Math for Game Developers: Procedural Mesh Animation with Non-linear Transforms

Michael Austin
Hidden Path Entertainment

GAME DEVELOPERS CONFERENCE
MARCH 18–22, 2019 | #GDC19

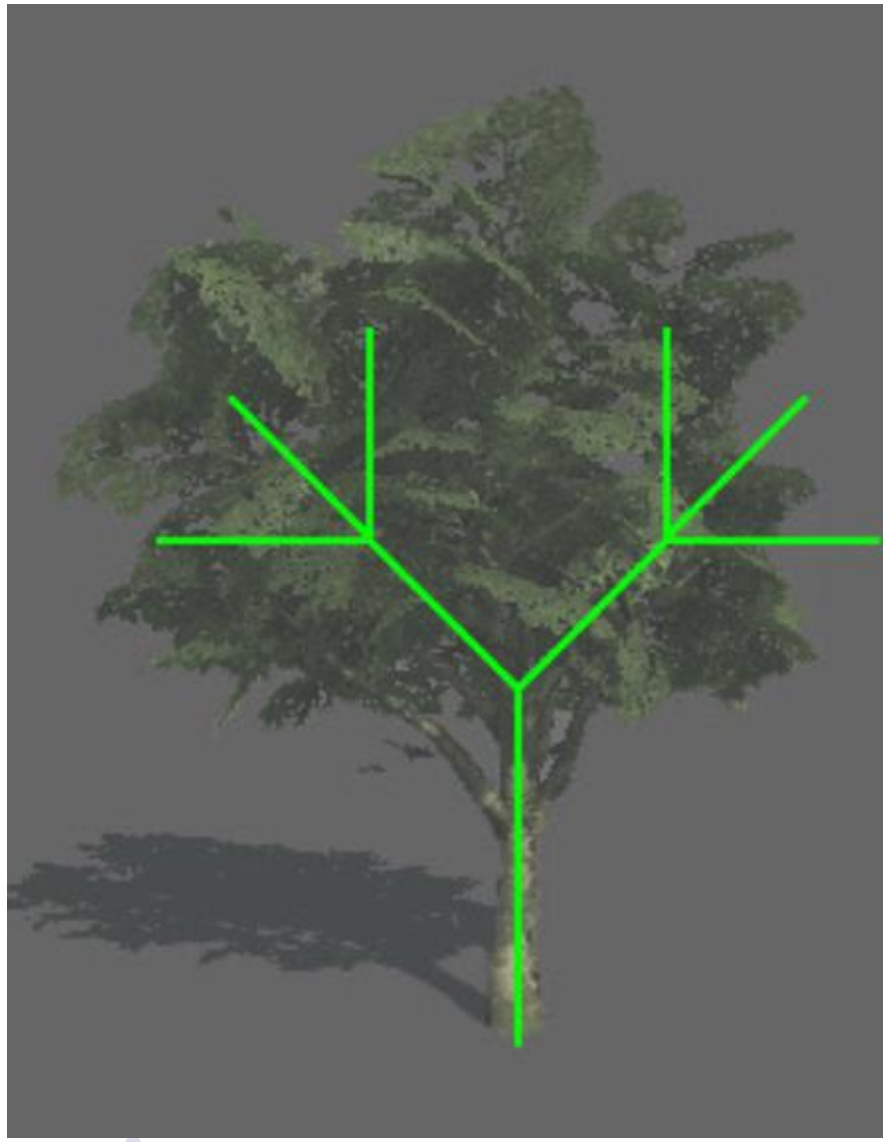
`


XNA
Microsoft® XNA™ Framework
Powerful next-generation gaming framework
Enable greater productivity writing games using C#
Share more code between Windows® and Xbox 360™ versions of your game
SAMSUNG
GDC 2006
culture
HIDDEN PATH
Microsoft
POWERVR

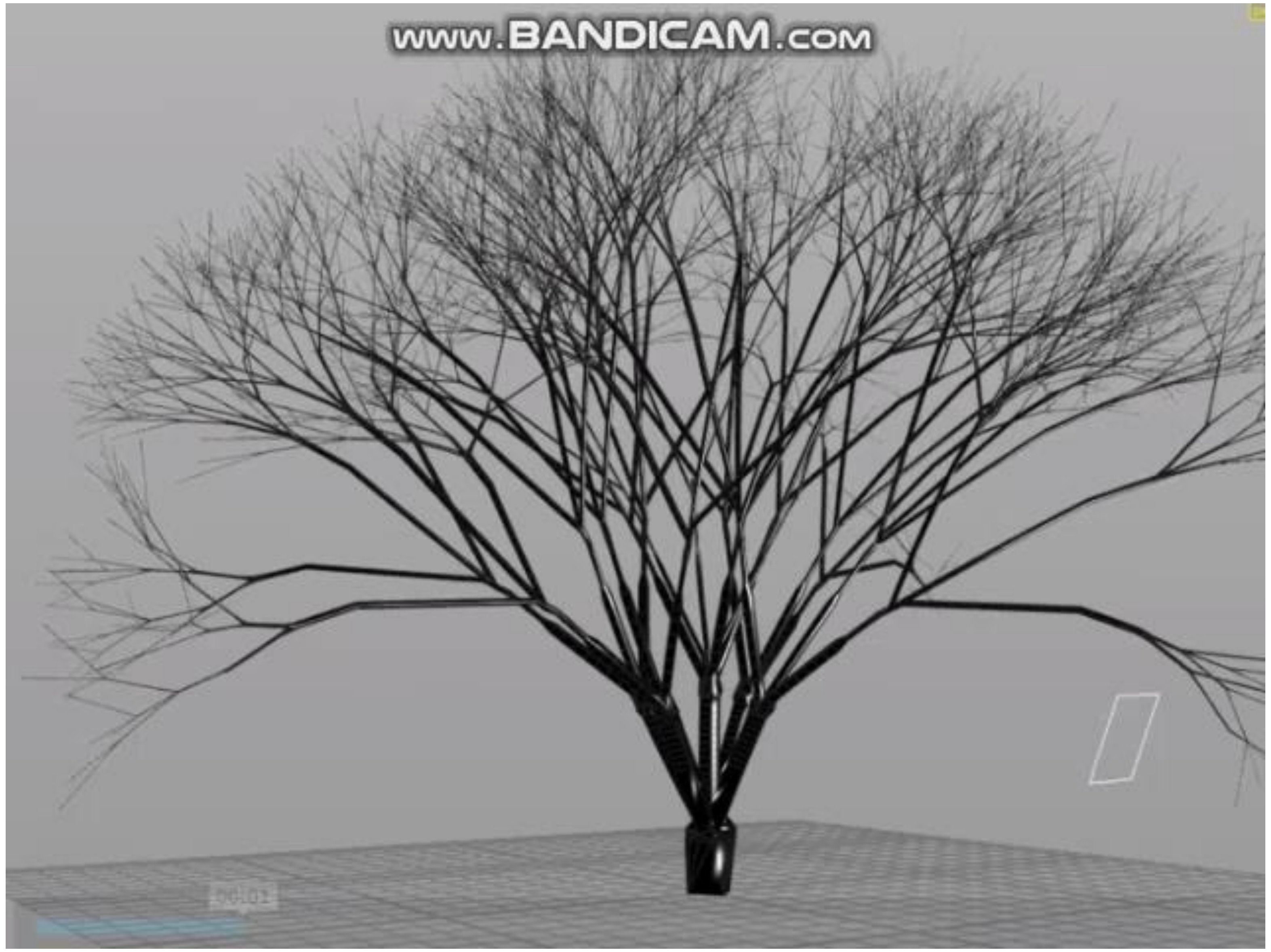
Animating in the Shader

- The vertex shader's job is to move around points (normally transforms)
- Math here is very fast
- Lets play with possibilities

Part 1: Bendy Trees and Flowers



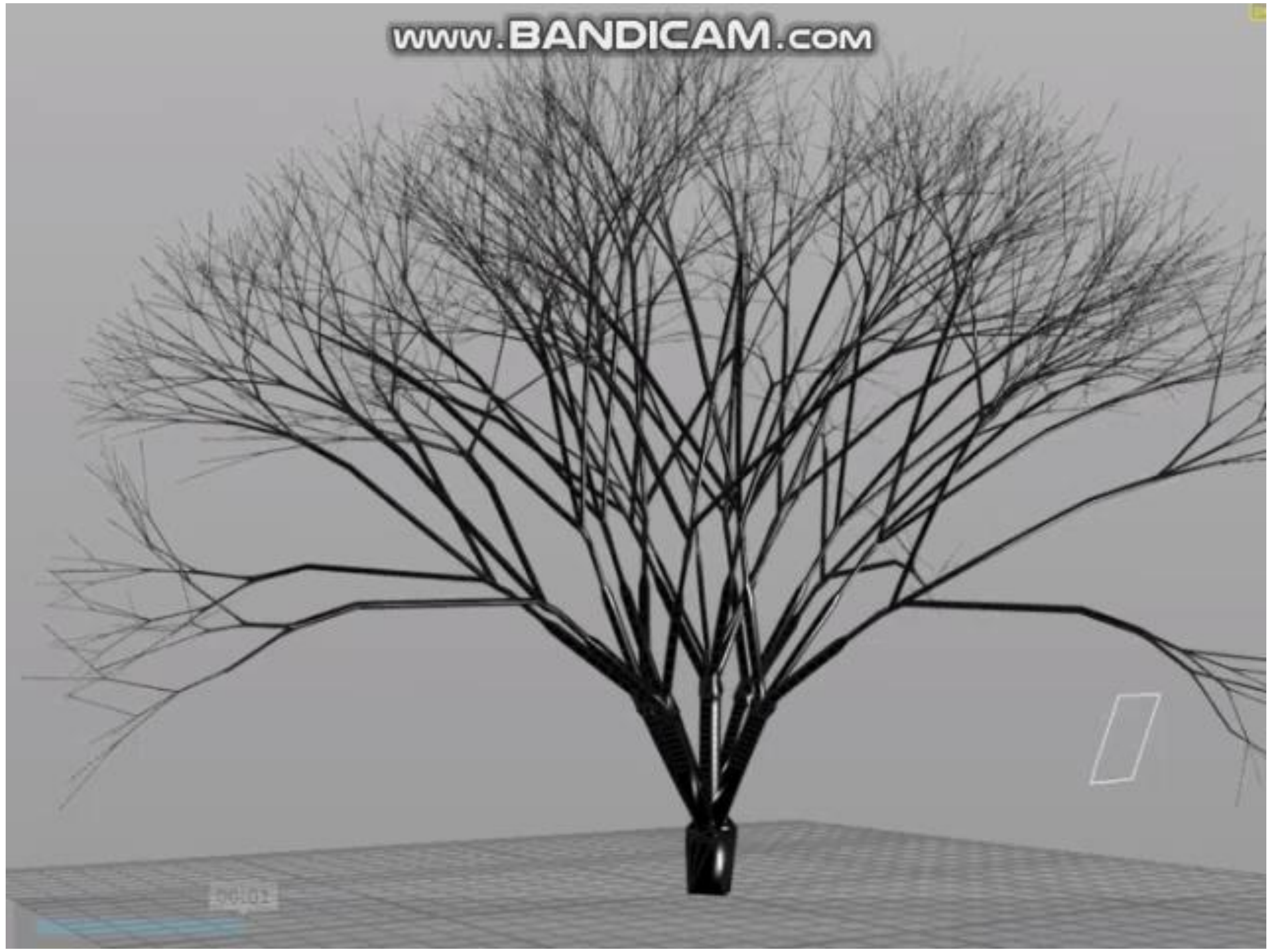
Credit: Tree Creator Tutorial Assets (Unity)



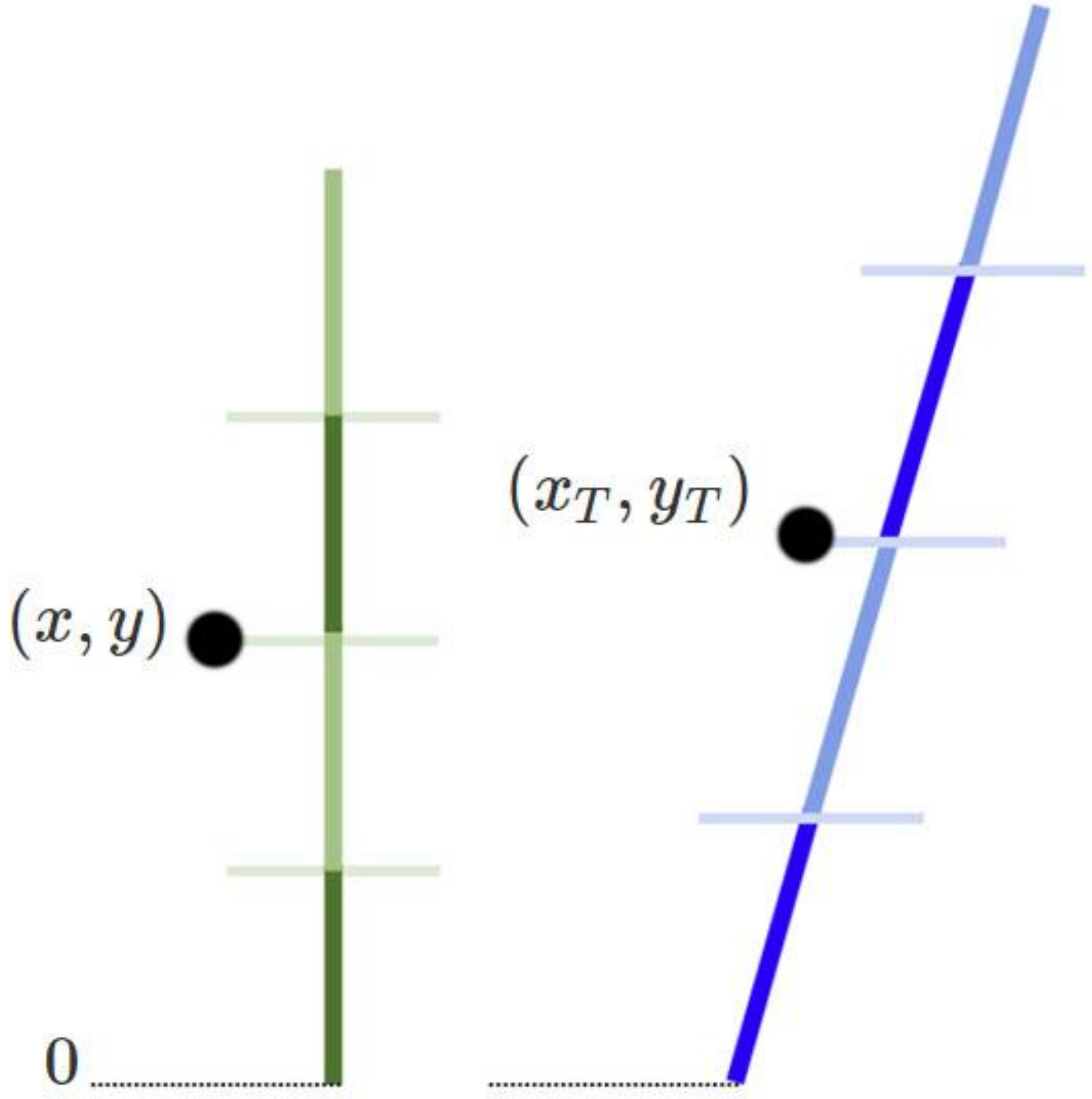
Credit: L-System tree rig / wind blow Submotion Pictures

Properties of Good Tree Bending

- Firmly planted (Right angle)
- Preserve volume
- Preserve distance along trunk/branches (uniform)
- Swaying leaves

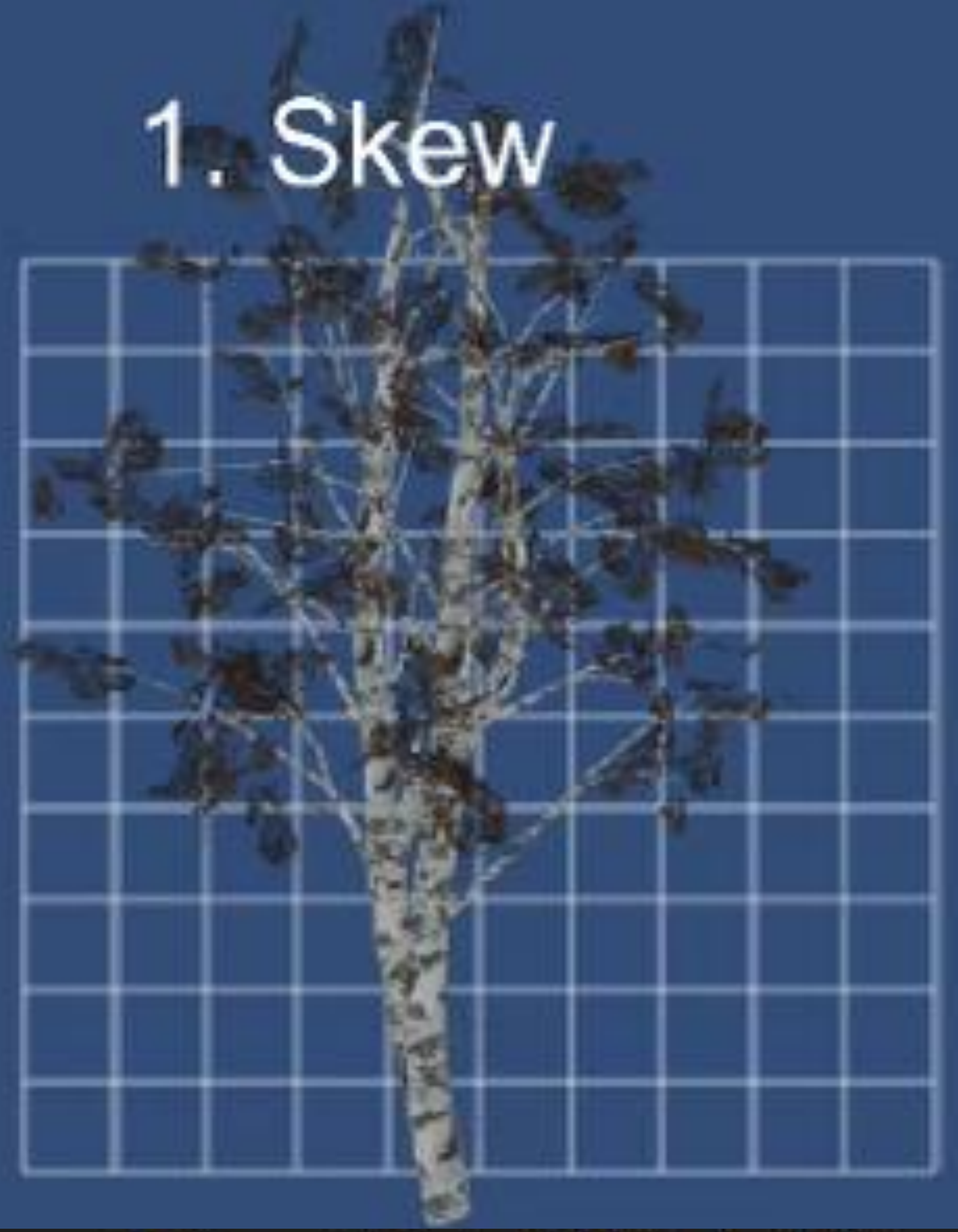


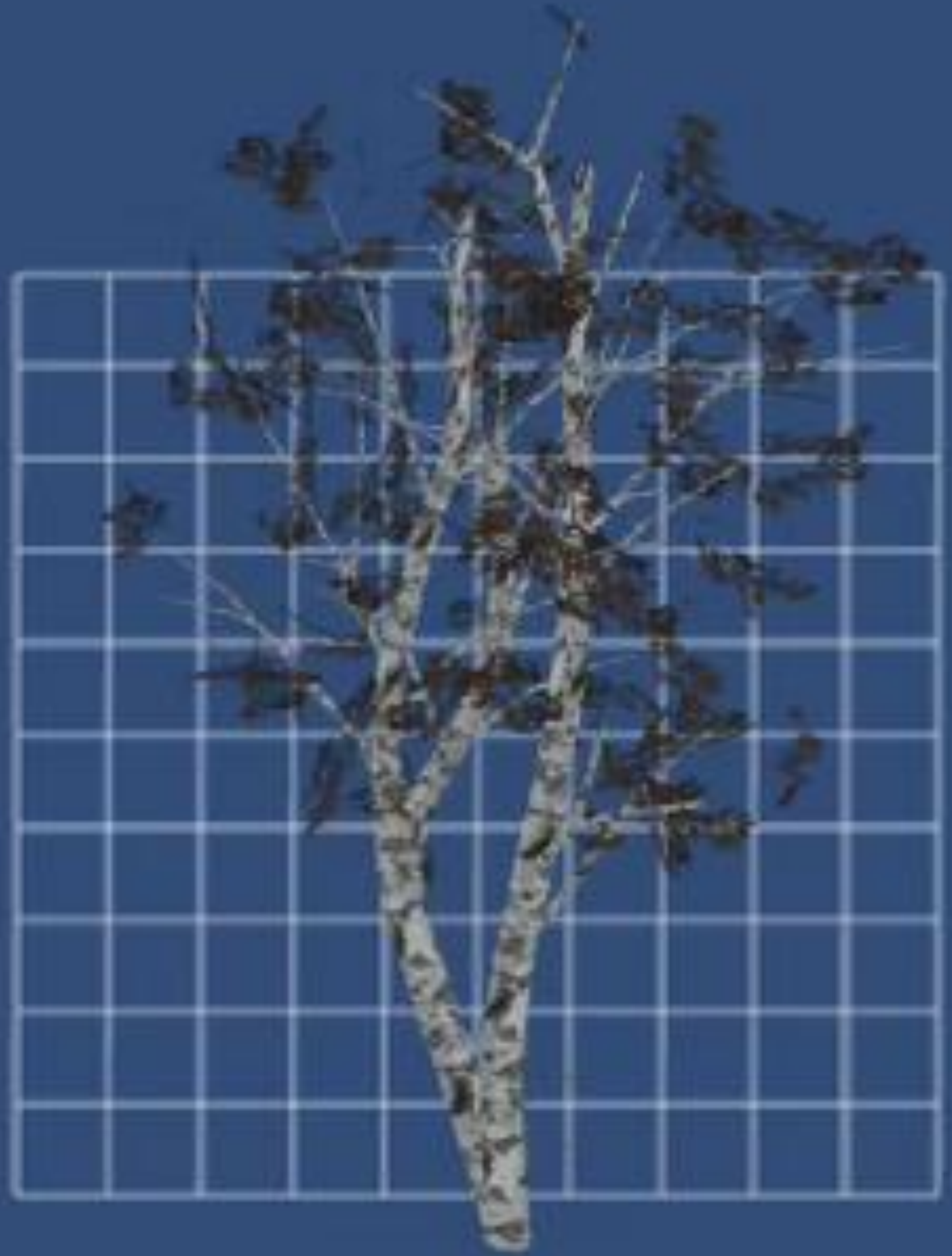
Skew



$$x_T = x + jy$$
$$y_T = y + ky$$

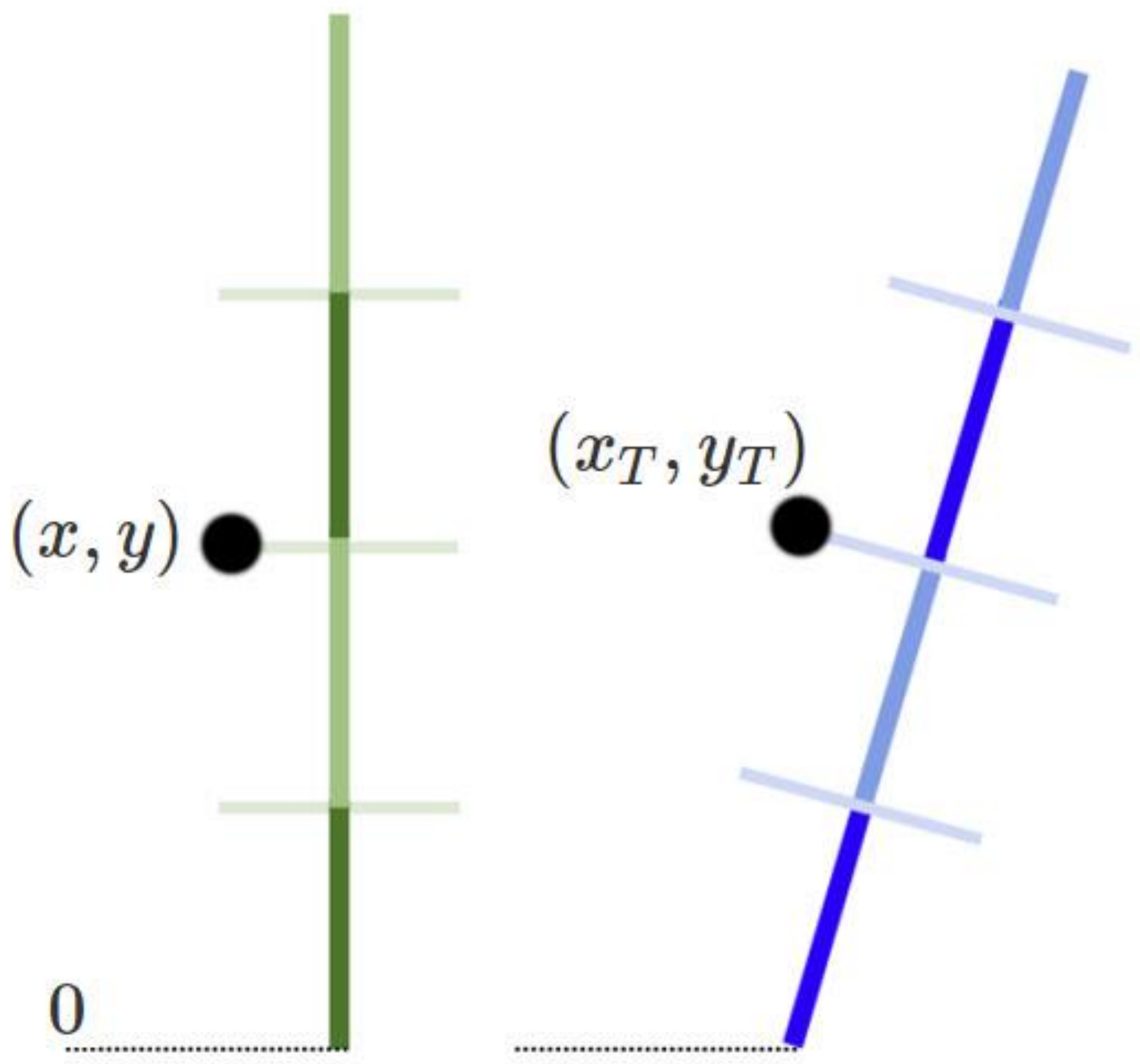
1. Skew





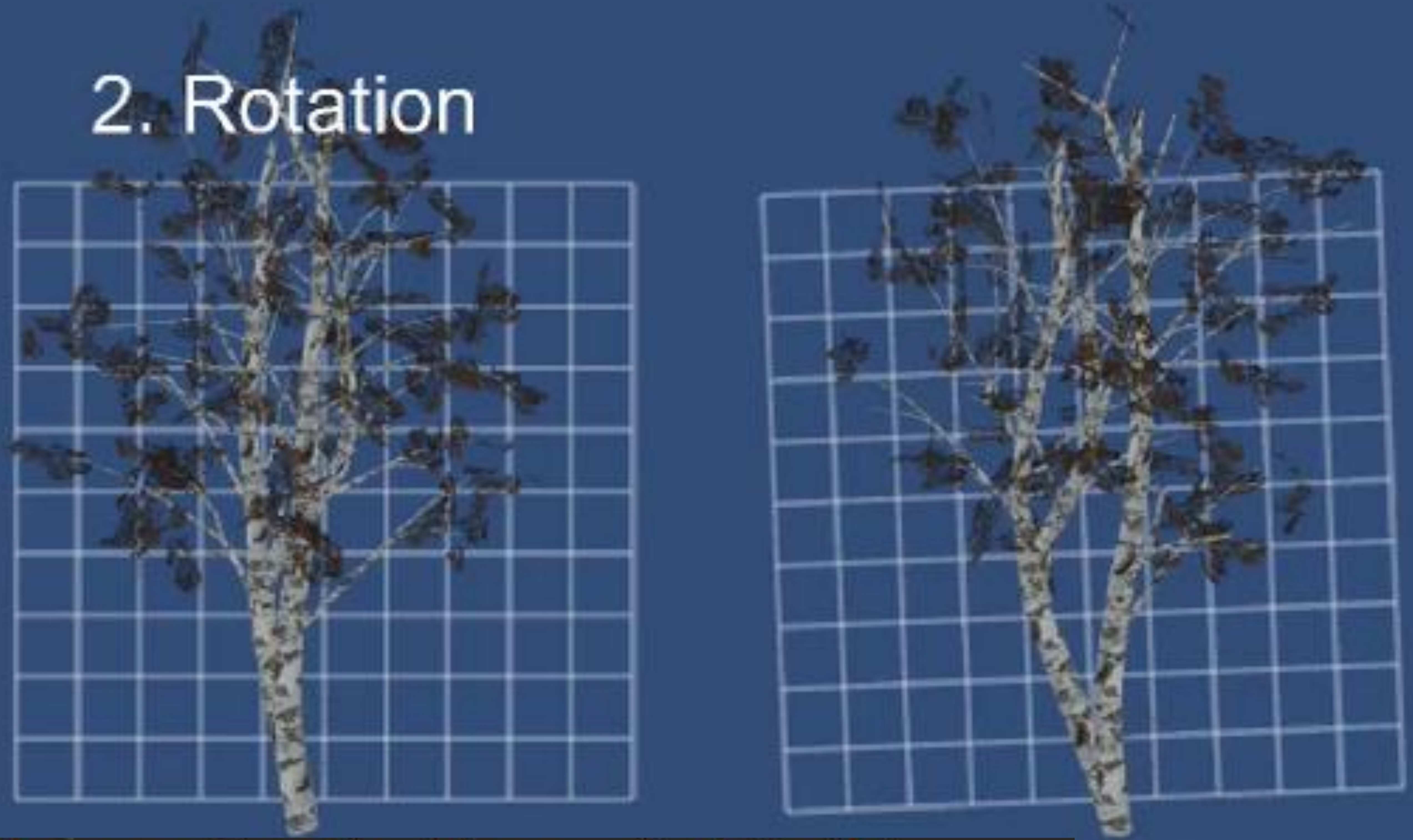
```
vt.x = v.x + xskew * v.y;
vt.y = v.y + yskew * v.y;
```

Rotation



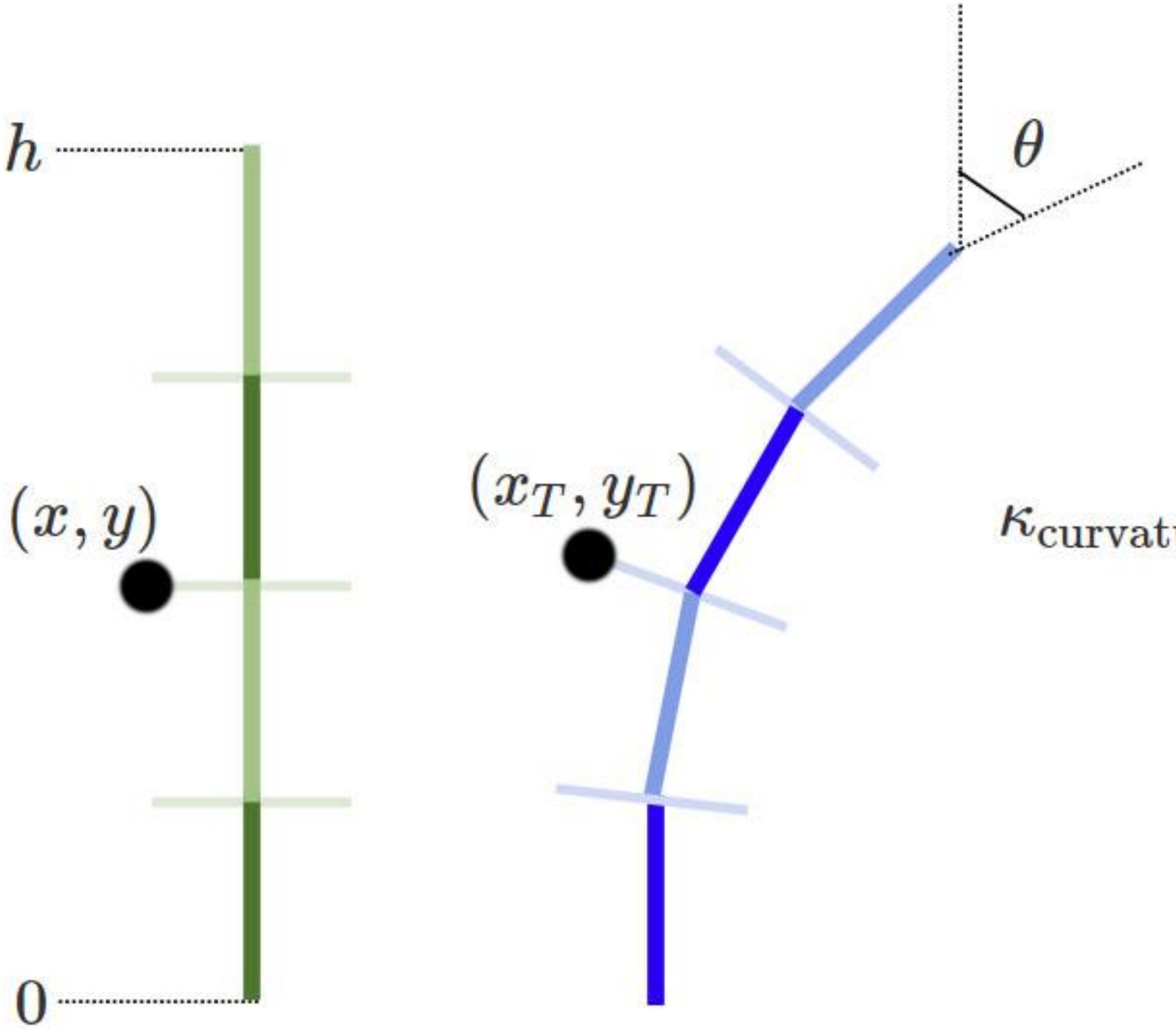
$$x_T = x\cos(\theta) + y\sin(\theta)$$
$$y_T = y\cos(\theta) - x\sin(\theta)$$

2. Rotation



```
vt.x = v.x * cos(angle) + v.y * sin(angle);
vt.y = v.y * cos(angle) - v.x * sin(angle);
```

Bend (Implicit)

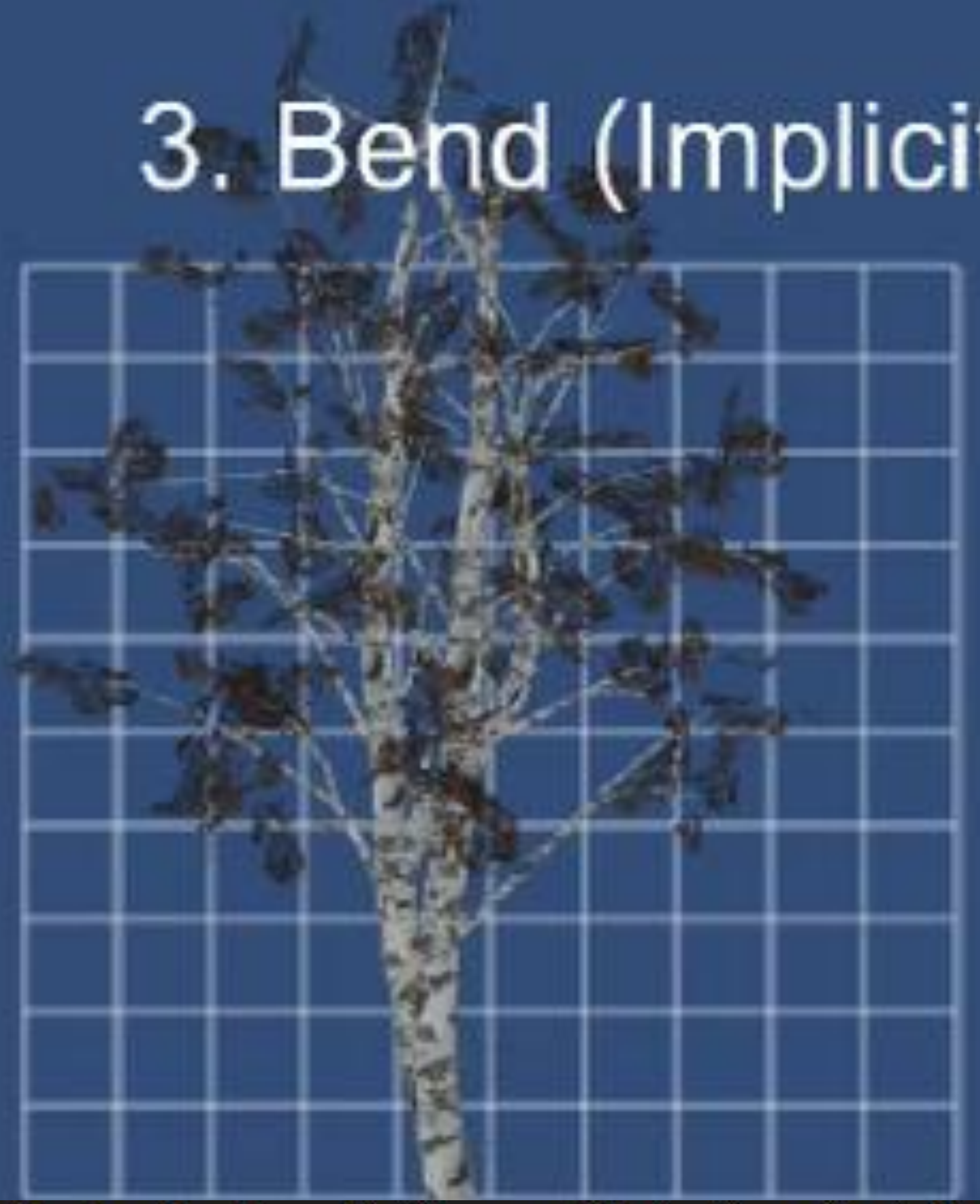


$$\kappa_{\text{curvature}} = \frac{\theta}{h}$$

$$x_T = \frac{1 - \cos(\kappa y)}{\kappa} + x\cos(\kappa y)$$

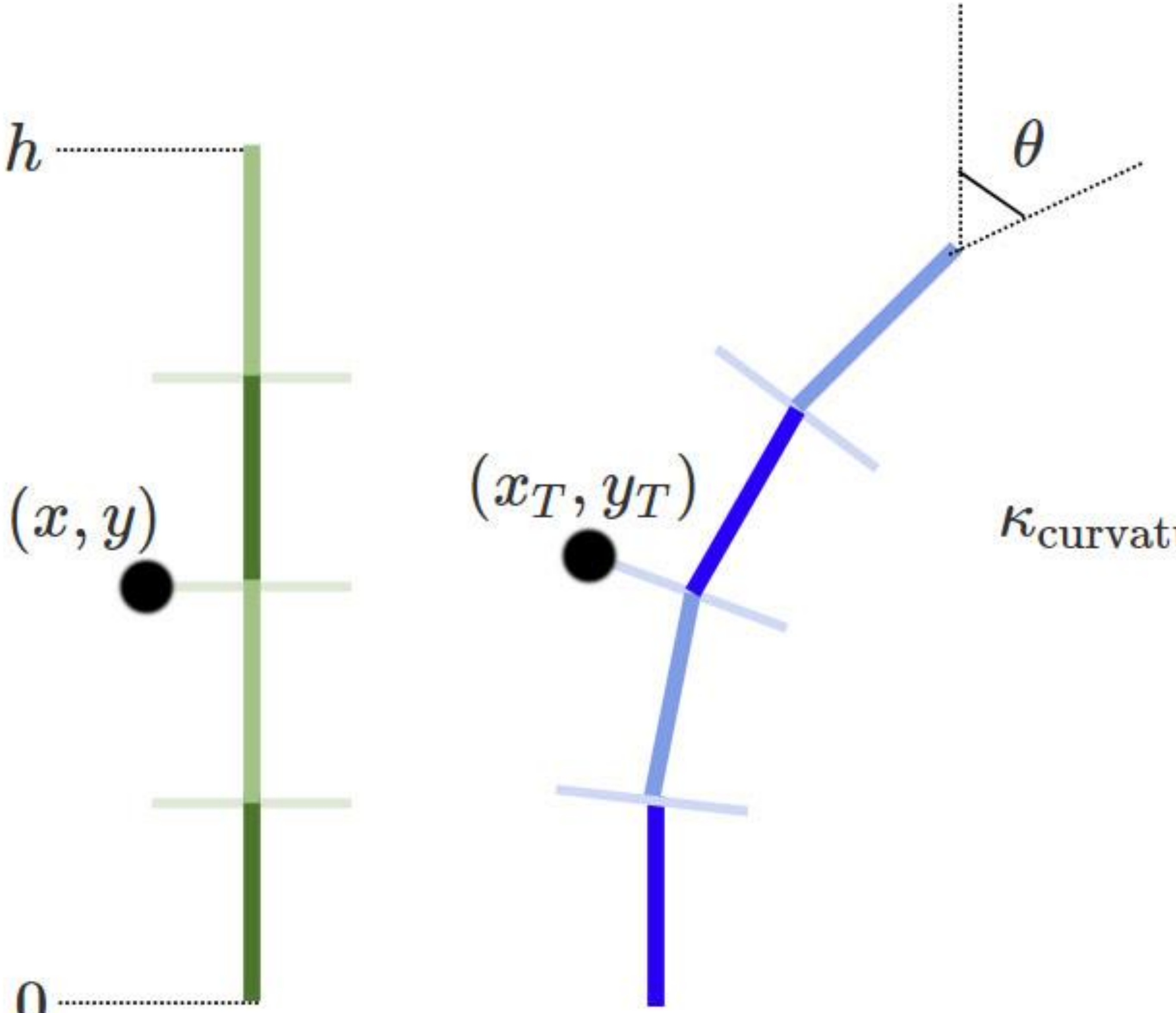
$$y_T = \frac{\sin(\kappa y)}{\kappa} - x\sin(\kappa y)$$

3. Bend (Implicit)



```
float x_bend = -(1.0f - cos(theta * v.y)) / theta;
float y_bend = sin(theta * v.y) / theta;

vt.x = x_bend + v.x * cos(theta * v.y);
vt.y = y_bend + v.x * sin(theta * v.y);
```

Bend (Explicit)



$$\kappa_{\text{curvature}} = \frac{\theta}{h}$$

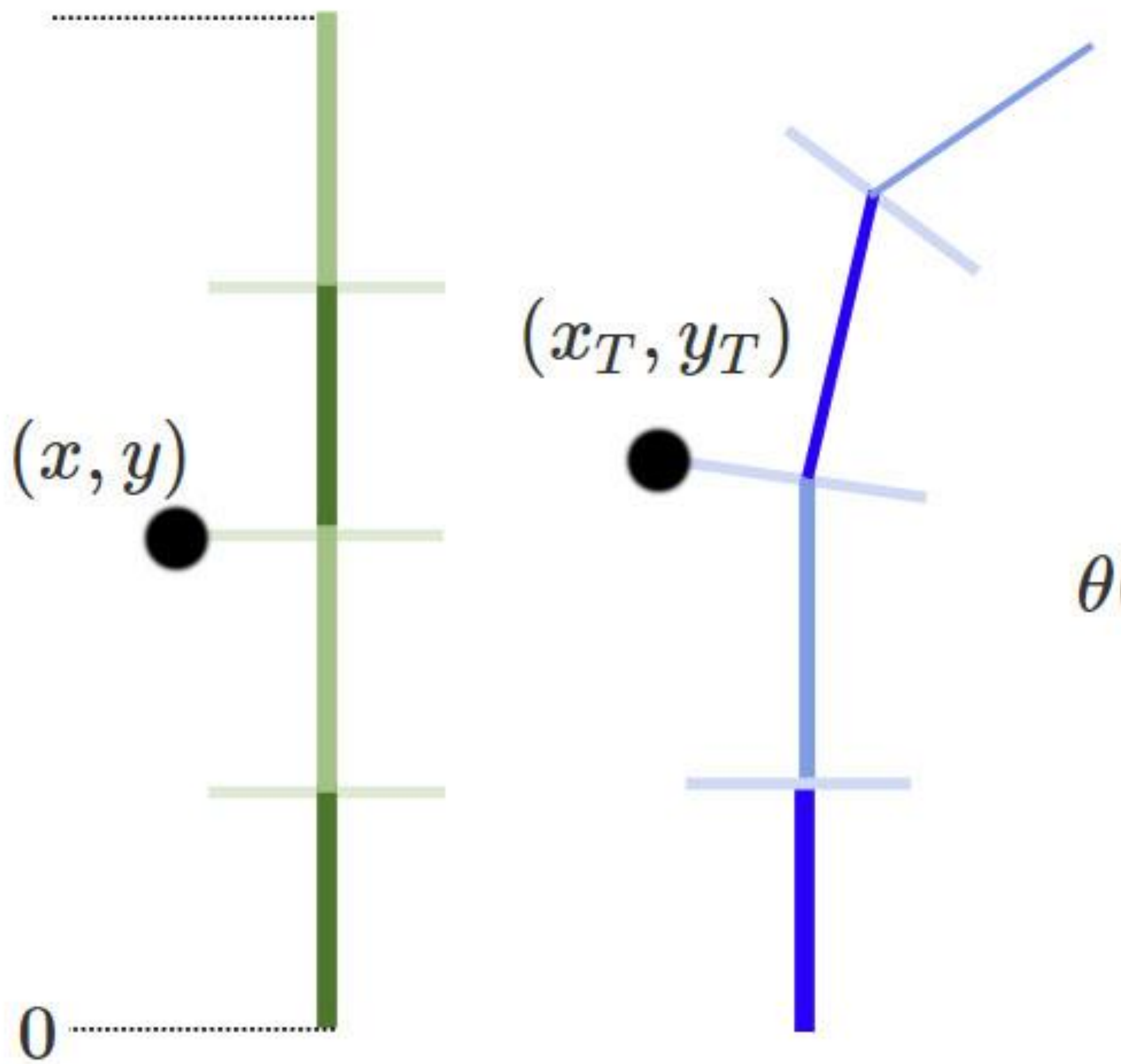
$$x_T = x\cos(\kappa y) + \frac{\kappa y^2}{2}$$

$$y_T = -x\sin(\kappa y) + y - \frac{\kappa^2 y^3}{6}$$

4. Bend (Taylor Series)

```
float x_bend = theta * v.y * v.y / -2;                      // (1-coskt) / t
float y_bend = v.y + theta * theta * v.y * v.y * v.y / -6;  // sin kt / t

vt.x = x_bend + v.x * cos(theta * v.y);
vt.y = y_bend + v.x * sin(theta * v.y);
```

Bend (General)

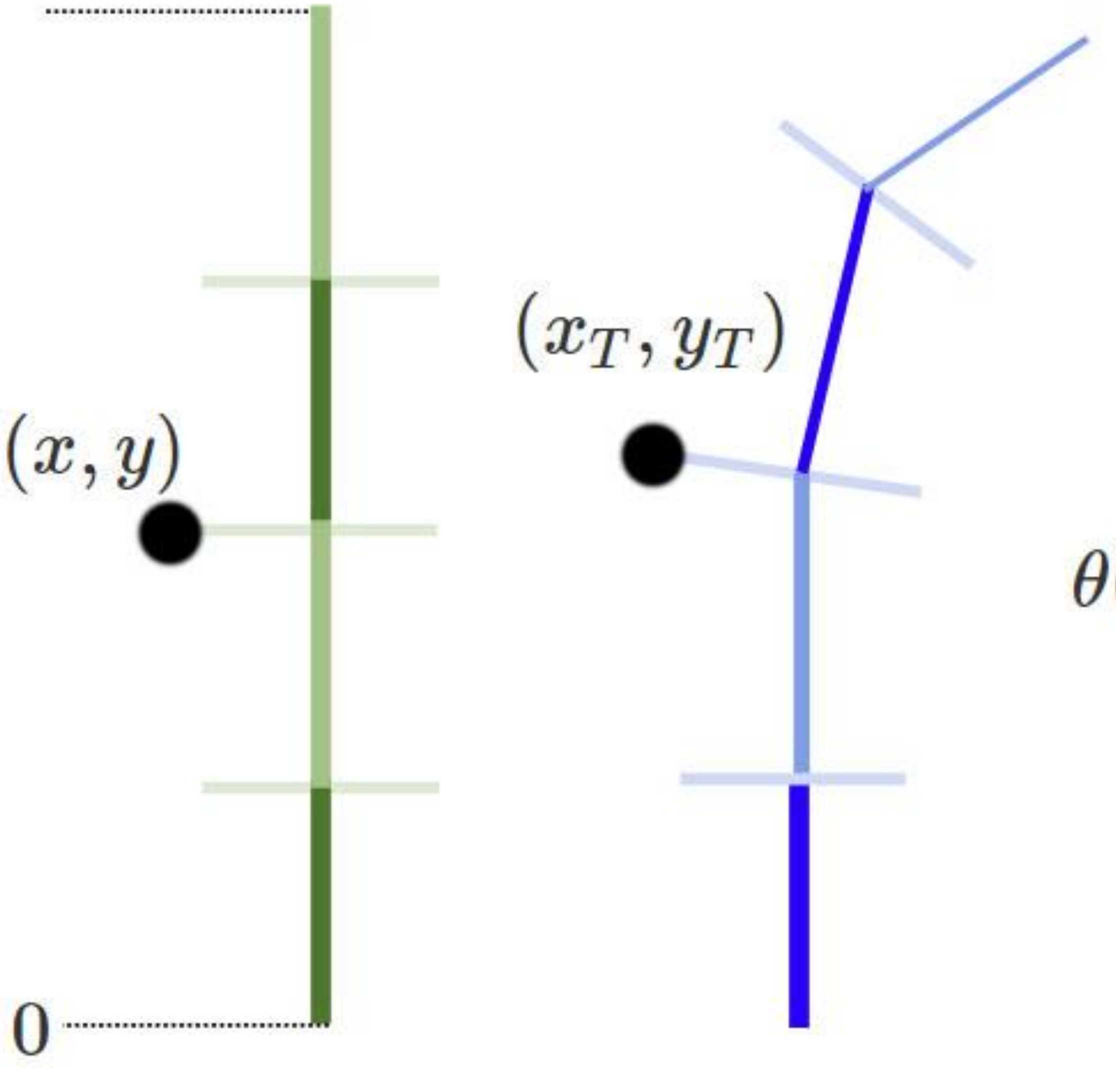


$\theta(y)$ = bend at height y

$$x_T = x\cos[\theta(y)] + \int_0^y \sin[\theta(h)]dh$$

$$y_T = -x\sin[\theta(y)] + \int_0^y \cos[\theta(h)]dh$$

Bend (General)

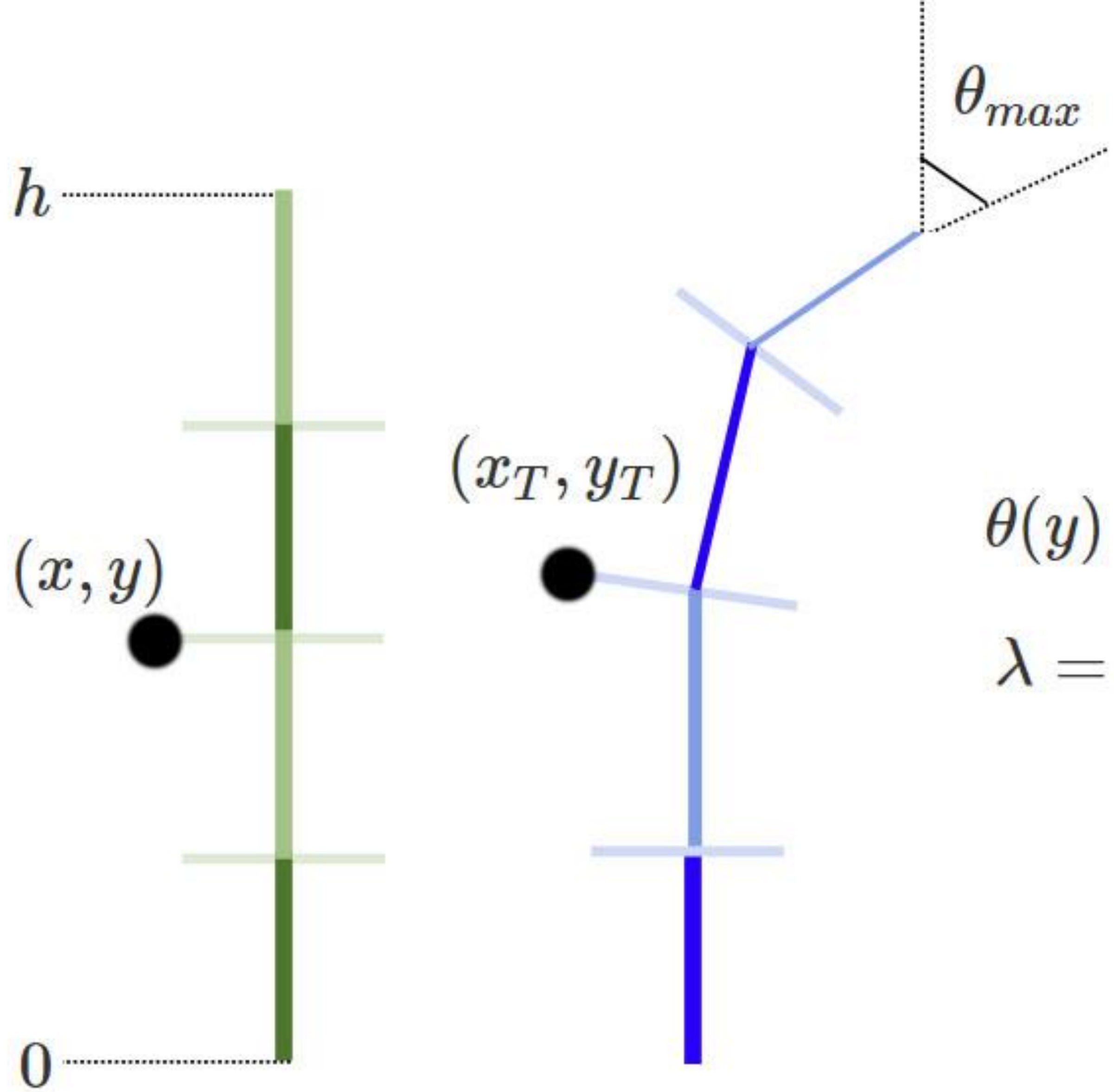


$\theta(y)$ = bend at height y

$$x_T = x\cos[\theta(y)] + \int_0^y \theta(h) - \frac{\theta(h)^3}{3!}dh$$

$$y_T = -x\sin[\theta(y)] + \int_0^y 1 - \frac{\theta(h)^2}{2!}dh$$

Bend (Gamma-Skewed)



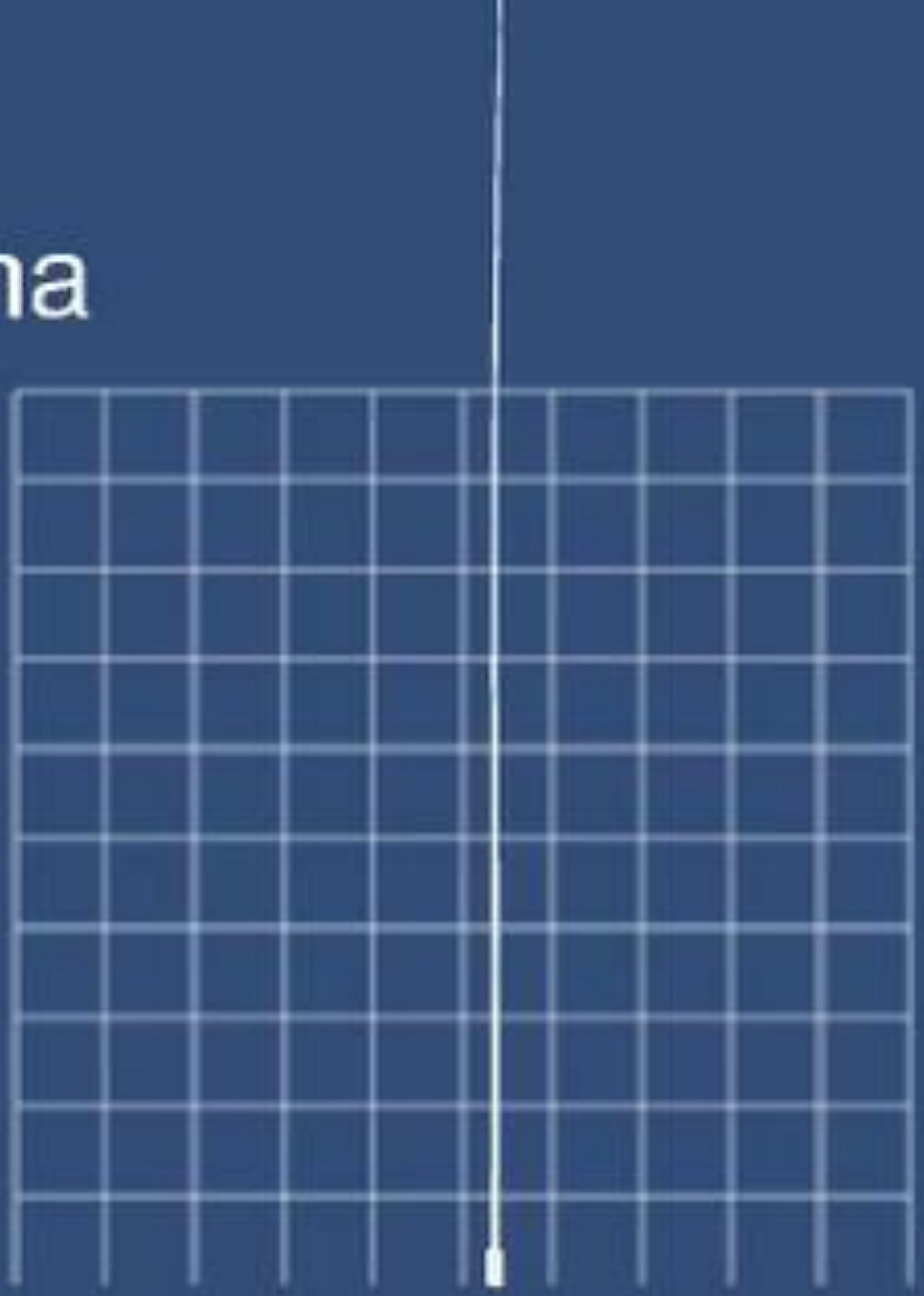
$$\theta(y) = \theta_{max} \cdot (\frac{y}{h})^\lambda$$

$\lambda = 3$ for fishing pole

$$x_T = x \cos[(\frac{y}{h})^\lambda \theta_{max}] + \frac{y^{1+\lambda}\theta_{max}}{h^\lambda(1+\lambda)}$$

$$y_T = -x \sin[(\frac{y}{h})^\lambda \theta_{max}] + y - \frac{y^{1+2\lambda}\theta_{max}^2}{2h^{2\lambda}(1+2\lambda)}$$

5. Bend with Gamma

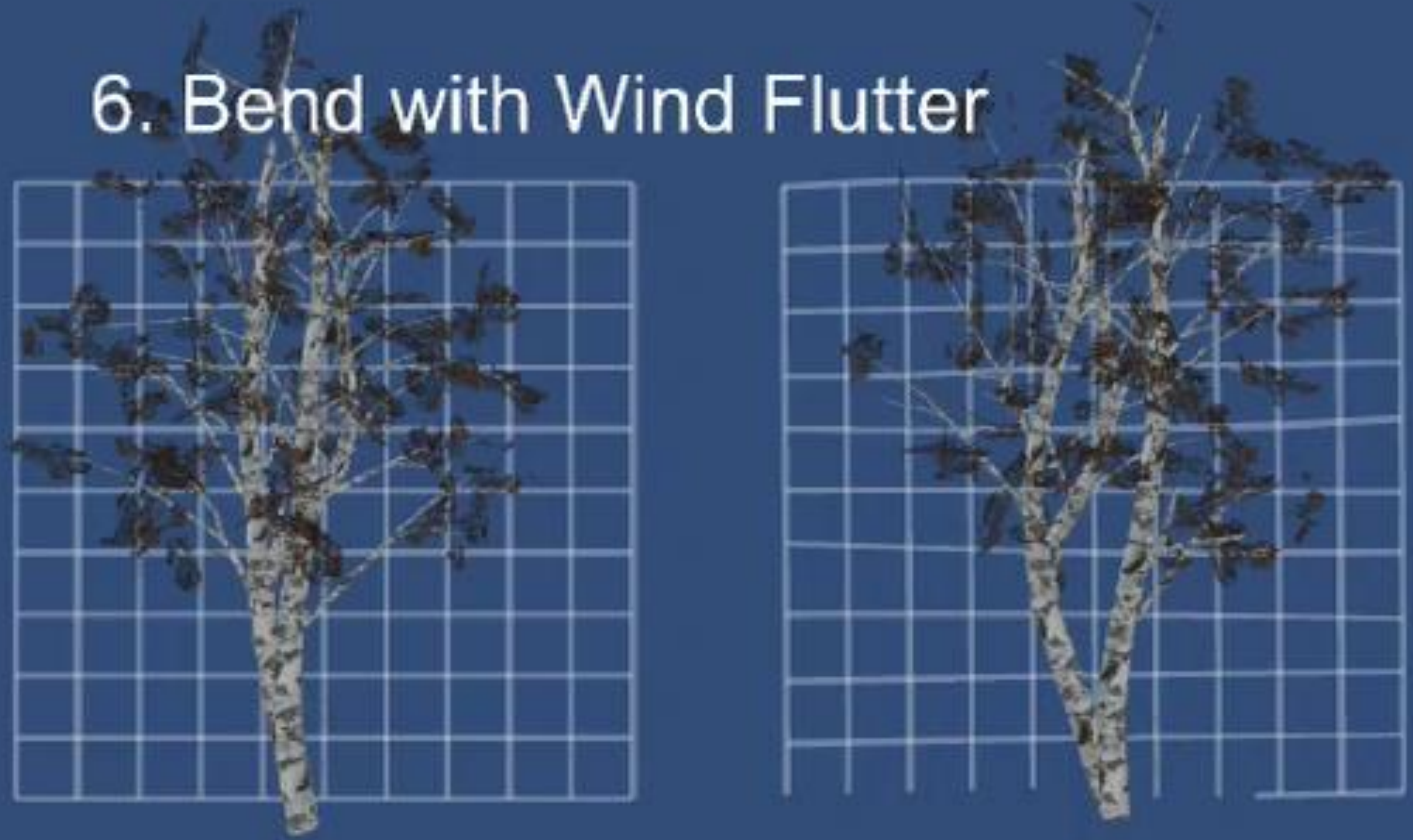


```
float x_bend = -theta * pow(v.y, 1.0f + k) / (1 + k);
float y_bend = v.y + theta * theta * pow(v.y, 1.0f + 2.0f * k) / (-2 * (1 + 2.0f * k));

vp.x = x_bend + v.x * cos(theta * pow(v.y, k));
vp.y = y_bend + v.x * sin(theta * pow(v.y, k));
```

```
float windTime = _Time.w * 0.25f;
// some octaves of noise
float windTheta = 0.125 * cos(windTime / 0.5 * 6.28f) + 0.25 * cos(windTime / 0.73 * 6.28f) +
    0.5 * cos(windTime / 1.28 * 6.28f) + 1.0 * cos(windTime / 3.9 * 6.28f);

// add some continuous variation based on x and y across the tree
float leafTime = _Time.w * 0.5f + v.y * 3.0f + v.x * v.y * 0.25f;

// some octaves of noise
float leafFlutter = 0.125 * cos(leafTime / 0.5 * 6.28f) + 0.25 * cos(leafTime / 0.73 * 6.28f) +
    0.5 * cos(leafTime / 1.28 * 6.28f) + 1.0 * cos(leafTime / 3.9 * 6.28f);
```

```
theta += 0.01 * windTheta; // add to the bend angle

v.y += isLeaf * v.x * leafFlutter * 0.02;

float k = 0.5f; // K = ~0.5 for a tree, 3.0 for a fishing rod

float yoffset = v.y + theta * theta * pow(v.y, 1.0f + 2.0f * k) / (-2 * (1 + 2.0f * k)); // sin kt / t
float xoffset = -theta * pow(v.y, 1.0f + k) / (1 + k);   // (1-coskt) / t

vt.x = xoffset + v.x * cos(theta * pow(v.y, k));
vt.y = yoffset + v.x * sin(theta * pow(v.y, k));
```

6. Bend with Wind Flutter



Part 2: Other Effects

- World streaming
- Soft body
- Exploding meshes
- Constructing buildings

7. Warp into place

```
float shift = min(-3.0 + 0.4 * t - v.x, 0.0f);
vt.x = v.x;
vt.y = v.y - shift * shift;
```

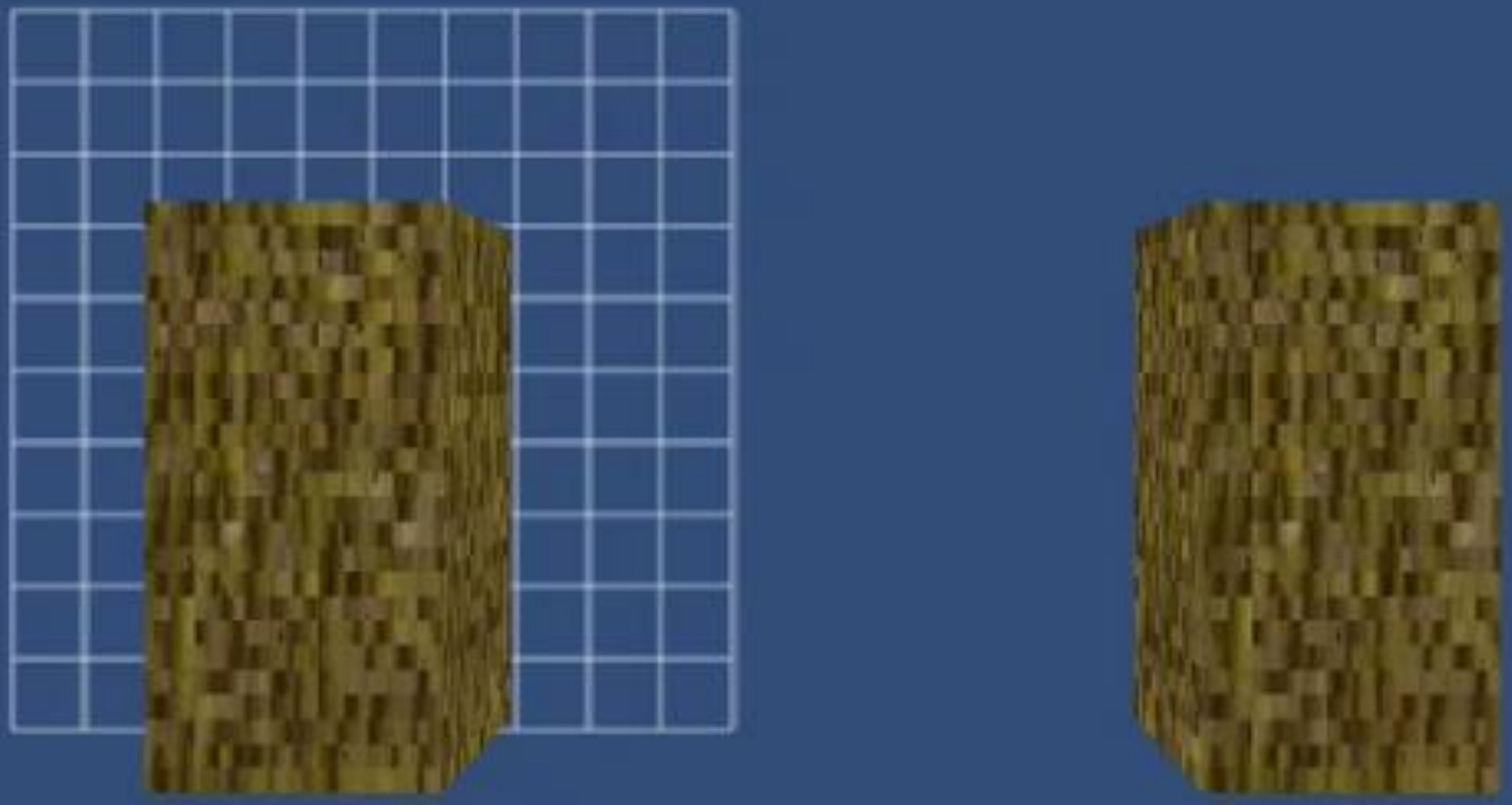
8. Soft body

```
float k = parameter.y + 1.0f;
float a = parameter.x * 3.0f;
float b = (1.0f - k - a * k * k * k / 3) * 2.0f / (k * k);
float y = vp.y * 0.5f;

float xs = a * y * y + b * y + 1;

vt.x = v.x * xs;
vt.y = v.y * k;
```

9. Construct

```
float t = fmod(_Time.w, 20) / 2.0f;
float tstart = uv.y + uv.x * 0.5 + v.y;  // adjust for aesthetics
float tend = uv.y + uv.x + 2.0f * v.y;   // adjust for aesthetics
float assembleT = saturate((t - tstart) / (tend - tstart));

// bend out in direction of normal, towards ground
vt = v;
vt += norm * vp.y * (cos(assembleT * 1.57075f))
vt.y *= sin(assembleT * 1.57075f)
```

10. Explode



```
float t = fmod(_Time.w, 30) / 3.0f;
float tstart = 3.0f + (length(vp * float3(1,0.5,1)) + uv.x + uv.y ) * 0.05 + vp.y * 0.2f;

// calculate some coherent but broken up variation
float forceVariation = cos(uv.x * 10) + sin(uv.y * 10);
float3 explodeDir = normalize(vp + float3(0,1,2));

// add some variation to the force just to break it up
float explodeForce = 10.0f * (forceVariation * 0.25f + 1.0f);

float g = 6.0f;

// quadratic solve for groundT so that things land on the ground
float yForce = explodeForce * explodeDir.y;
float groundT = (yForce + sqrt(yForce*yForce + 4 * g * vp.y)) / (2*g);

// clamp T to between these values
t = clamp(t - tstart, 0, groundT);

float gravity = g * t * t;
vt = v;
vt.y -= gravity;
vt += (explodeForce * t) * explodeDir;
```

Questions?

Thanks for attending!

michael@hiddenpath.com

@QuanticMage