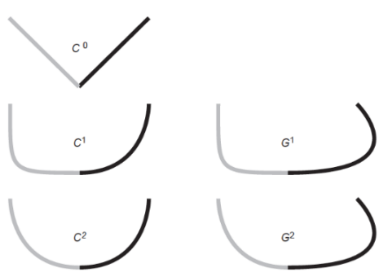
Transformation Stack → Transformations are maintained by a **stack**:
`ctx.save()`: copies the top element (transformation) and pushes on the stack
`ctx.restore()`: pops the top stack (goes the the latest ctx.save())
`ctx.moveTo(x,y)`: moves the pen (stays there even if we translates or apply transformations)
`ctx.lineTo(x,y)`: creates a line to point

Homogeneous matrix → +1 dimension to the vector (3-d matrix for 2-d vectors)
Changes multiplication + addition to only multiplication

$$\begin{bmatrix} acos(\theta) & -sin(\theta) & c \\ sin(\theta) & bcos(\theta) & d \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

This is equal to `ctx.translate(c,d)` * `ctx.scale(a,b)` * `ctx.rotate(theta)`
GLmatrix `mat3.multiply(out, a, b)` sets the value of out to `a*b`
Point `(x,y,c)` is interpreted as `(x/c, y/c)` in R2



Curves and Continuity (right implies left)
C_0 > G_1 > C_1 > G_2 > C_2 >

C_0: two curves meet at a common point
G_1: two curves meet at a common point, and also has same tangent direction
C_1: two curves meet at a common point, and also has same tangent direction and magnitude
G_2: two curves meet at a common point, and also has same tangent direction and magnitude(double derivative direction only)
C_2: two curves meet at a common point, and also has same tangent direction and magnitude (double derivative as well)

$$\mathbf{f}(u) = \begin{bmatrix} x(u) & y(u) \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & u & u^2 \cdots & u^N \end{bmatrix}}_{\mathbf{u}} \underbrace{\begin{bmatrix} a_0 & b_0 \\ a_1 & b_1 \\ a_2 & b_2 \\ \vdots & \vdots \\ a_N & b_N \end{bmatrix}}_{\mathbf{A}}$$

Arc-length parametrization: The derivative of the parametric curve has a magnitude of 1 → |C'(t)| = 1

Polynomial Curves → f(u) = uA = uBP, b(u) = basis
P = [f(1), f(2), ..., f(n+1)] and C = [u(1), u(2), ..., u(n+1)], then P=CA so A =inv(C)P and B=inv(C)

$$\mathbf{b}(u) = \begin{bmatrix} 1 & u & u^2 & u^3 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ -3 & -2 & 3 & -1 \\ 2 & 1 & -2 & 1 \end{bmatrix}$$
$$= \begin{bmatrix} b_0(u) & b_1(u) & b_2(u) & b_3(u) \end{bmatrix}$$

Hermites – fix start/end points & tangents
P = [start p, start tangent, end p, end tangent]
Has C1 but not C2, good local control

B-Splines Curves: Doesn't necessary go through the control points (can but will lose C_2)
Bezier Curves: Special case of hermite curve. Special property → curve stays in convex hull
Natural Cubic: Give C(0), C'(0), C''(0), C(1) → very easy to enforce C_2 but loses local control

- For polynomial curves, *three* of the following properties can be satisfied simultaneously (not all 4!)
  - C2 continuity of the curve — Counter-example: Hermite
  - Interpolation of all "control points" — Counter-example: B-splines
  - Local control of curve — Counter-example: Natural cubics
  - The polynomials have order no more than 3

Lookat Transform: Transforms from world coordinates to camera coordinates

```
(static) lookAt(out, eye, center, up) → {mat4}
```

Generates a look-at matrix with the given eye position, focal point, and up axis.

Parameters:

Name	Type	Description
out	mat4	mat4 frustum matrix will be written into
eye	ReadonlyVec3	Position of the viewer
center	ReadonlyVec3	Point the viewer is looking at
up	ReadonlyVec3	vec3 pointing up

Center in world coords will lie along the negative part of the w-axis (0,0,-w)
Target in world coords will lie on origin
Up vector is a vector (in world coordinates), that when viewed from camera will show as vertical. Up vector is not the same as the v-axis of the camera system! (same plane, but doesn't have to be the same)

John pork wishes you the best :)