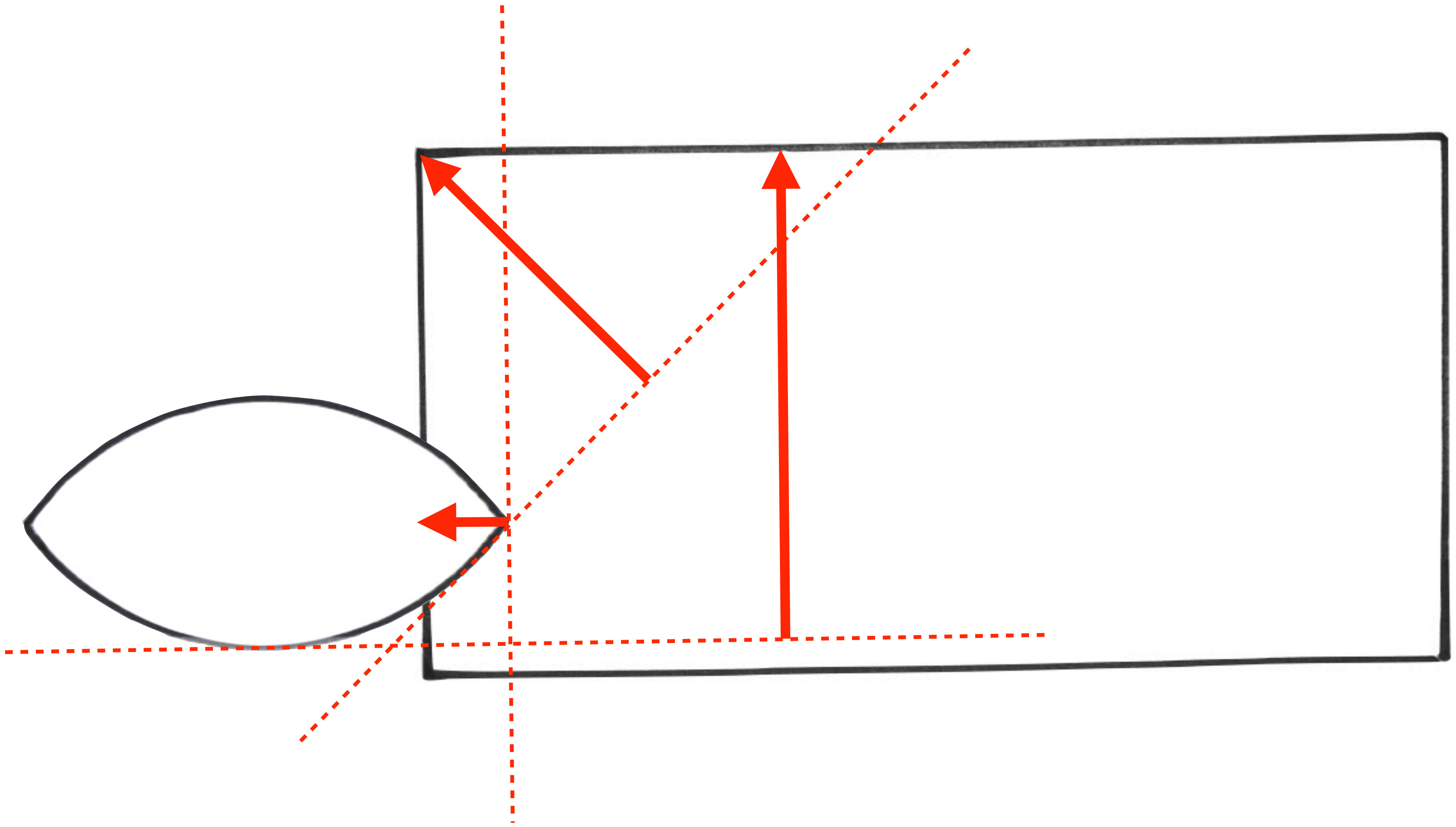




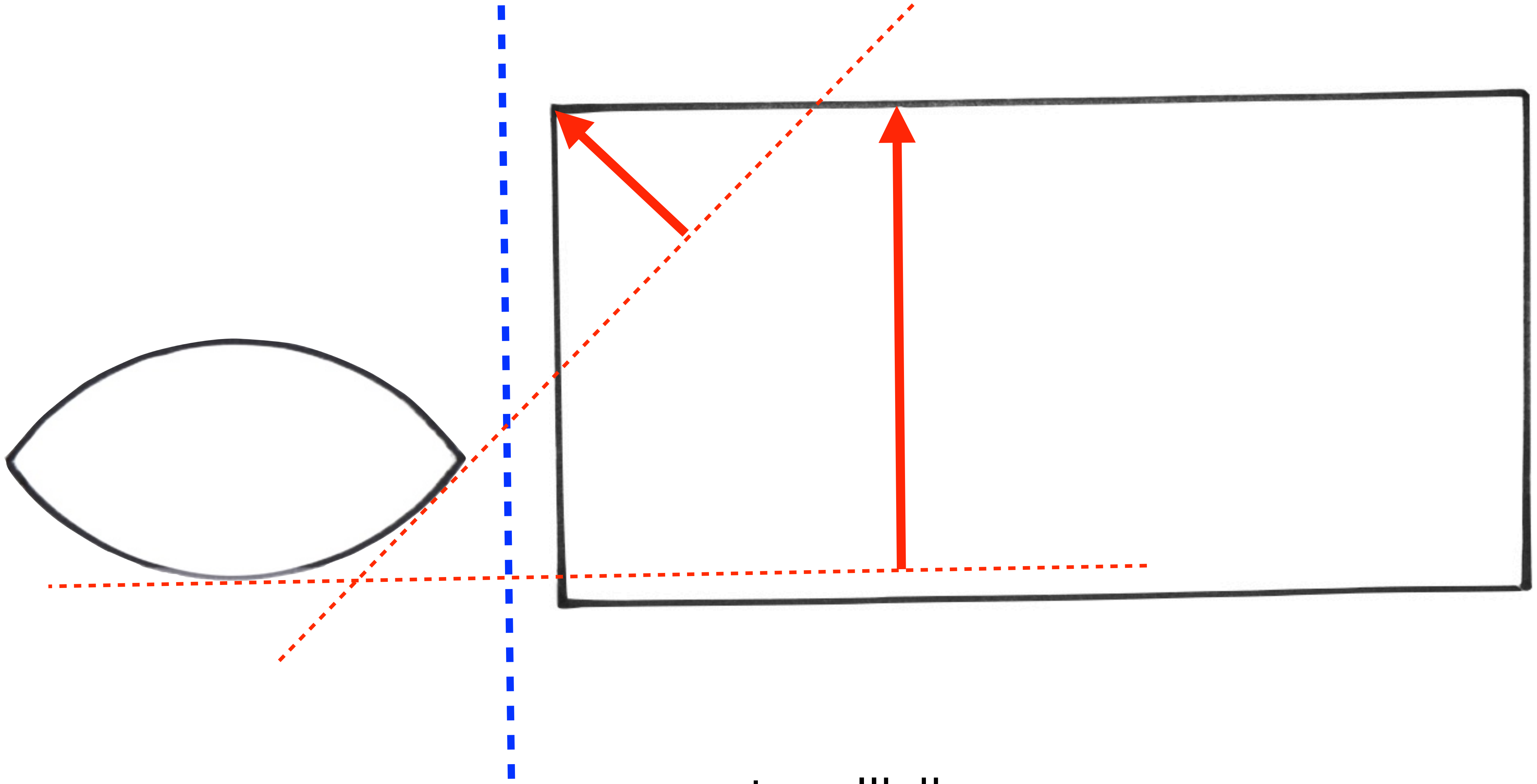




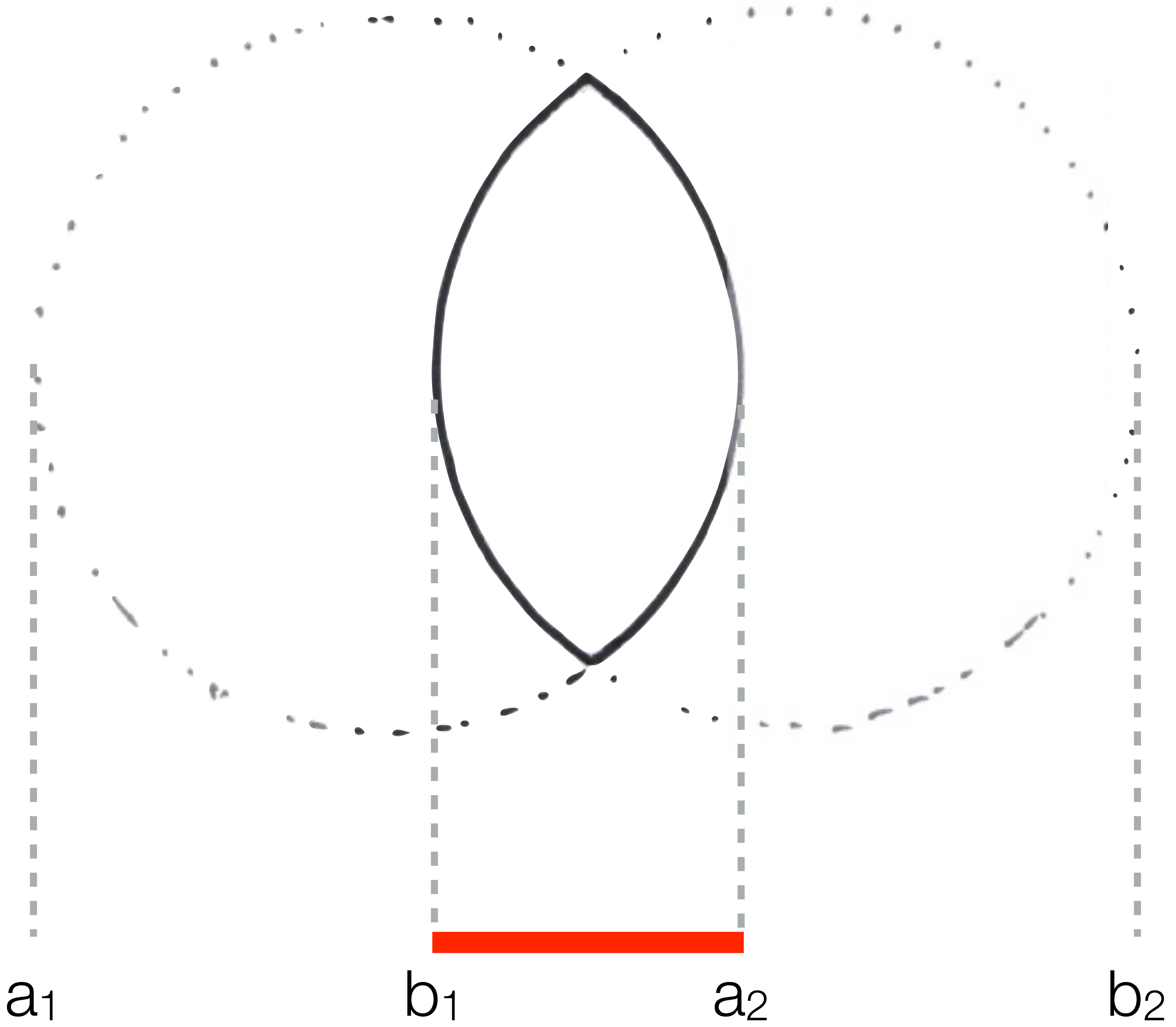
Separating Axis Test

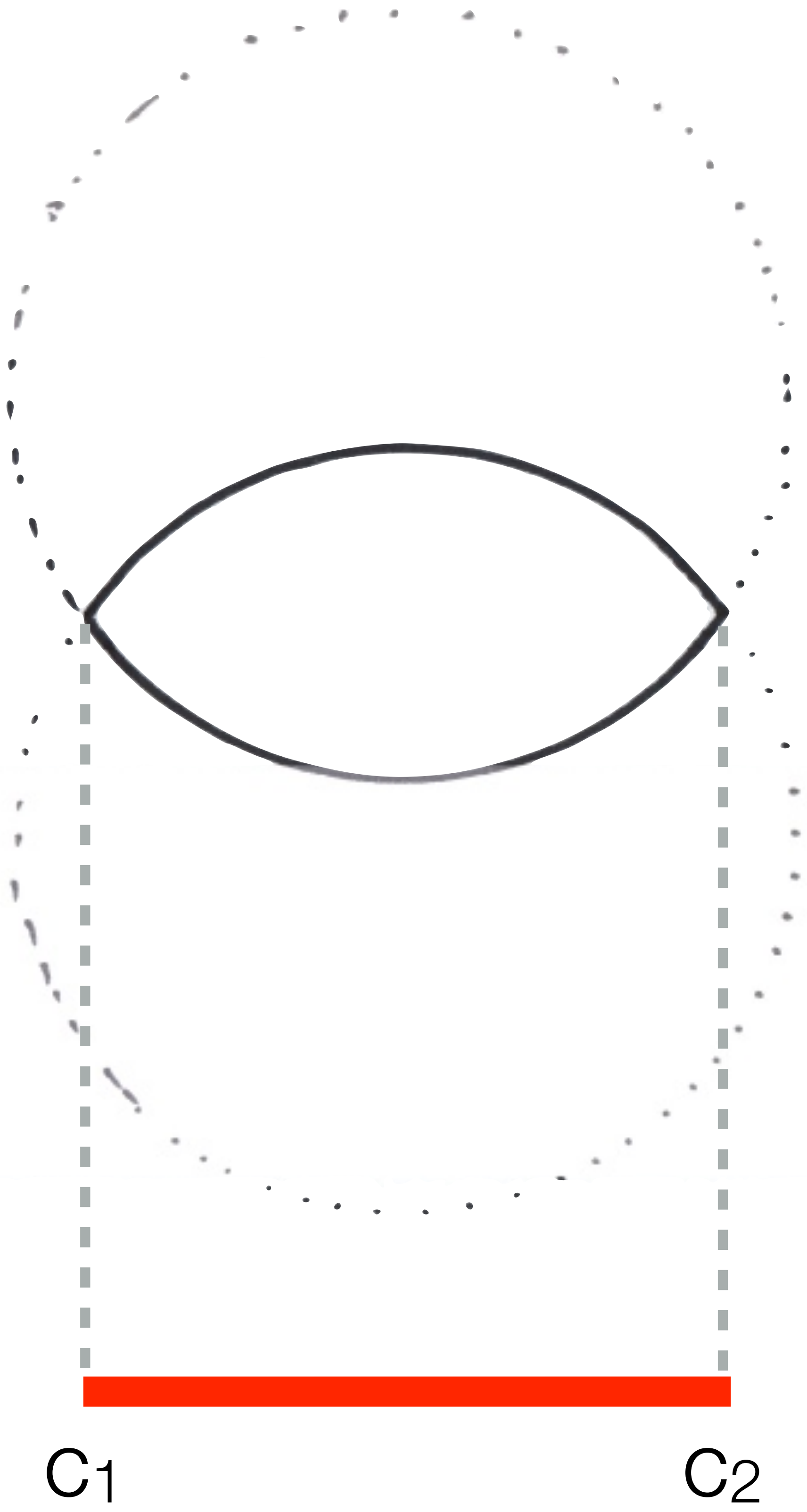


colliding



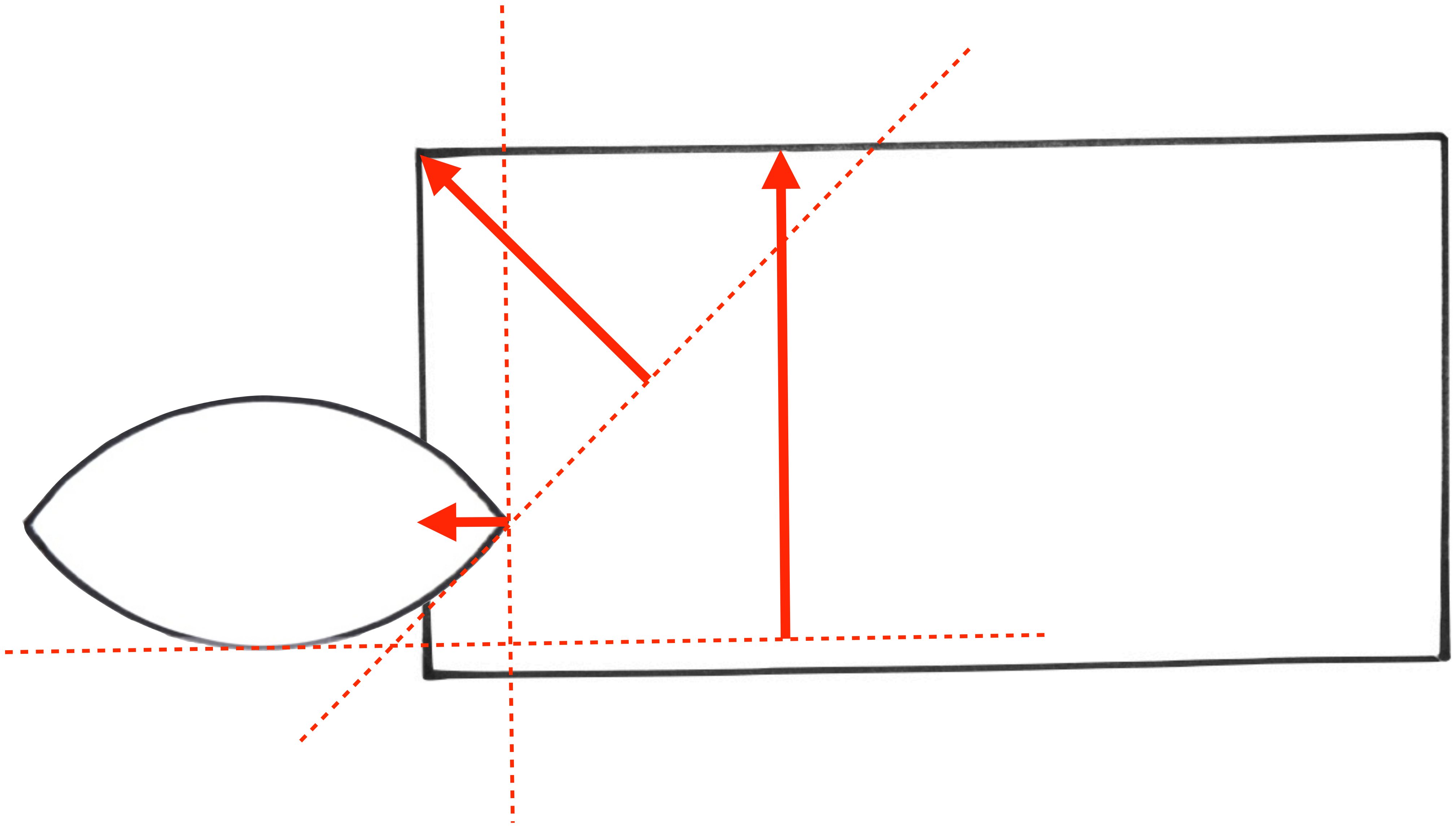
not colliding


a_1
b_1
a_2
b_2

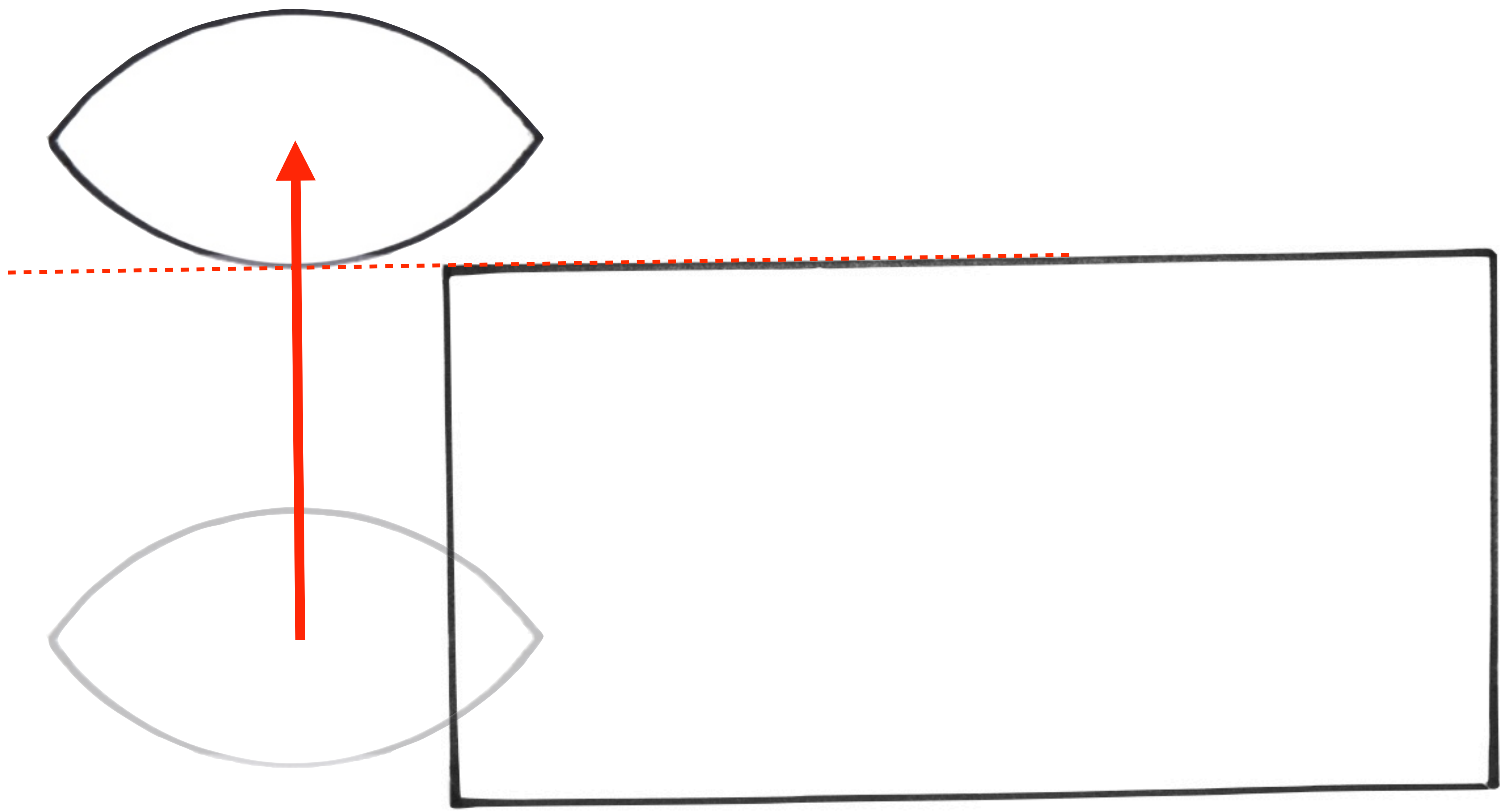


C1
C2

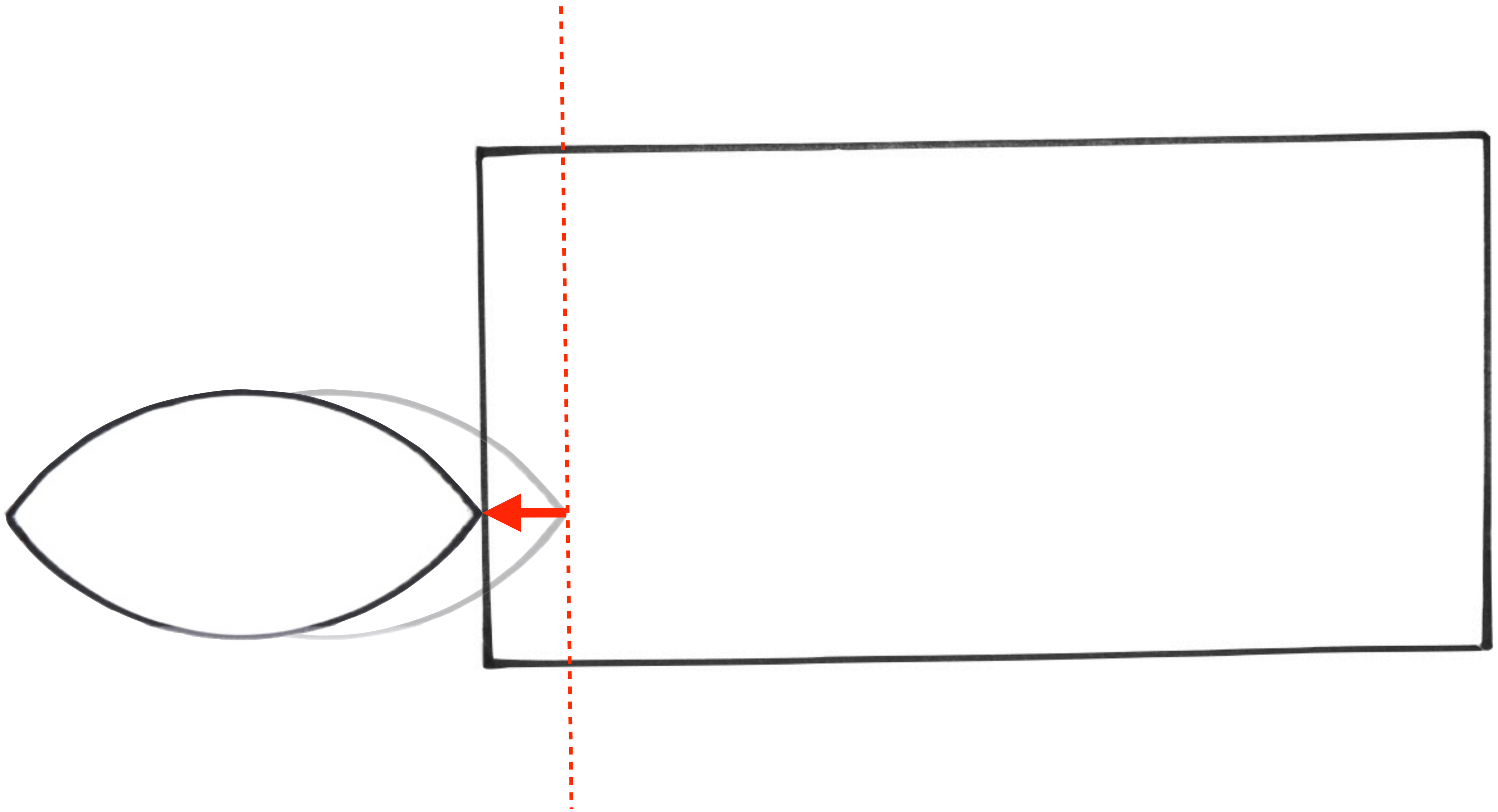
Support demo



which axis to push out along?

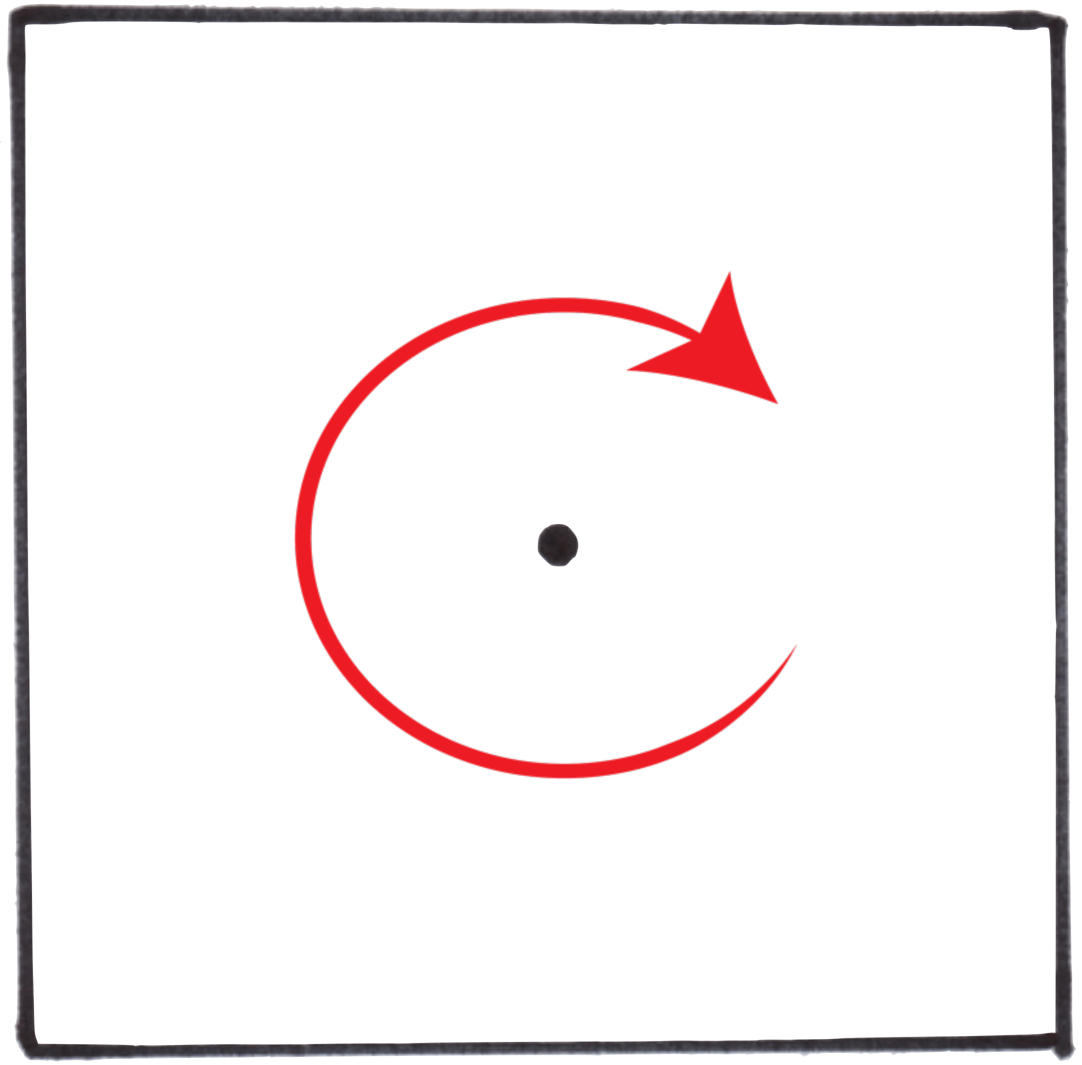


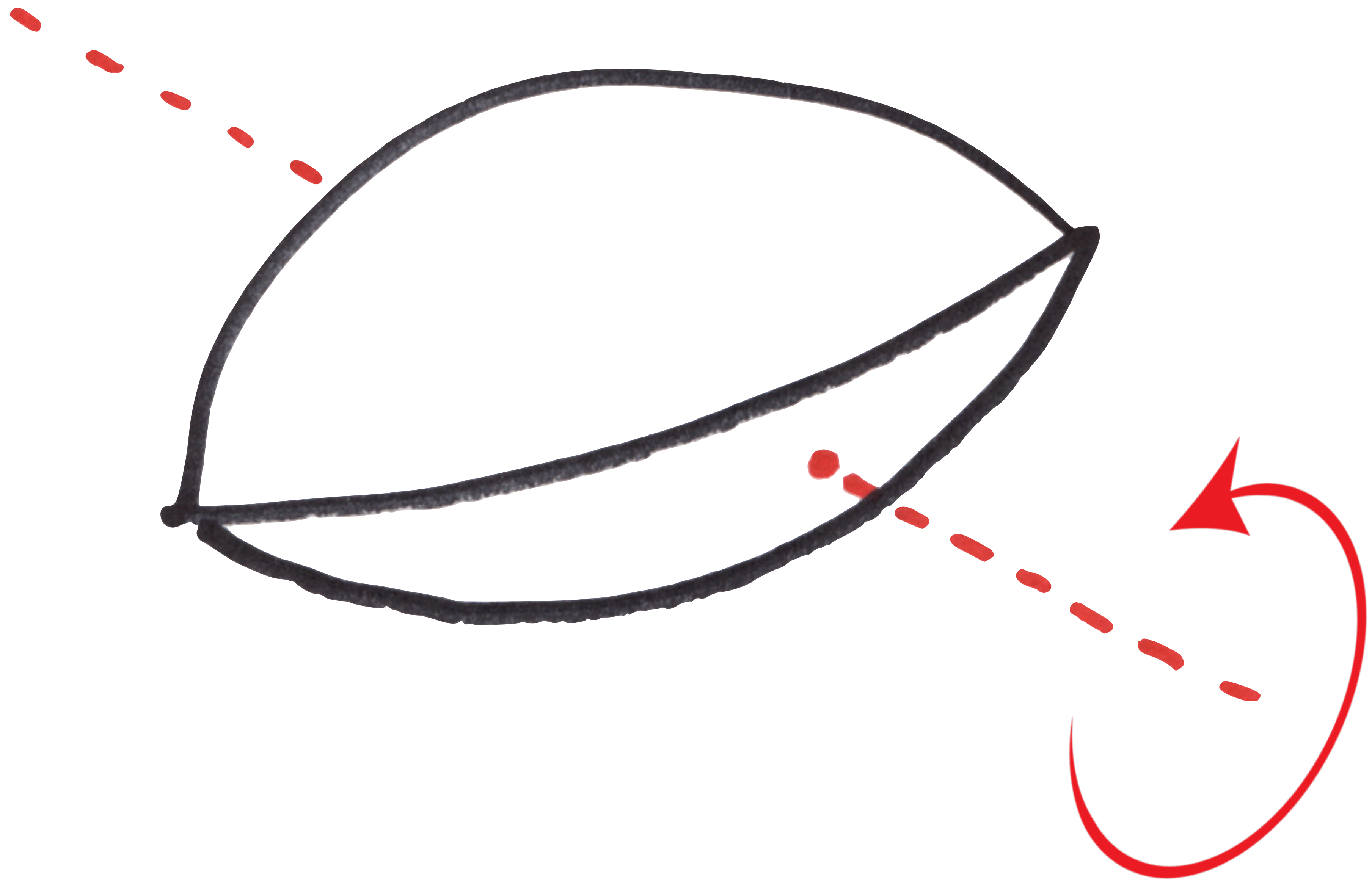
axis of <u>most</u> penetration



axis of <u>least</u> penetration

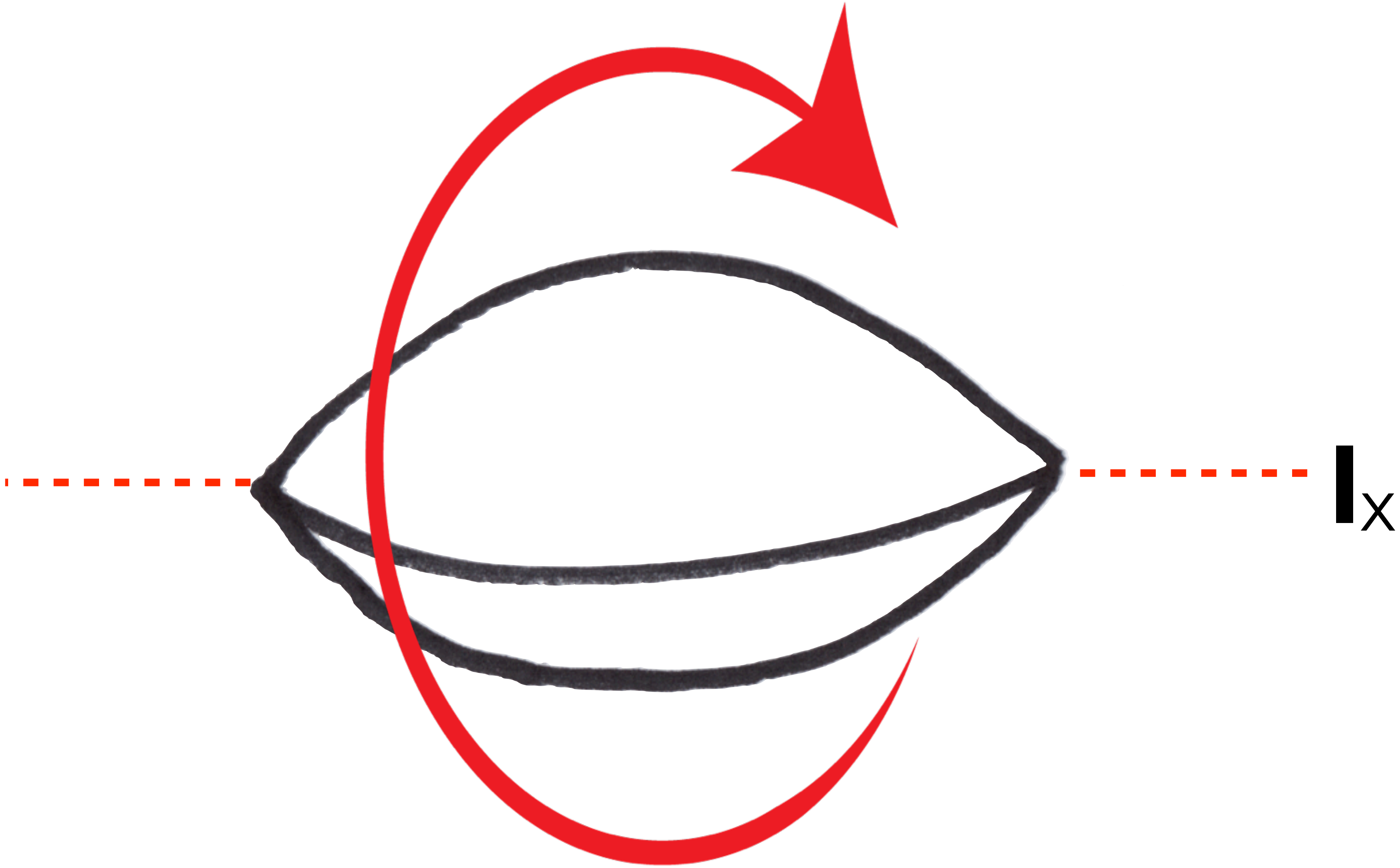
Inertia tensor

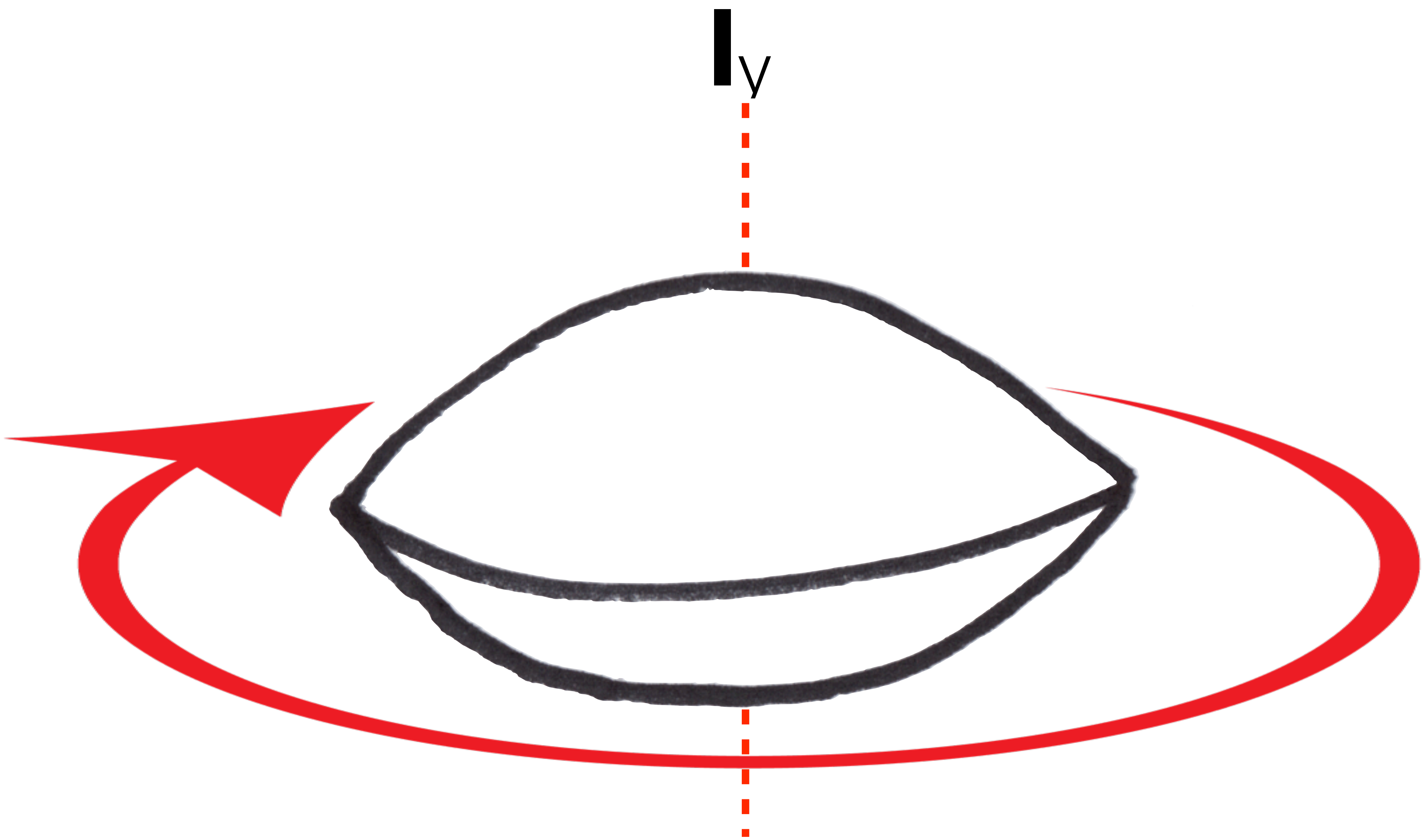


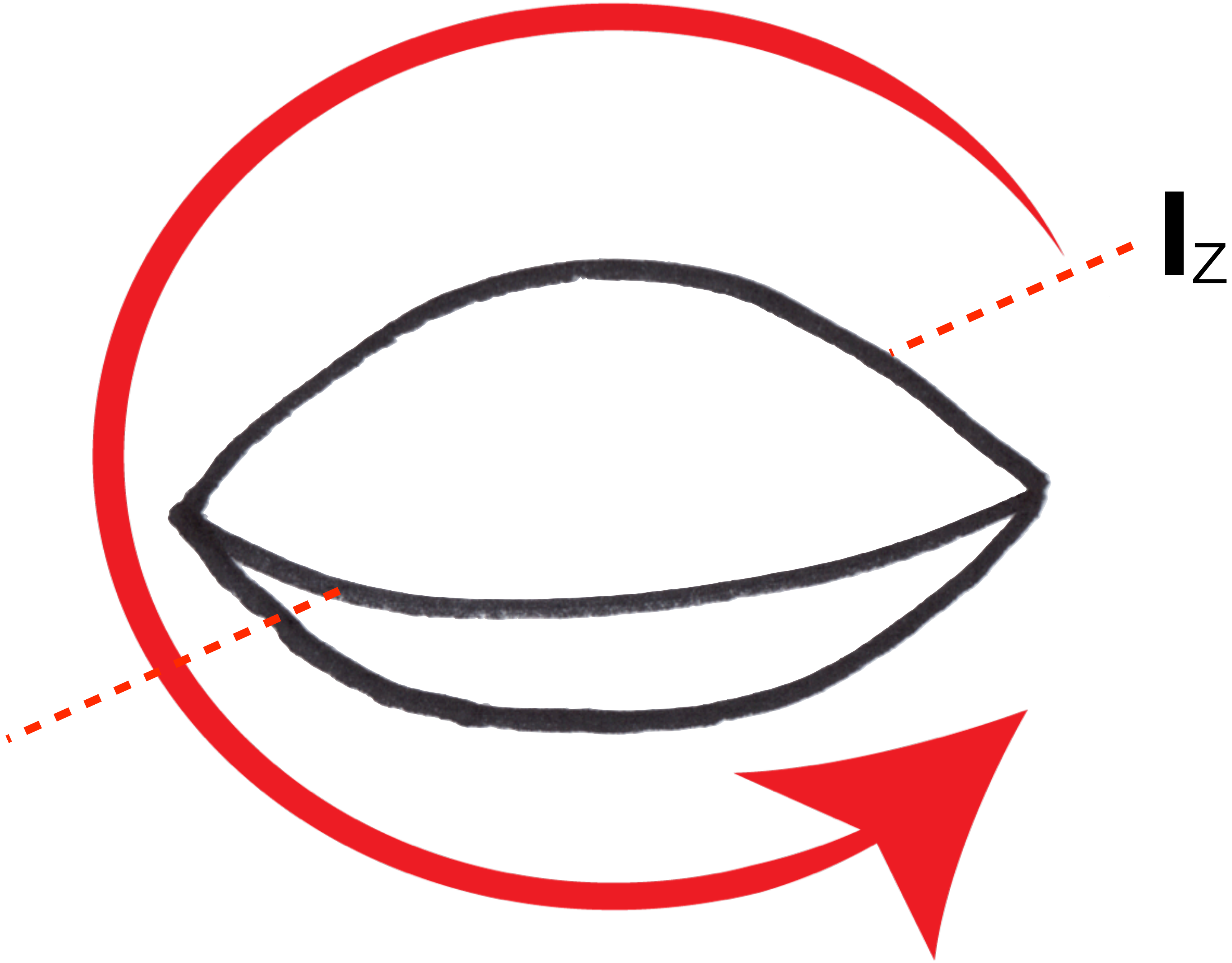


$$\boldsymbol{I} = \begin{bmatrix} I_{xx} & I_{yx} & I_{zx} \\ I_{xy} & I_{yy} & I_{zy} \\ I_{xz} & I_{yz} & I_{zz} \end{bmatrix}$$

$$\boldsymbol{I} = \begin{bmatrix} I_x & 0 & 0 \\ 0 & I_y & 0 \\ 0 & 0 & I_z \end{bmatrix}$$


I_x

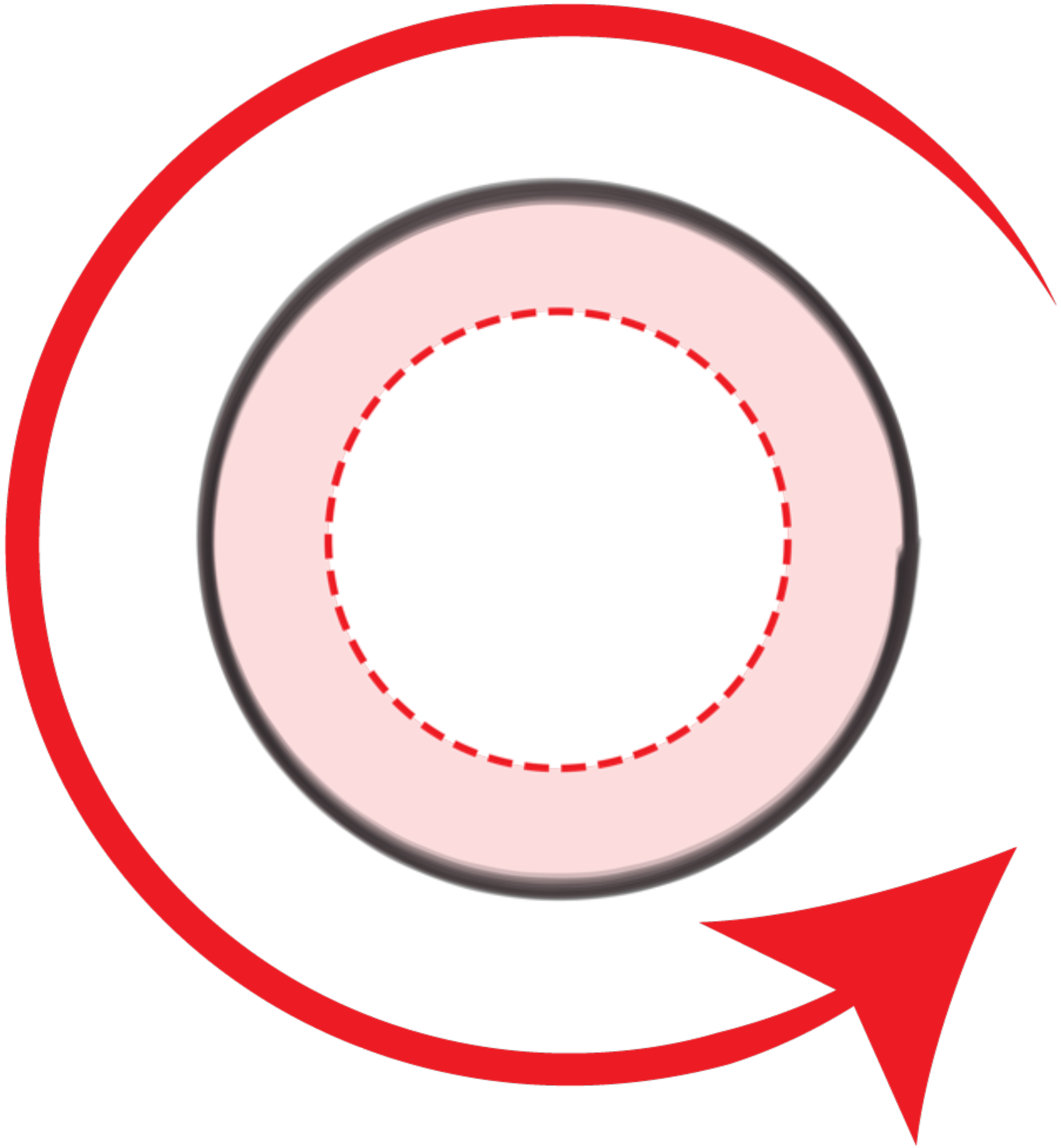

$\mathbf{I}_y$

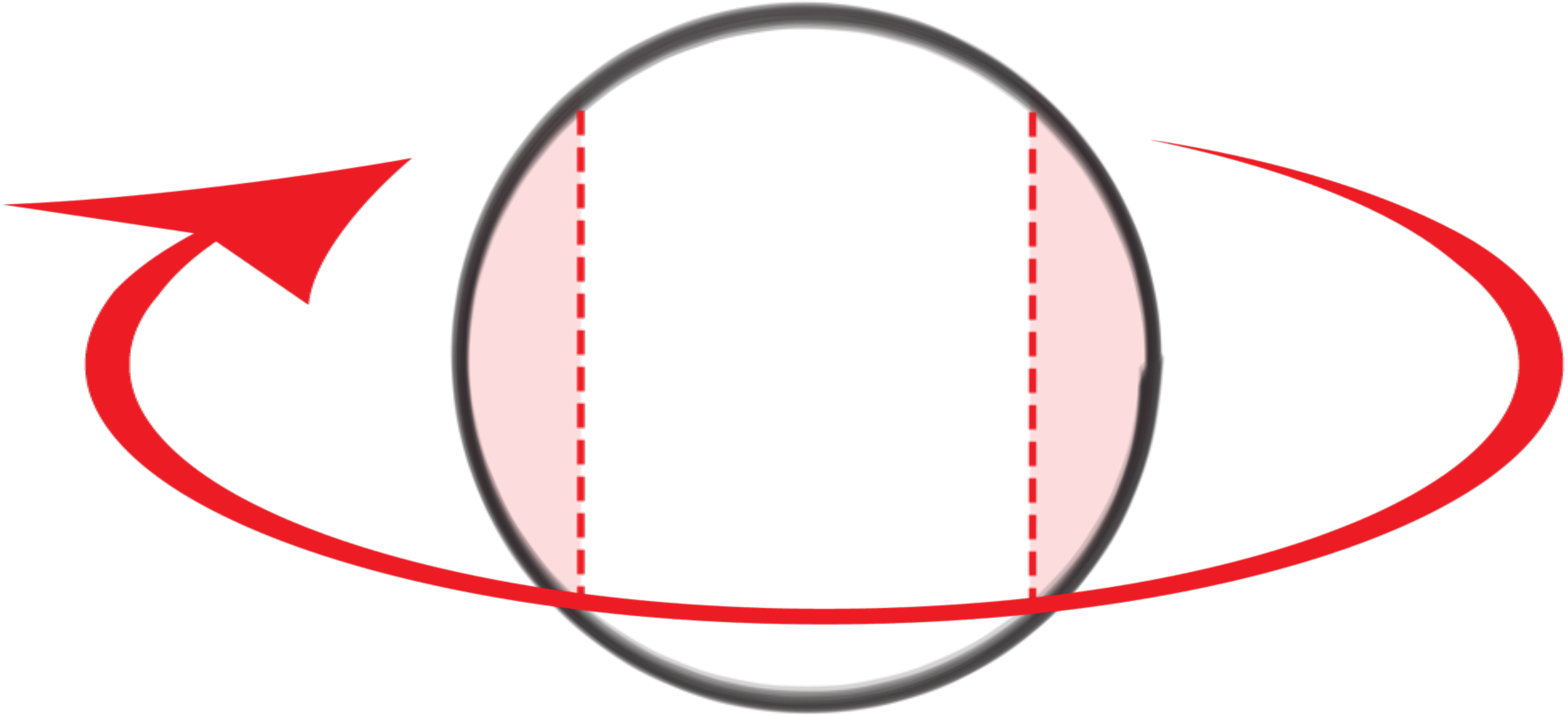


I_z

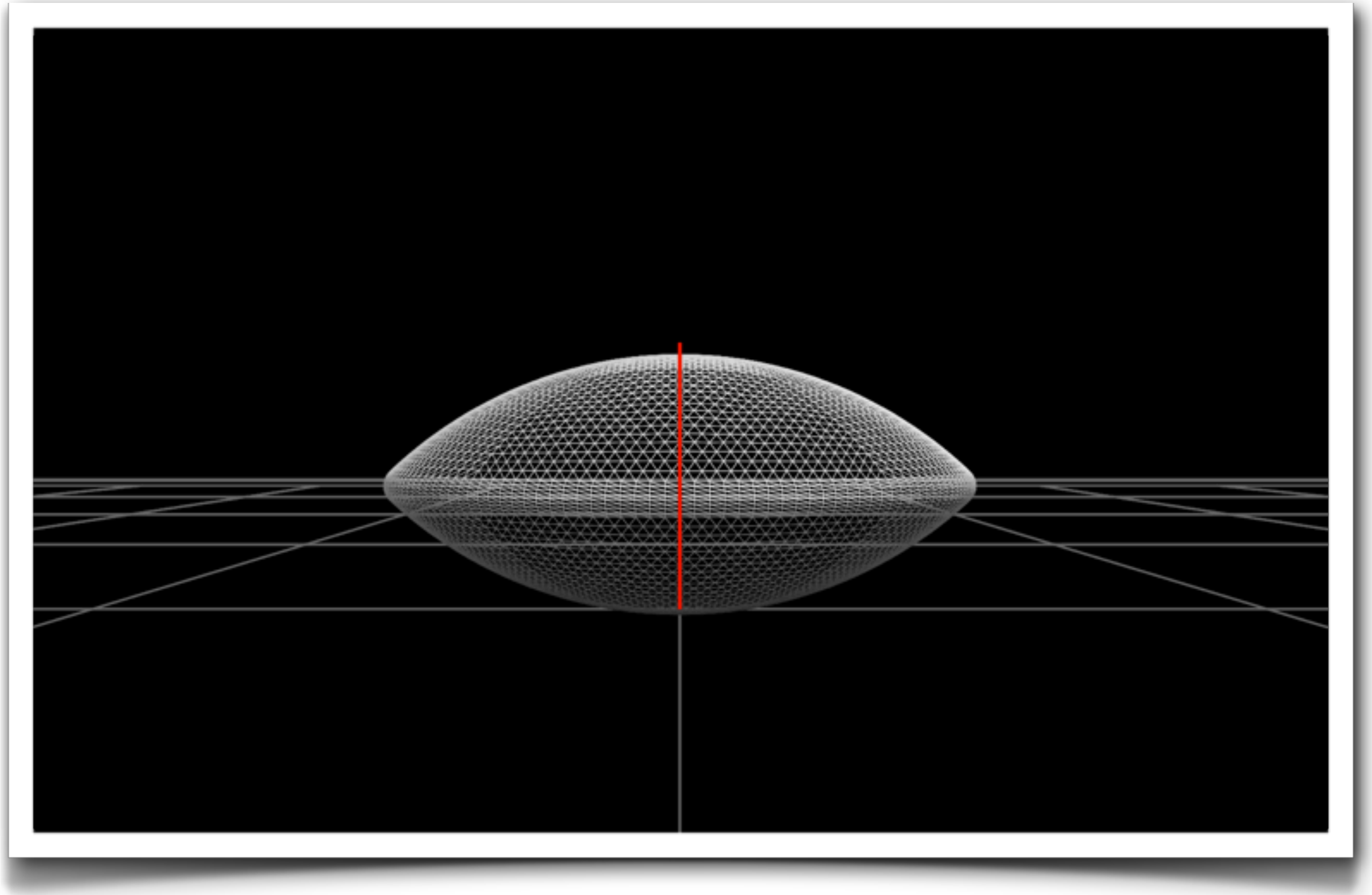
33号（9.2ミリ）

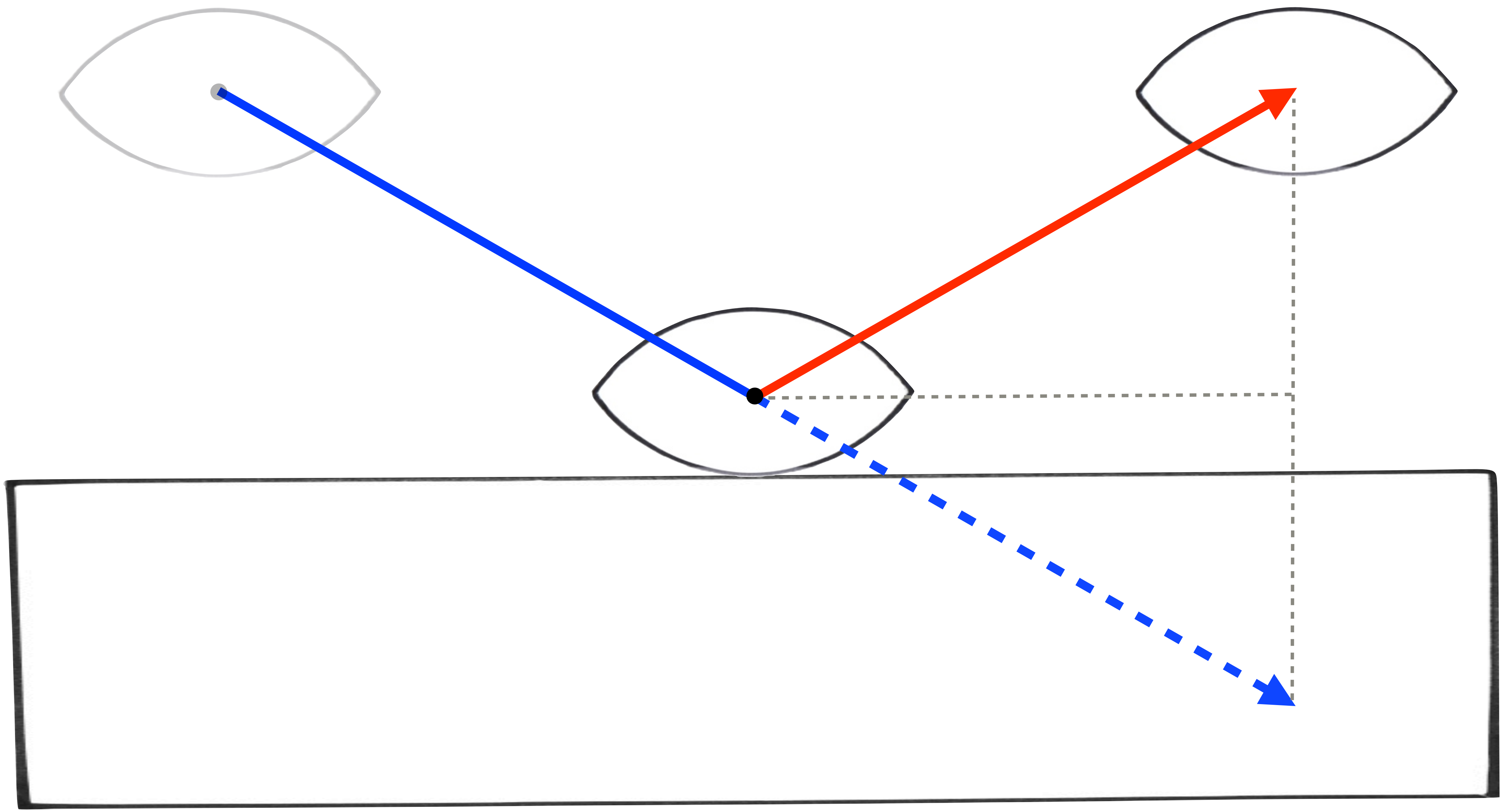
$$\mathbf{I} = \begin{bmatrix} 0.177721 & 0 & 0 \\ 0 & 0.304776 & 0 \\ 0 & 0 & 0.177721 \end{bmatrix}$$

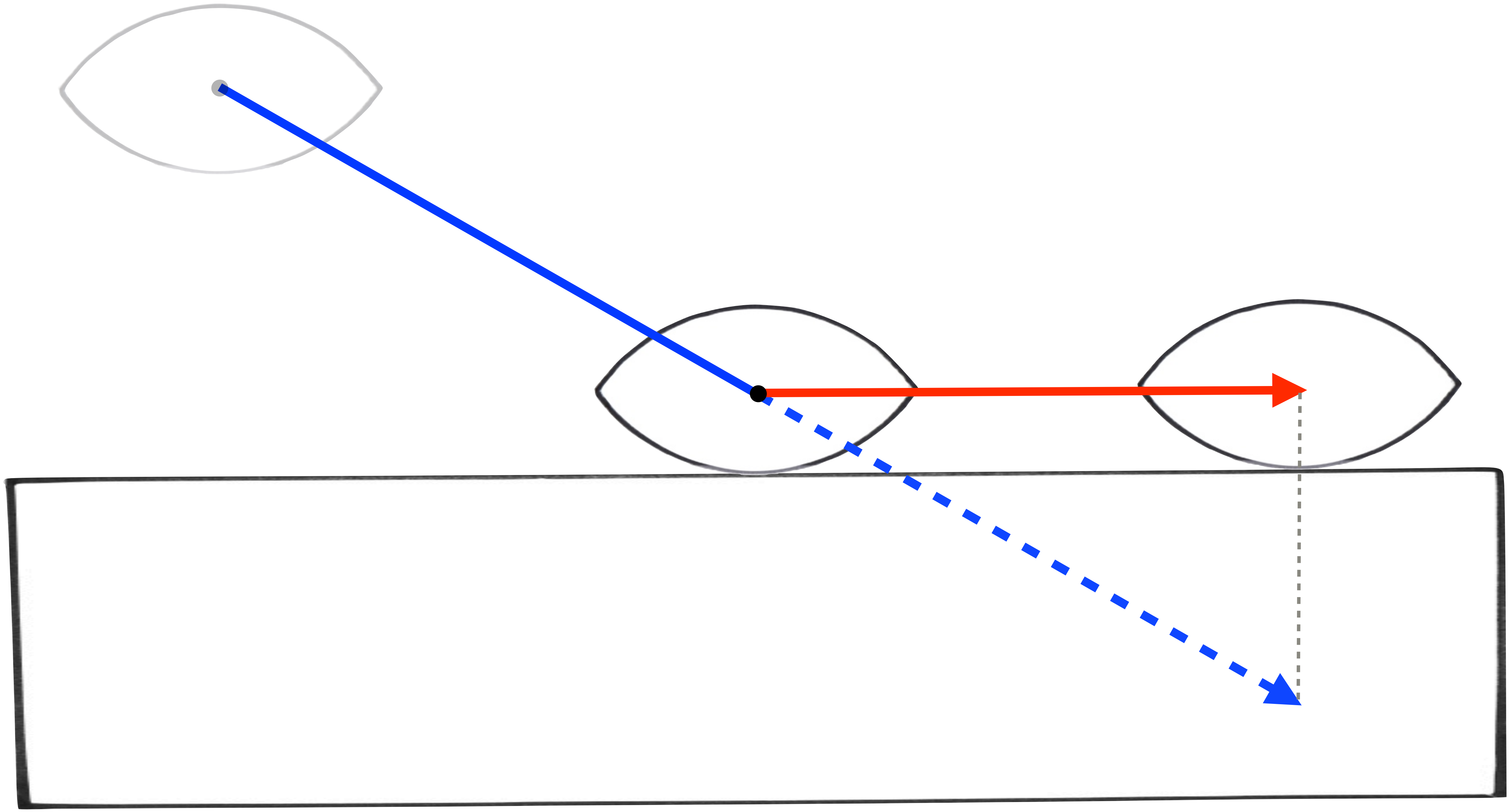


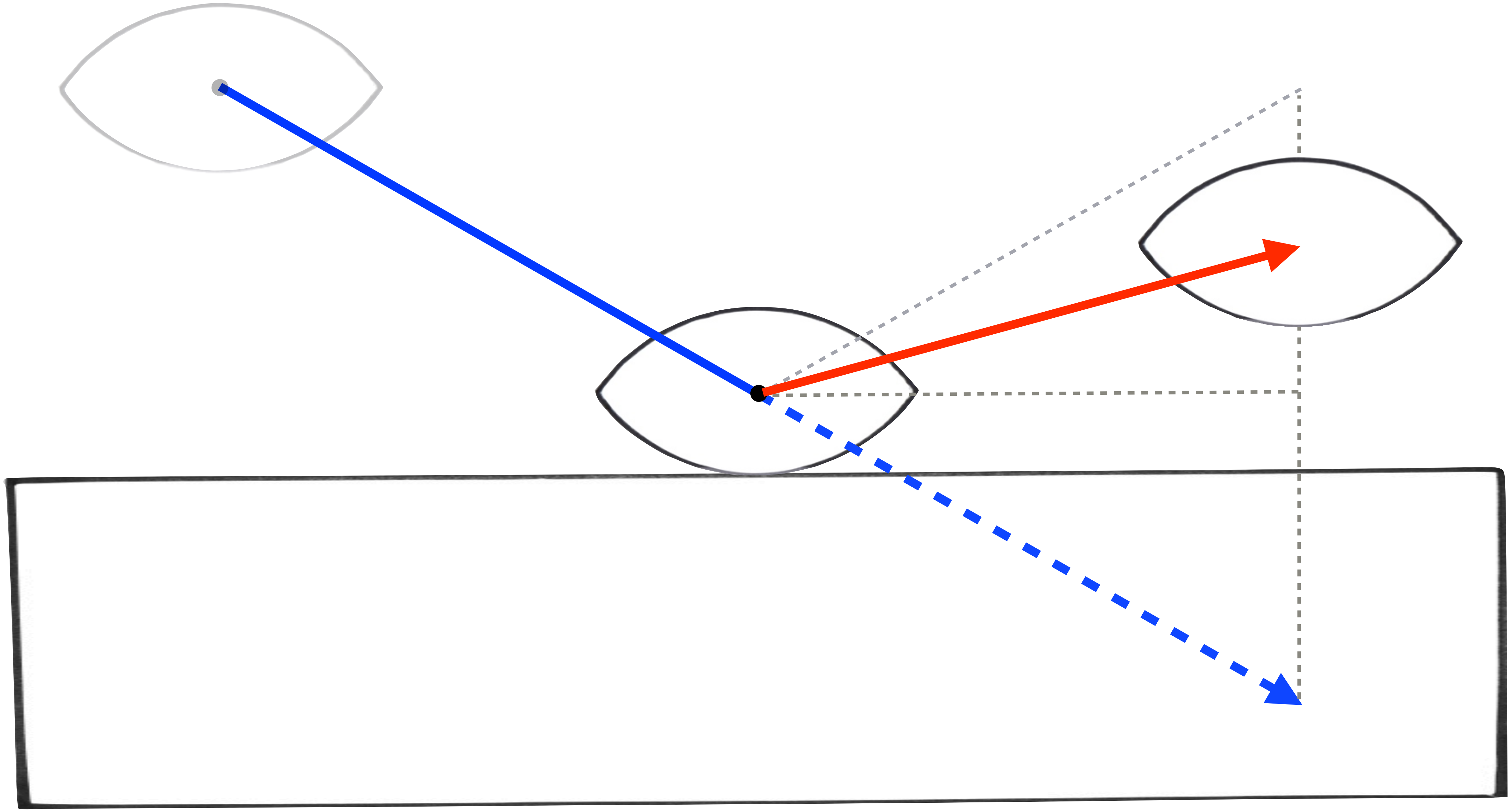


Collision response



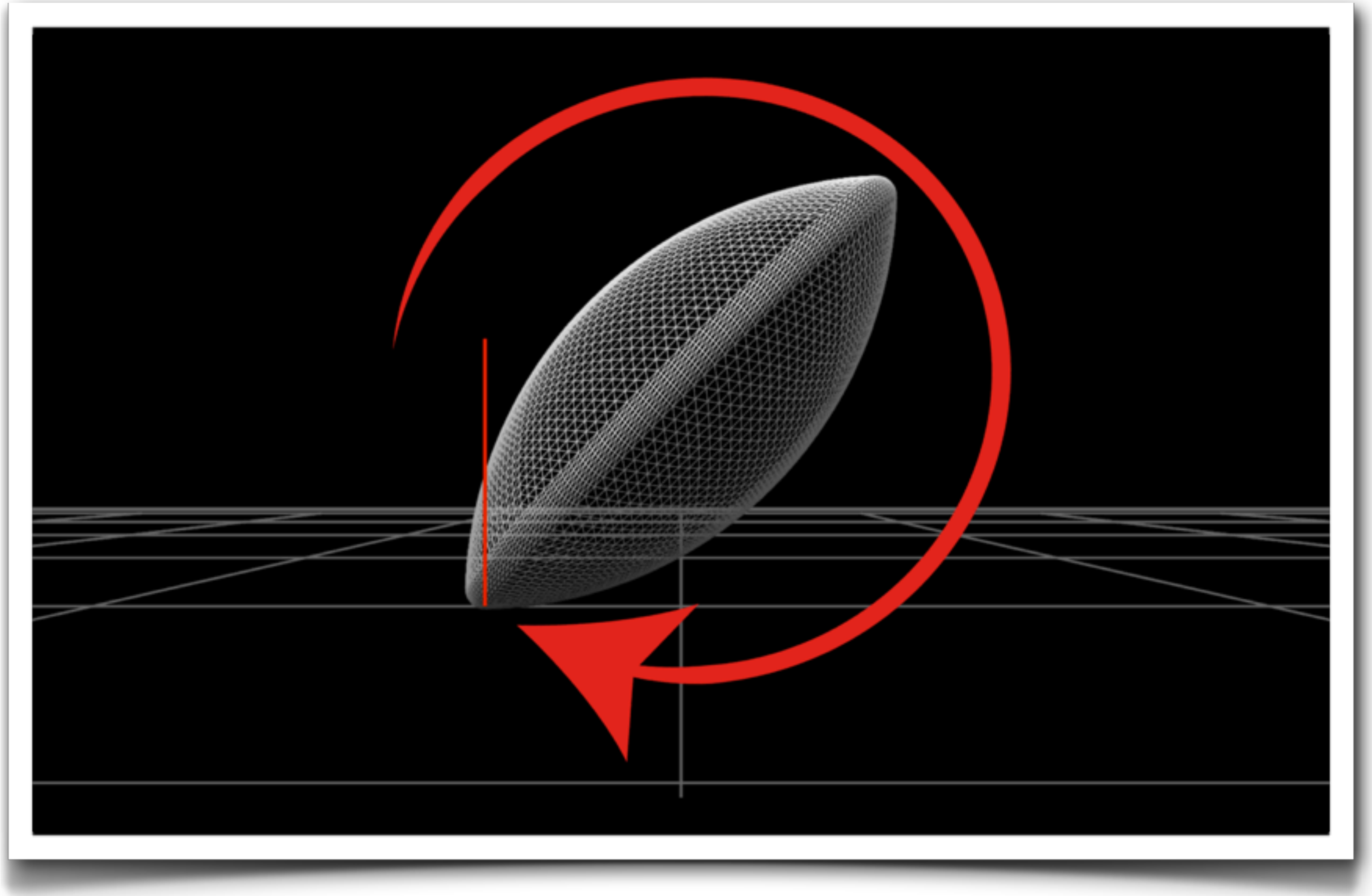






$$j = -(1 + e)\boldsymbol{p} \cdot \boldsymbol{n}$$

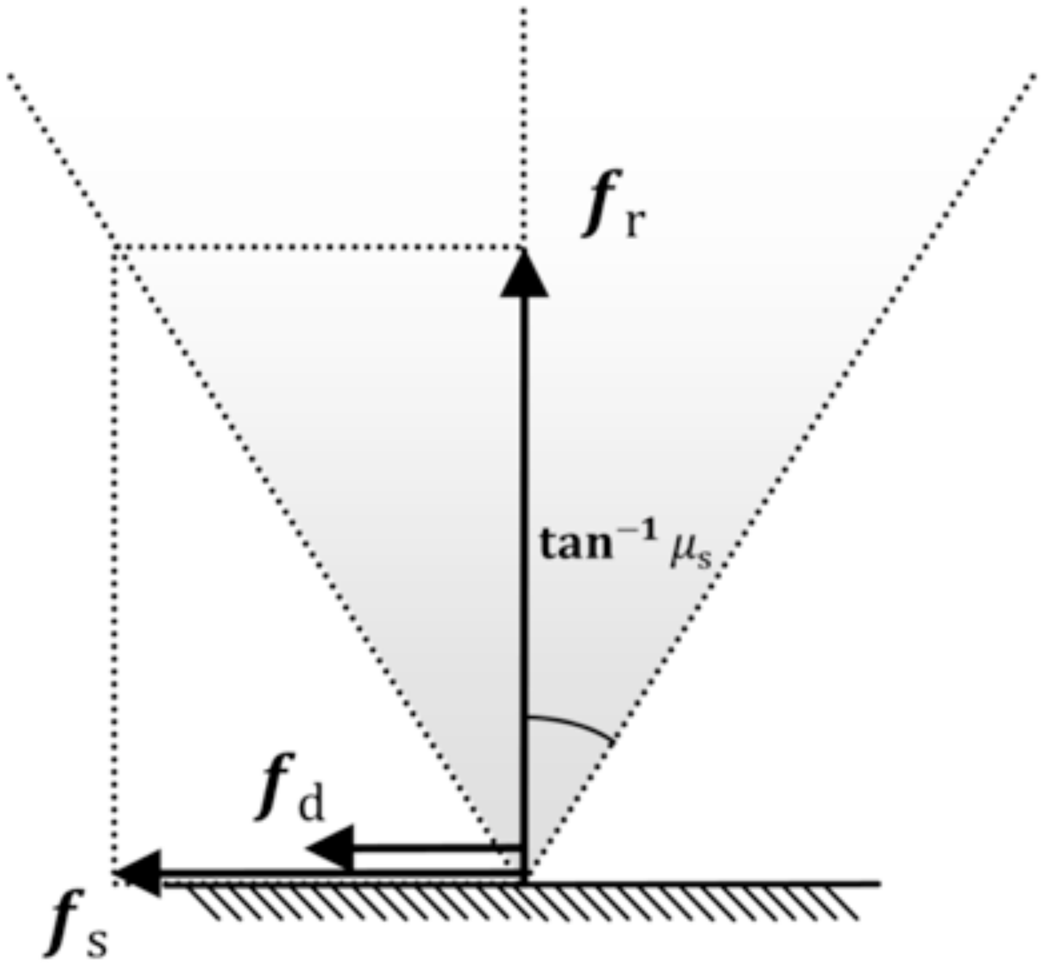
Linear collision demo



$$j = \frac{-(1+e)\boldsymbol{v_r} \cdot \boldsymbol{n}}{m_1^{-1} + m_2^{-1} + (\boldsymbol{I_1^{-1}}(\boldsymbol{r_1} \times \boldsymbol{n}) \times \boldsymbol{r_1} + \boldsymbol{I_2^{-1}}(\boldsymbol{r_2} \times \boldsymbol{n}) \times \boldsymbol{r_2}) \cdot \boldsymbol{n}}$$

$$j = \frac{-(1+e)\boldsymbol{v} \cdot \boldsymbol{n}}{m^{-1} + (\boldsymbol{I^{-1}}(\boldsymbol{r} \times \boldsymbol{n}) \times \boldsymbol{r}) \cdot \boldsymbol{n}}$$

Angular collision demo


$\boldsymbol{f}_{\mathrm{r}}$
$\tan^{-1}\mu_{\mathrm{s}}$
$\boldsymbol{f}_{\mathrm{d}}$
$\boldsymbol{f}_{\mathrm{s}}$

$$j_t = \frac{-\boldsymbol{v} \cdot \boldsymbol{t}}{m^{-1} + (\boldsymbol{I}^{-1}(\boldsymbol{r} \times \boldsymbol{t}) \times \boldsymbol{r}) \cdot \boldsymbol{t}}$$

Friction demo

Thank you

Glenn Fiedler

www.gafferongames.com

twitter

@gafferongames