

Parametric Surfaces & Tessellation

EDAF80

Michael Doggett



Slides by Jacob Munkberg 2012-13

Today

- Parametric Surfaces
- Tessellation
- Interpolation
- Animation

Parametric Surfaces

Creating objects

The xy -plane

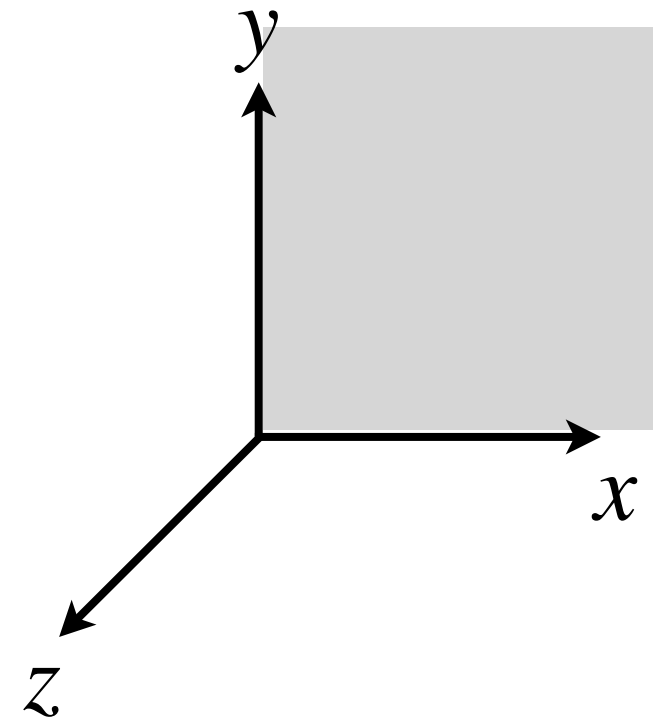
- Implicit form
 - Defined by normal $\mathbf{n} = (0,0,1)$ and point on plane $P_o = (0,0,0)$
 - Point P on plane if

$$(P - P_o) \cdot \mathbf{n} = 0$$

$$P \cdot (0, 0, 1) = 0$$

$$P_z = 0$$

i.e., all points of the form $P = (x, y, 0)$



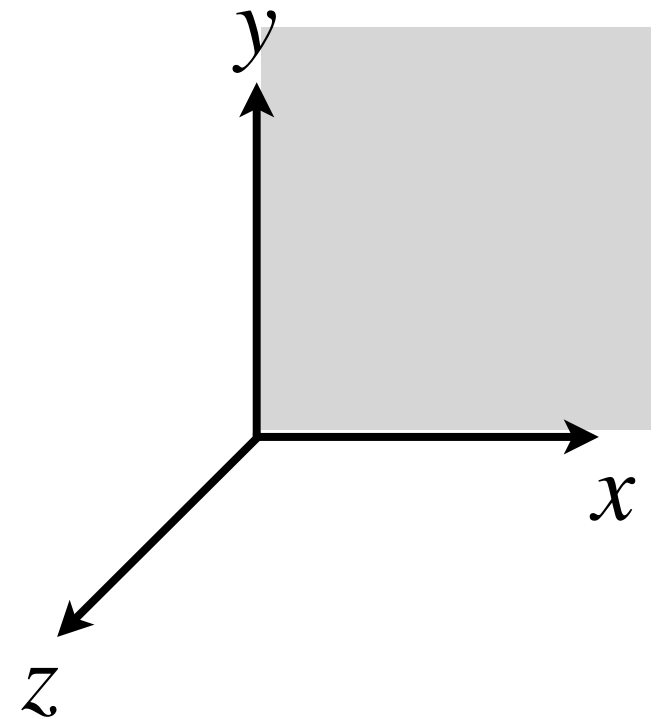
The xy -plane

- Explicit form
 - Map from \mathbf{R}^2 to \mathbf{R}^3

$$\mathbf{p}(u, v) : \mathbf{R}^2 \rightarrow \mathbf{R}^3$$

$$\mathbf{p}(u, v) = (u, v, 0)$$

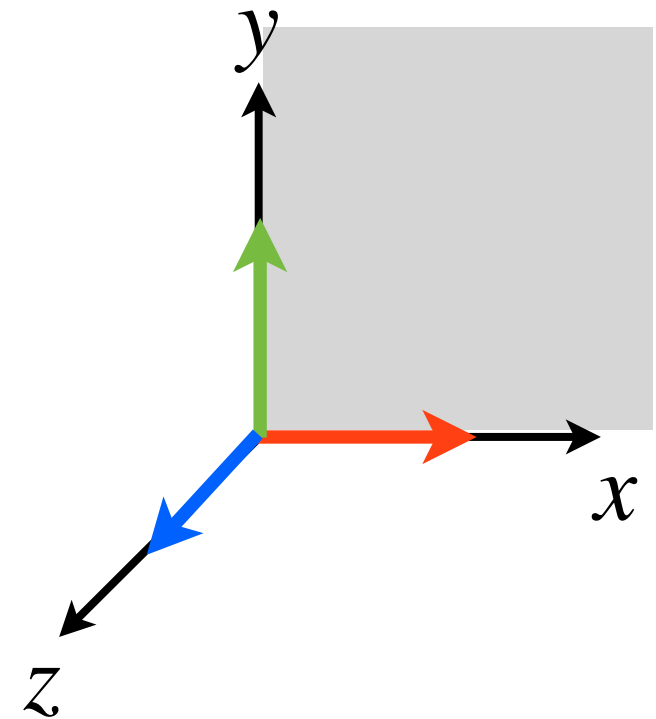
$$\begin{bmatrix} x(u, v) \\ y(u, v) \\ z(u, v) \end{bmatrix} = \begin{bmatrix} u \\ v \\ 0 \end{bmatrix}$$



The xy -plane

$$\mathbf{p}(u, v) = (u, v, 0)$$

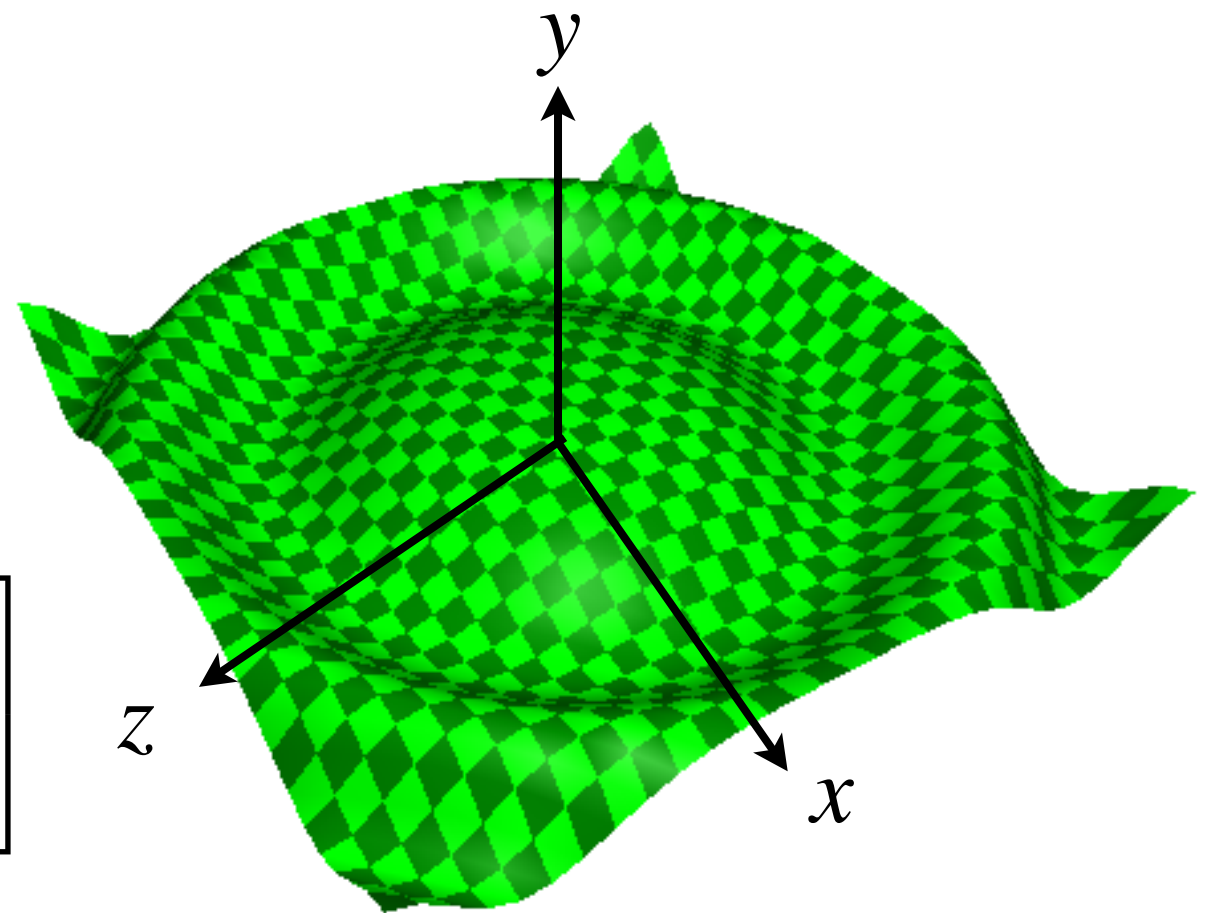
- **Tangent:** $\mathbf{t} = \frac{\partial \mathbf{p}}{\partial u} = (1, 0, 0)$
- **Binormal:** $\mathbf{b} = \frac{\partial \mathbf{p}}{\partial v} = (0, 1, 0)$
- **Normal direction** $\mathbf{n} = \frac{\partial \mathbf{p}}{\partial u} \times \frac{\partial \mathbf{p}}{\partial v} = (0, 0, 1)$



Height field - Zoneplate

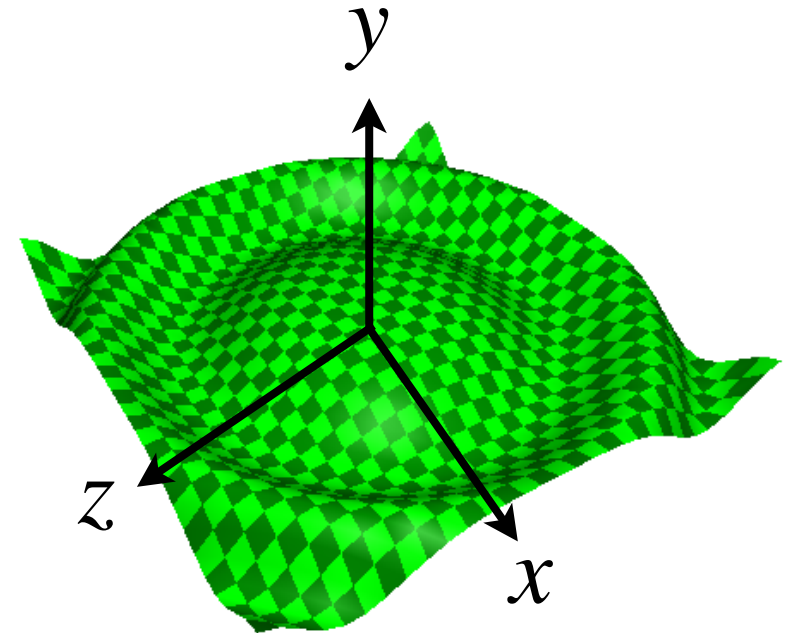
- Explicit form
 - Map from \mathbb{R}^2 to \mathbb{R}^3
 - Defines a height value for each (x,z) point

$$\begin{bmatrix} x(u, v) \\ y(u, v) \\ z(u, v) \end{bmatrix} = \begin{bmatrix} u \\ A(1 + \cos(2\pi(u^2 + v^2))) \\ v \end{bmatrix}$$



Height field - Zoneplate

$$\begin{bmatrix} x(u, v) \\ y(u, v) \\ z(u, v) \end{bmatrix} = \begin{bmatrix} u \\ A(1 + \cos(2\pi(u^2 + v^2))) \\ v \end{bmatrix}$$



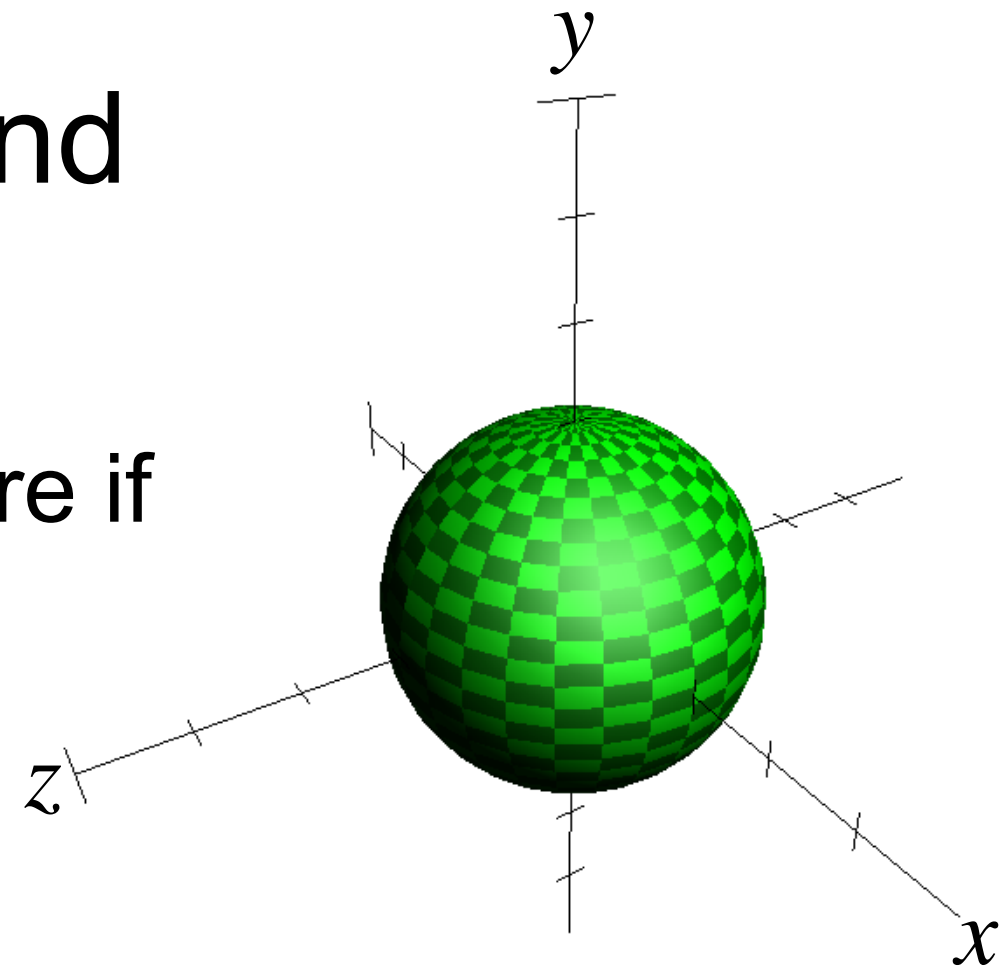
$$\frac{\partial \mathbf{p}}{\partial u} = (1, -4A\pi u \sin(2\pi(u^2 + v^2)), 0)$$

$$\frac{\partial \mathbf{p}}{\partial v} = (0, -4A\pi v \sin(2\pi(u^2 + v^2)), 1)$$

$$\mathbf{n} = \frac{\partial \mathbf{p}}{\partial u} \times \frac{\partial \mathbf{p}}{\partial v} = \begin{bmatrix} -4A\pi u \sin(2\pi(u^2 + v^2)) \\ 1 \\ -A4\pi v \sin(2\pi(u^2 + v^2)) \end{bmatrix}$$

Sphere

- Defined by a radius, r and origin $\mathbf{o} = (o_x, o_y, o_z)$
 - Implicit form: Point P on sphere if
 - Example: Unit sphere ($r=1$), centered at origin ($\mathbf{o}=(0,0,0)$).



Let $P = (x,y,z)$, then: $|P - \mathbf{o}| = r$

$$|P| = 1$$

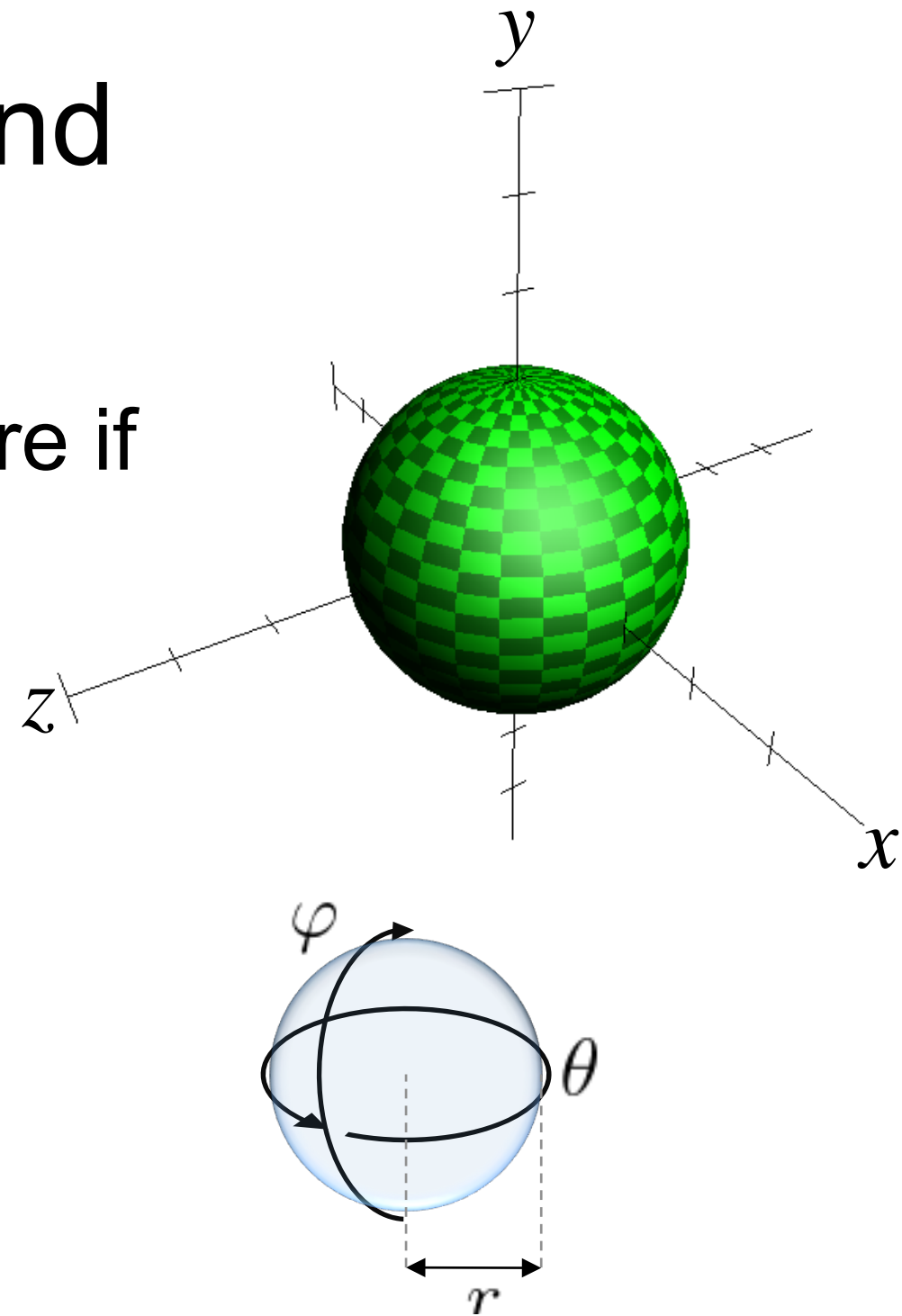
$$\sqrt{x^2 + y^2 + z^2} = 1$$

$$x^2 + y^2 + z^2 = 1$$

Sphere

- Defined by a radius, r and origin $\mathbf{o} = (o_x, o_y, o_z)$
 - Implicit form: Point P on sphere if
$$|P - \mathbf{o}| = r$$
 - Parametric definition:

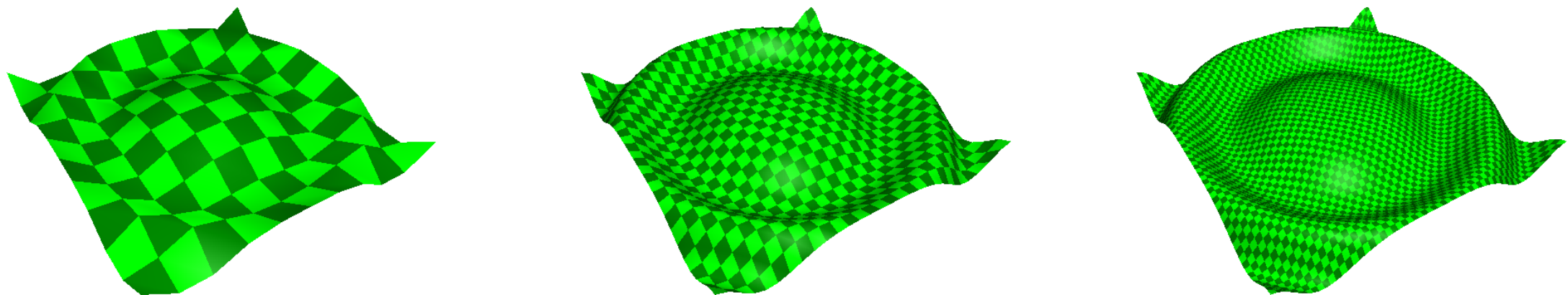
$$s(\theta, \varphi) = \begin{bmatrix} r \sin \theta \sin \varphi \\ -r \cos \varphi \\ r \cos \theta \sin \varphi \end{bmatrix}$$



Tessellation

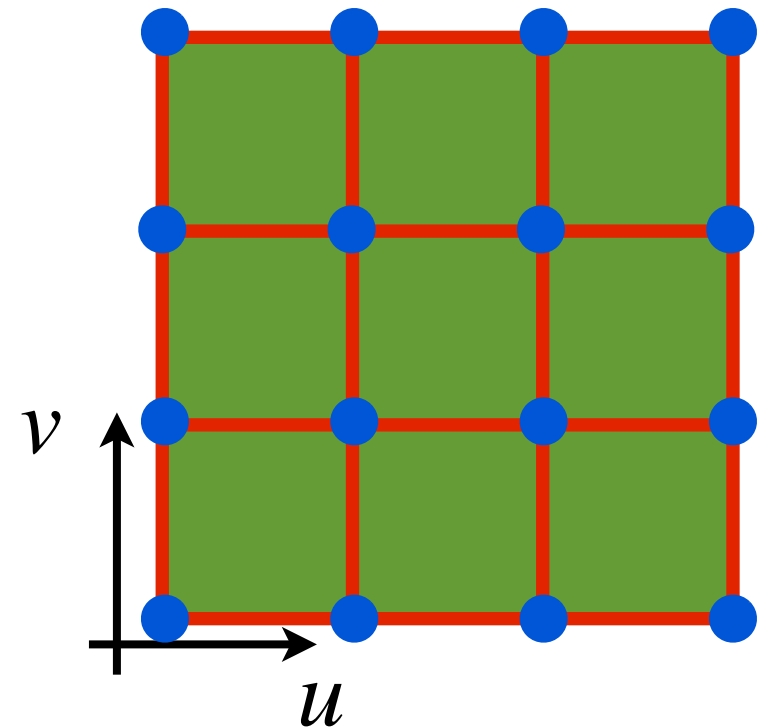
- Construction of polygon meshes
 - From parametric representations
 - Example: Tessellating the zoneplate.
Evaluate parametric surface at many positions

$$(u, v) = \left(\frac{i}{m}, \frac{j}{n}\right), \quad i = 0 \dots m, \quad j = 0 \dots n$$



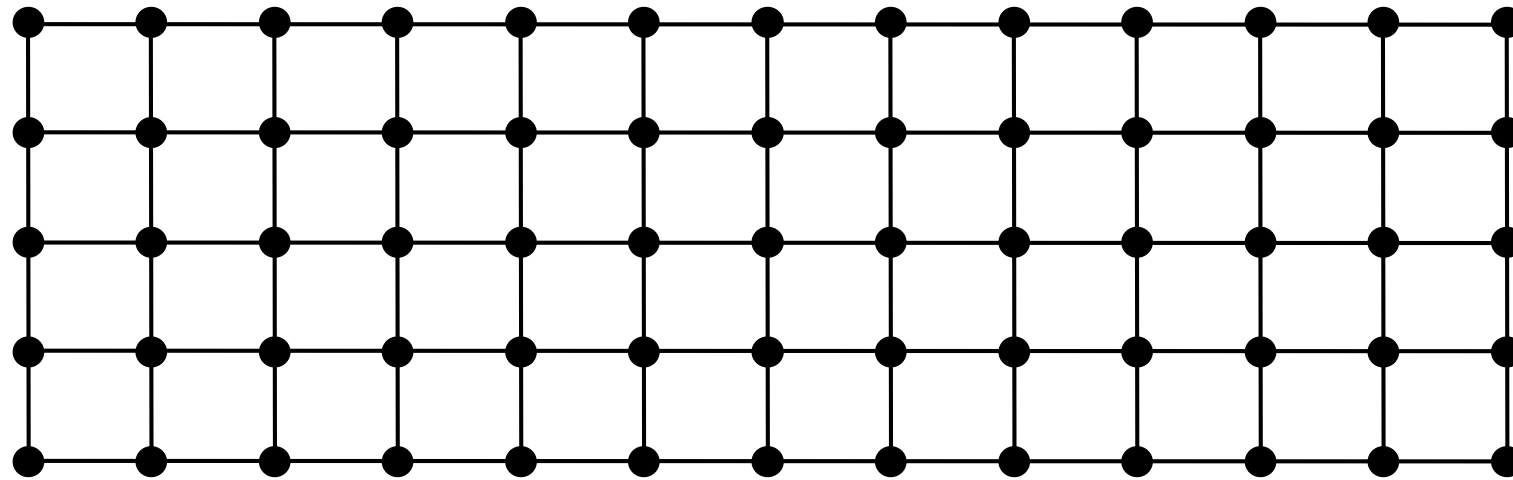
Mesh Topology

- Topology of the mesh
 - **Vertices**, **faces** and **edges** defined by a grid
- Grid parameterization
 - Location of grid vertices
- Position of vertices
 - Evaluate parametric surface function at each parametric location of the grid

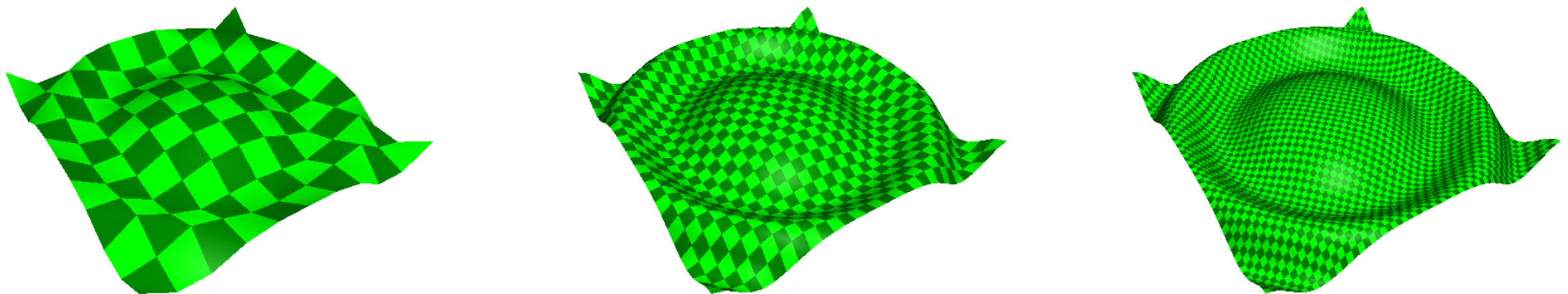


Grid Topologies

- The simplest topology is a quad

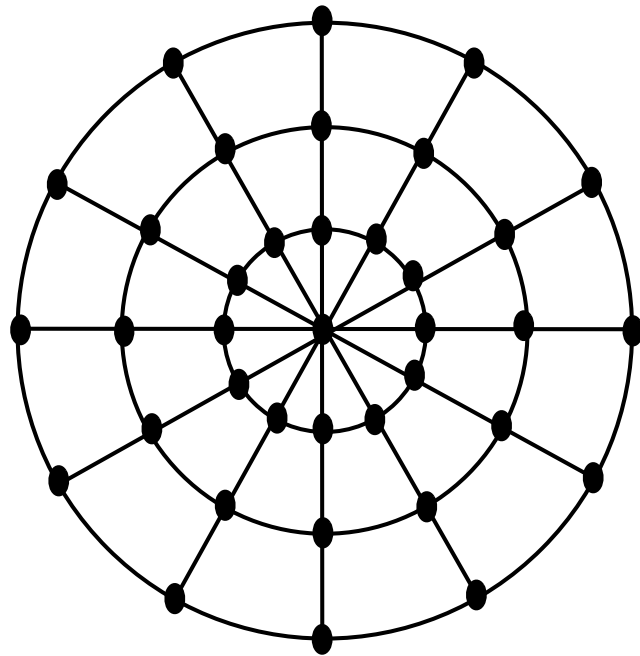


$$(u, v) = \left(\frac{i}{m}, \frac{j}{n}\right), \quad i = 0 \dots m, \quad j = 0 \dots n$$

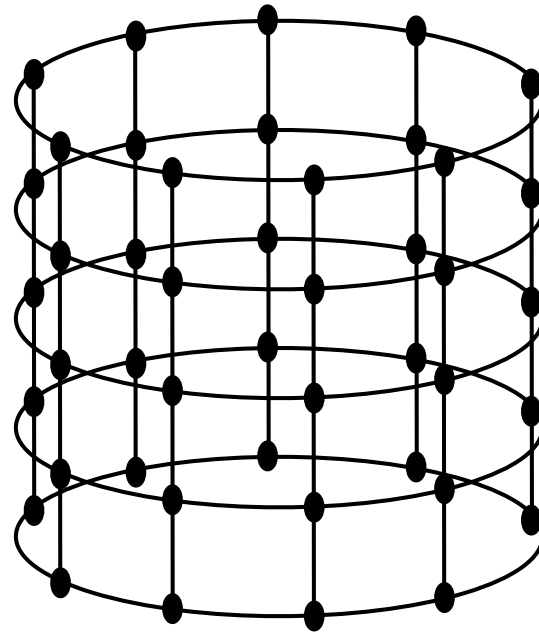


Increasing m & n

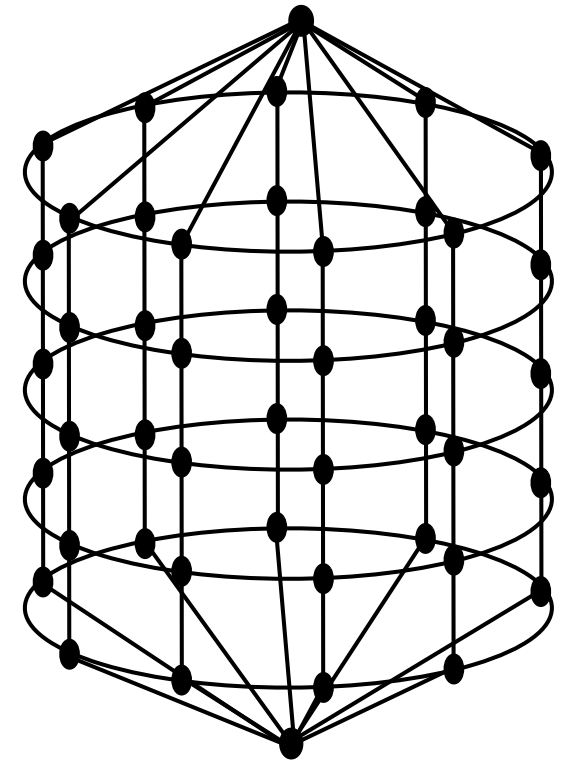
Other grid topologies



Web

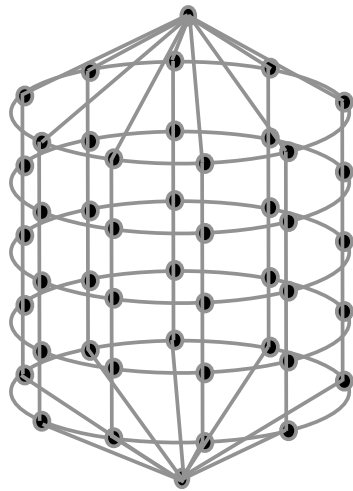


Tube

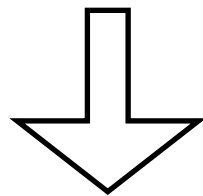


Capsule

Tessellating a Sphere

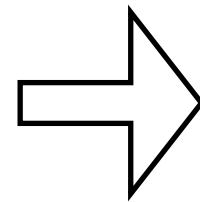


capsule grid



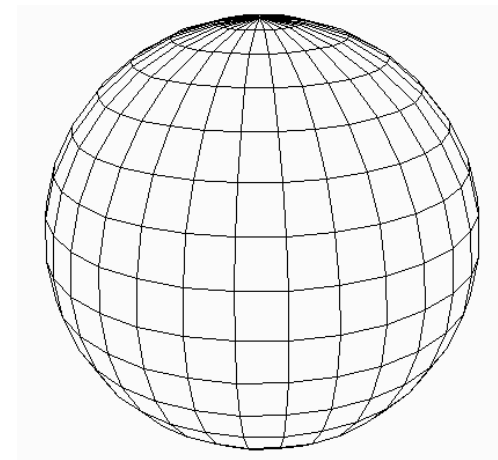
$$(u, v) = \begin{bmatrix} 0,1 \\ 0,3/4 & 1/8,3/4 & 2/8,3/4 & 3/8,3/4 & 4/8,3/4 & 5/8,3/4 & 6/8,3/4 & 7/8,3/4 & 1,3/4 \\ 0,2/4 & 1/8,2/4 & 2/8,2/4 & 3/8,2/4 & 4/8,2/4 & 5/8,2/4 & 6/8,2/4 & 7/8,2/4 & 1,2/4 \\ 0,1/4 & 1/8,1/4 & 2/8,1/4 & 3/8,1/4 & 4/8,1/4 & 5/8,1/4 & 6/8,1/4 & 7/8,1/4 & 1,1/4 \\ 0,0 \end{bmatrix}$$

parameterization

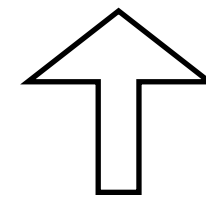


$$s(u, v) = \begin{bmatrix} r \sin(2\pi u) \sin(\pi v) \\ -r \cos(\pi v) \\ r \cos(2\pi u) \sin(\pi v) \end{bmatrix}$$

position function

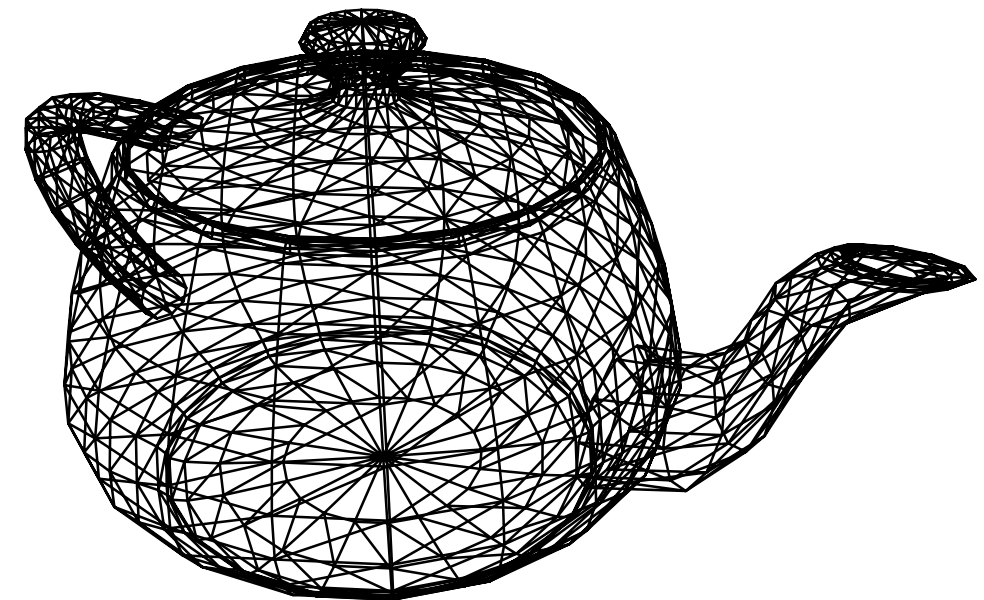


mesh



Polygon Mesh

- A **mesh** is a set of topologically related planar polygons. Often triangles.
- Represented as:
 - List of vertices
 - Set of indices into vertex list



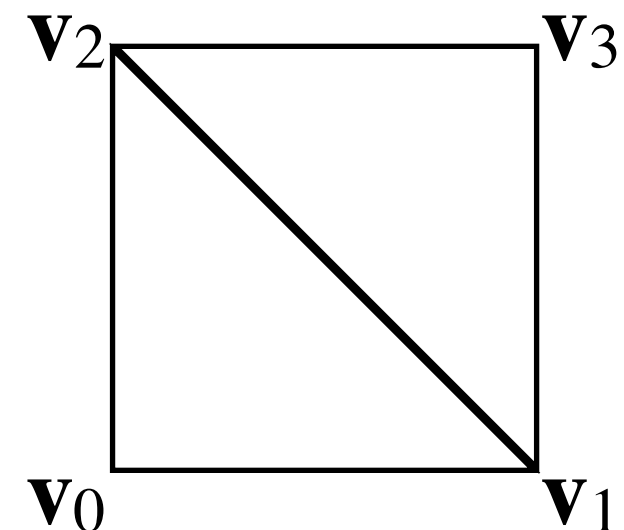
Vertices

0	$\mathbf{v}_0=(0,0,0)$
1	$\mathbf{v}_1=(1,0,0)$
2	$\mathbf{v}_2=(0,1,0)$
3	$\mathbf{v}_3=(1,1,0)$

Indices

Tri 0 : {0,1,2}

Tri 1 : {2,1,3}



In practice : OBJ format

```
mtllib elephant.mtl
```

```
g default
```

```
v -1.809173 8.068286 -6.605641
```

```
v -1.711079 8.213382 -6.732996
```

```
v -1.687598 8.087677 -6.827210
```

```
v -1.742430 7.929831 -6.708415
```

```
...
```

```
vn -0.896939 -0.062018 -0.437782
```

```
vn -0.877770 -0.046475 -0.476823
```

```
vn -0.913599 0.085210 -0.397587
```

```
vn -0.935617 -0.033324 -0.351440
```

```
....
```

```
f 1/1/1 2/2/2 3/3/3 4/4/4
```

```
f 2/2/2 5/5/5 6/6/6 3/3/3
```

```
f 3/3/3 6/6/6 7/7/7 8/8/8
```

```
f 4/4/4 3/3/3 8/8/8 9/9/9
```

```
f 10/10/10 11/11/11 12/12/12 13/13/13
```

```
f 11/11/11 14/14/14 15/15/15 12/12/12
```

```
f 12/12/12 15/15/15 1/1/1 16/16/16
```

```
f 13/13/13 12/12/12 16/16/16 17/17/17
```

```
....
```

```
f 2541/2501/2541 11092/10988/11092 11422/11318/11422 11372/11268/11372
```

```
f 11092/10988/11092 1590/1570/1590 7674/7573/7674 11422/11318/11422
```

```
f 11422/11318/11422 7674/7573/7674 7673/7572/7673 11421/11317/11421
```

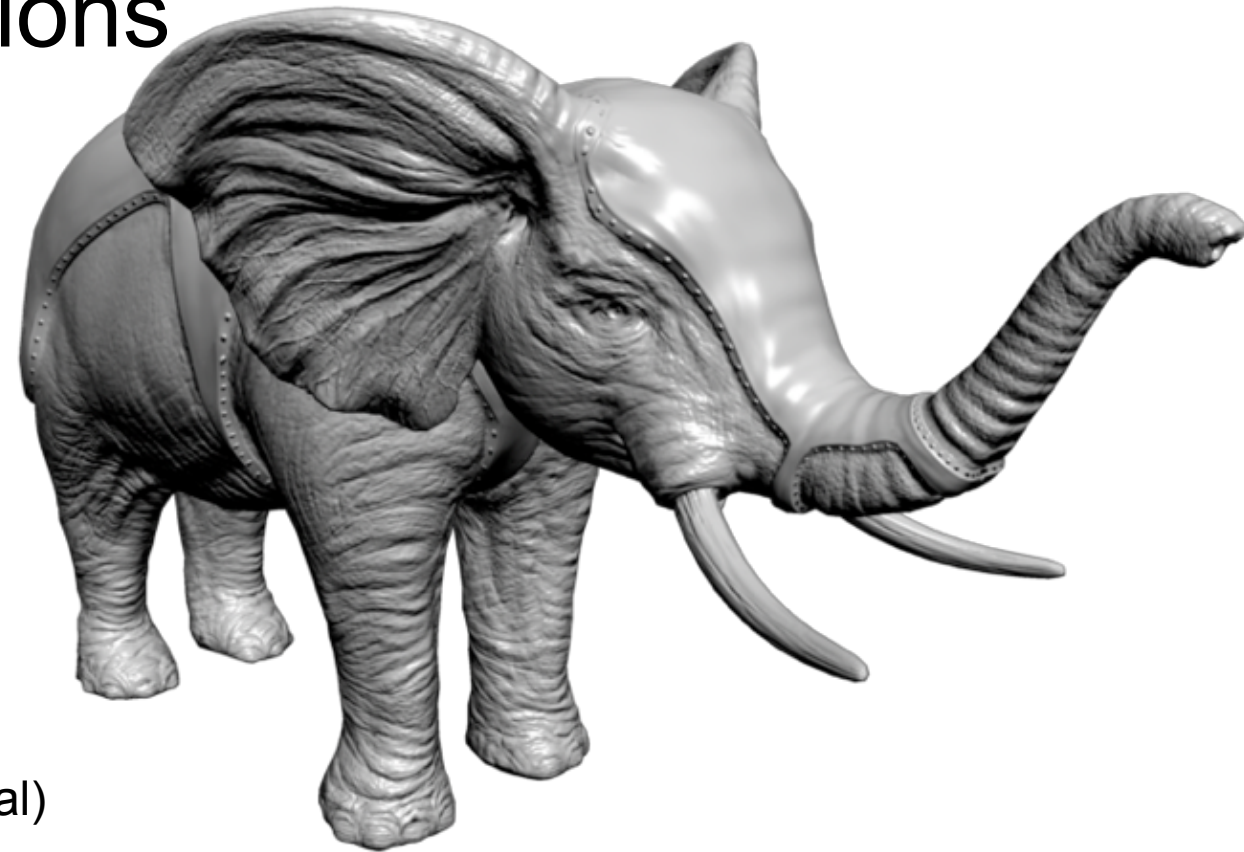
```
f 11372/11268/11372 11422/11318/11422 11421/11317/11421 2562/2522/2562
```

vertex positions

normals

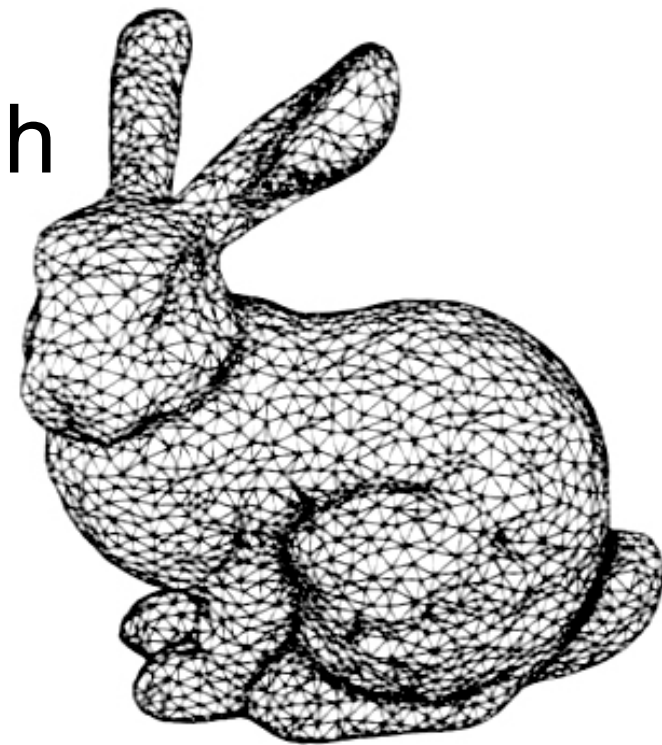
faces

index to
(vertex/texcoords/normal)



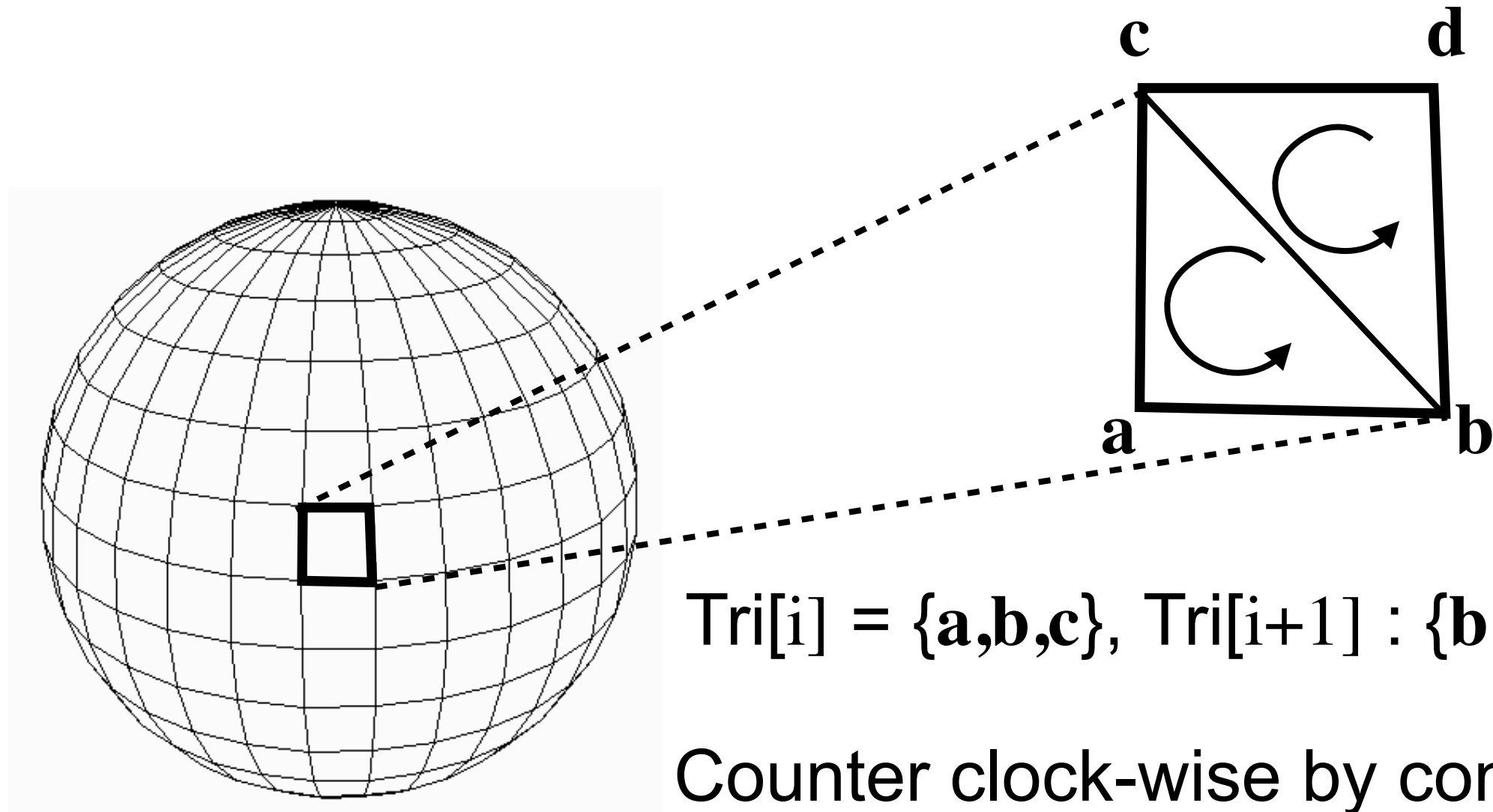
How to create meshes?

- Modeling packages
 - Blender, Maya, 3D Studio Max, ZBrush
- From point clouds,
 - i.e., 3D scan data
- This course only covers mesh construction from simple parametric surfaces
 - Feel free to import your own meshes in Labs!



Triangulation

- Create triangles from mesh faces



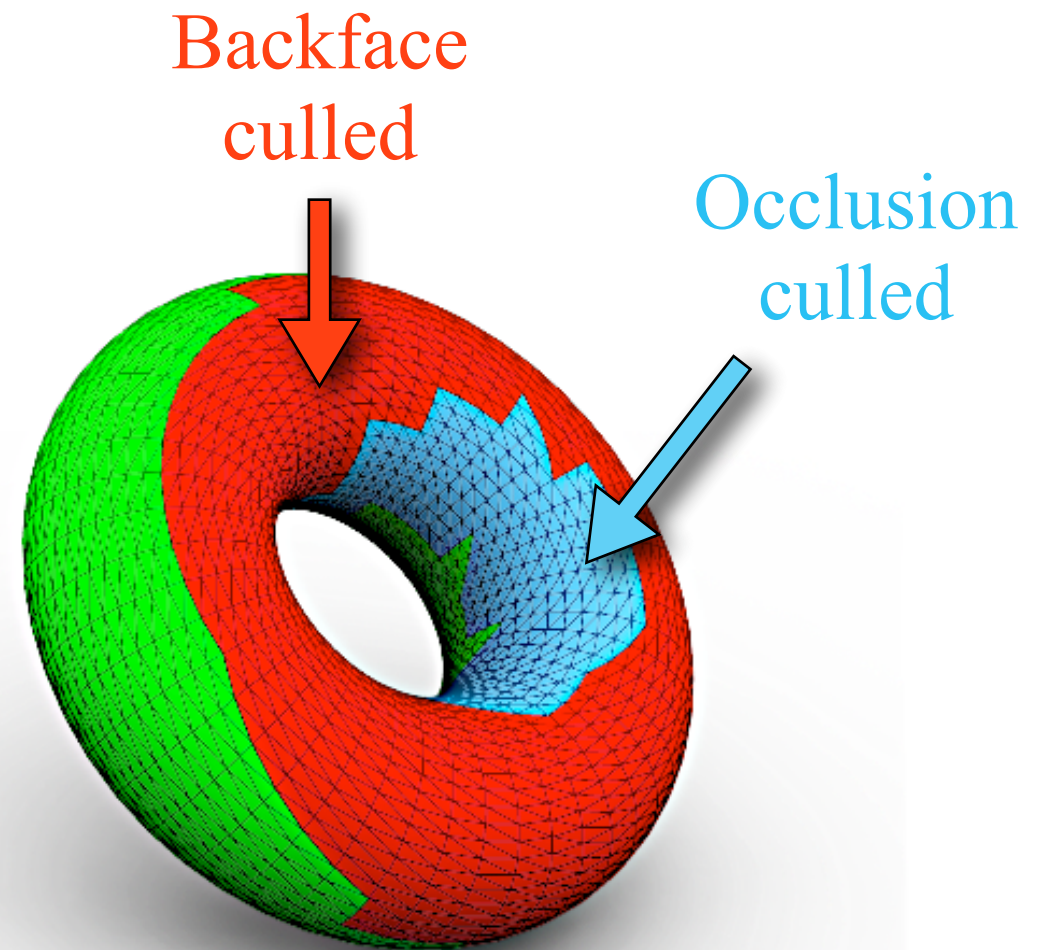
$\text{Tri}[i] = \{a,b,c\}, \text{Tri}[i+1] : \{b,d,c\}$

Counter clock-wise by convention

Winding order determines front and back faces

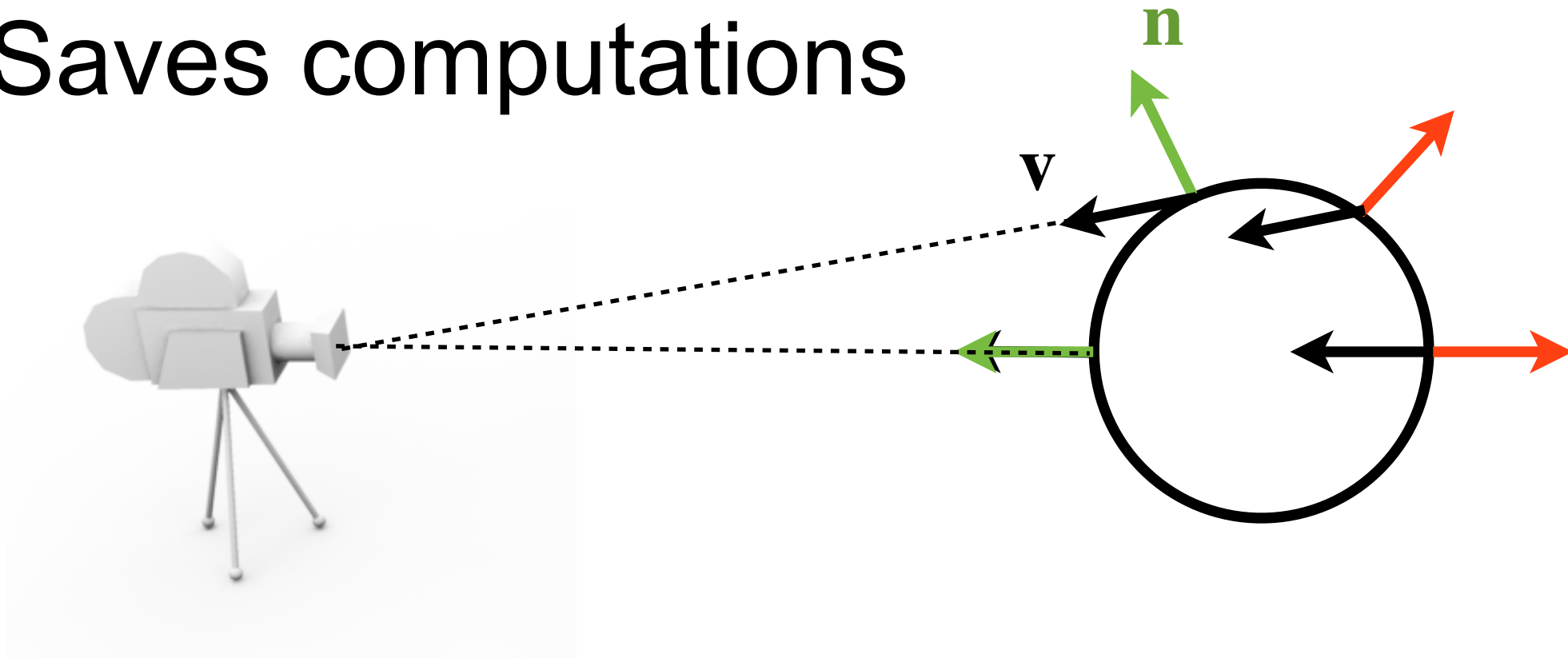
Backface Culling

- Remove triangles facing away from the camera



Backface Culling

- Remove triangles facing away from the camera
- Saves computations



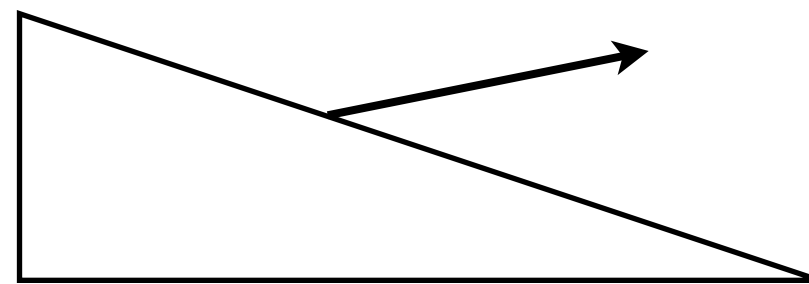
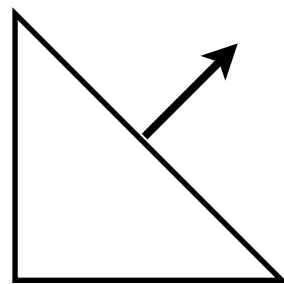
Backfacing if $\mathbf{n} \cdot \mathbf{v} < 0$

Transforming Normals

- Can't transform normals as other vectors and points

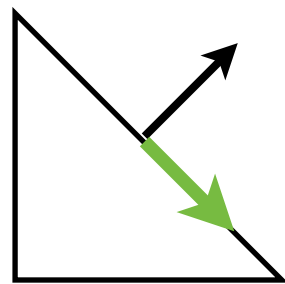
– Example: Non-uniform scaling $\mathbf{v}' = \begin{bmatrix} 3 & 0 \\ 0 & 1 \end{bmatrix} \mathbf{v}$

$$\mathbf{v}' = \mathbf{M}\mathbf{v}$$

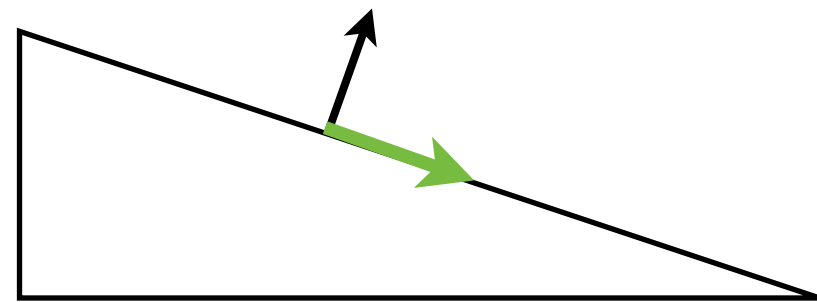


Transforming Normals

- Exploit that the tangent and normal should be perpendicular before and after transform, i.e., $\mathbf{n} \cdot \mathbf{t} = 0$



$$\mathbf{n} \cdot \mathbf{t} = 0$$



$$\mathbf{n}' \cdot \mathbf{t}' = 0$$

Tangent is difference between two points on surface - transforms as surface points

Transforming Normals

- Tangent

- Transformed with the matrix **M** same as used for points and vectors $\mathbf{t}' = \mathbf{M}\mathbf{t}$

- Normal

- Transformed with an **unknown** matrix **Q** $\mathbf{n}' = \mathbf{Q}\mathbf{n}$
- Exploit $\mathbf{n}' \cdot \mathbf{t}' = 0$

Transforming Normals

- Exploit $\mathbf{n} \cdot \mathbf{t} = 0$ $\mathbf{t}' = \mathbf{M}\mathbf{t}$
 $\mathbf{n}' \cdot \mathbf{t}' = 0$ $\mathbf{n}' = \mathbf{Q}\mathbf{n}$

$$\mathbf{n}' \cdot \mathbf{t}' = 0$$

$$\mathbf{n}'^T \mathbf{t}' = 0$$

$$(\mathbf{Q}\mathbf{n})^T (\mathbf{M}\mathbf{t}) = 0$$

$$\mathbf{n}^T (\mathbf{Q}^T \mathbf{M}) \mathbf{t} = 0$$

$$\mathbf{n} \cdot \mathbf{t} = 0$$

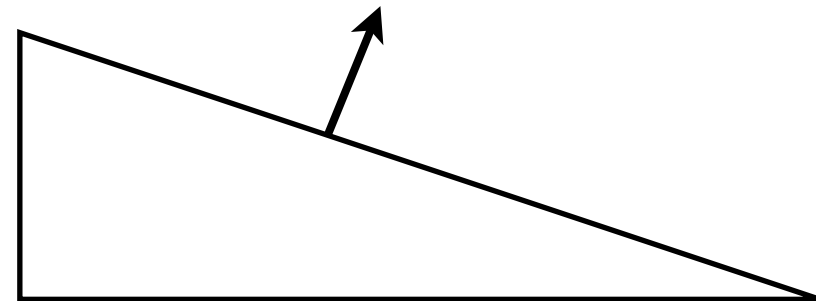
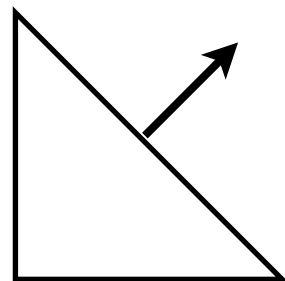
$$\mathbf{Q}^T \mathbf{M} = \mathbf{I} \Leftrightarrow \mathbf{Q} = (\mathbf{M}^{-1})^T$$

Use $(\mathbf{M}^{-1})^T$ to transform normals!

Transforming Normals

- Return to our non-uniform scaling example:

$$\mathbf{v}' = \begin{bmatrix} 3 & 0 \\ 0 & 1 \end{bmatrix} \mathbf{v}$$
$$\mathbf{n}' = \begin{bmatrix} \frac{1}{3} & 0 \\ 0 & 1 \end{bmatrix} \mathbf{n}$$



$$\mathbf{M} = \begin{bmatrix} 3 & 0 \\ 0 & 1 \end{bmatrix}$$

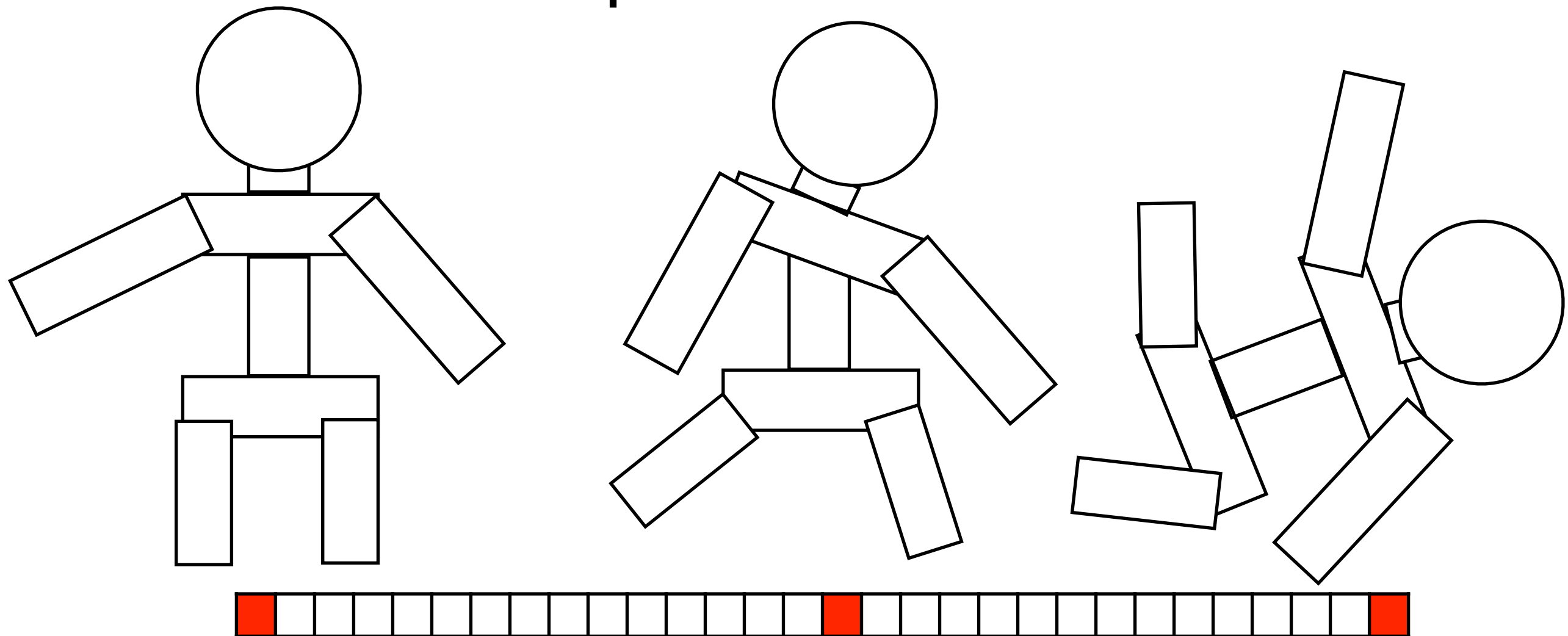
$$\mathbf{M}^{-1} = (\mathbf{M}^{-1})^T = \begin{bmatrix} \frac{1}{3} & 0 \\ 0 & 1 \end{bmatrix}$$

Animating Objects

- We know how to position objects
- We know how to move objects
- How to move objects **smoothly** from A to B over time?

Key-frame Animation

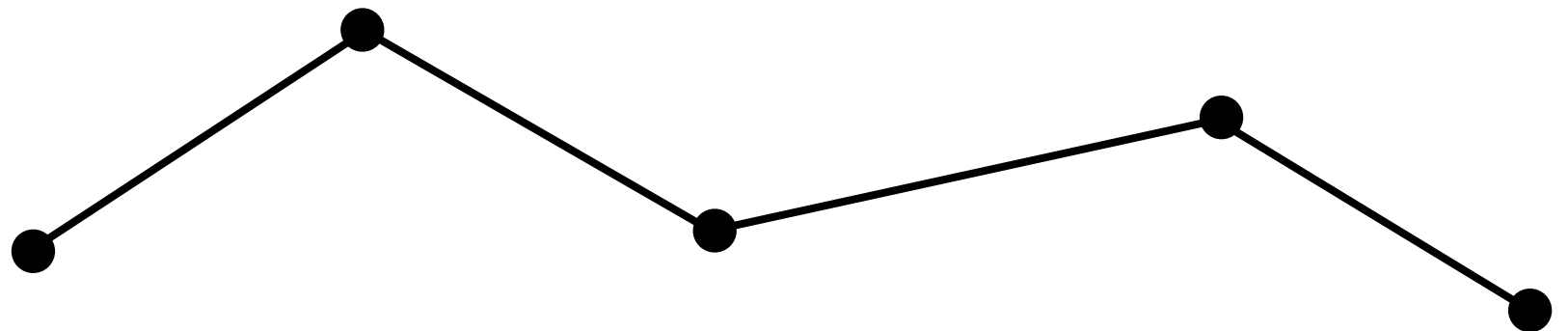
- The animator positions the object at a set of times - the key frames, then the software interpolates between frames



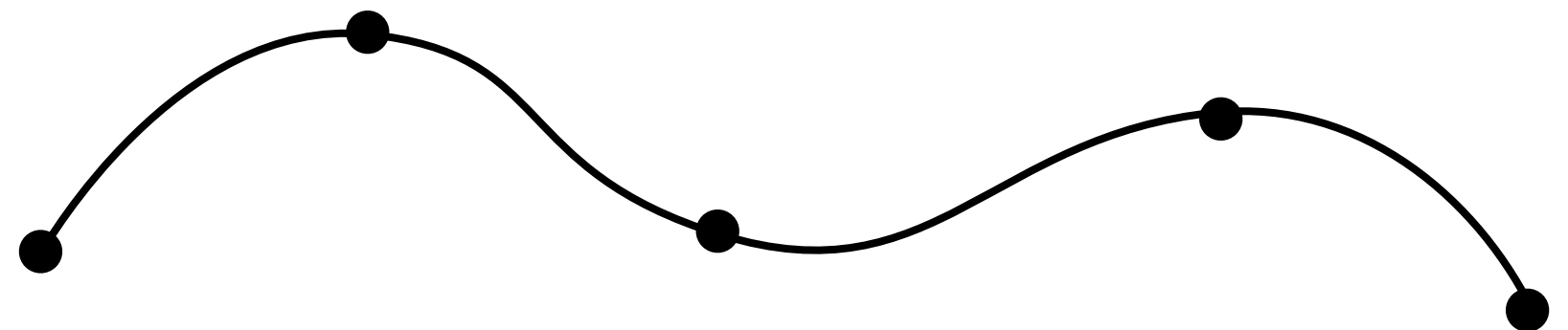
Path Following

- Define path by a point list
- Interpolate points in-between

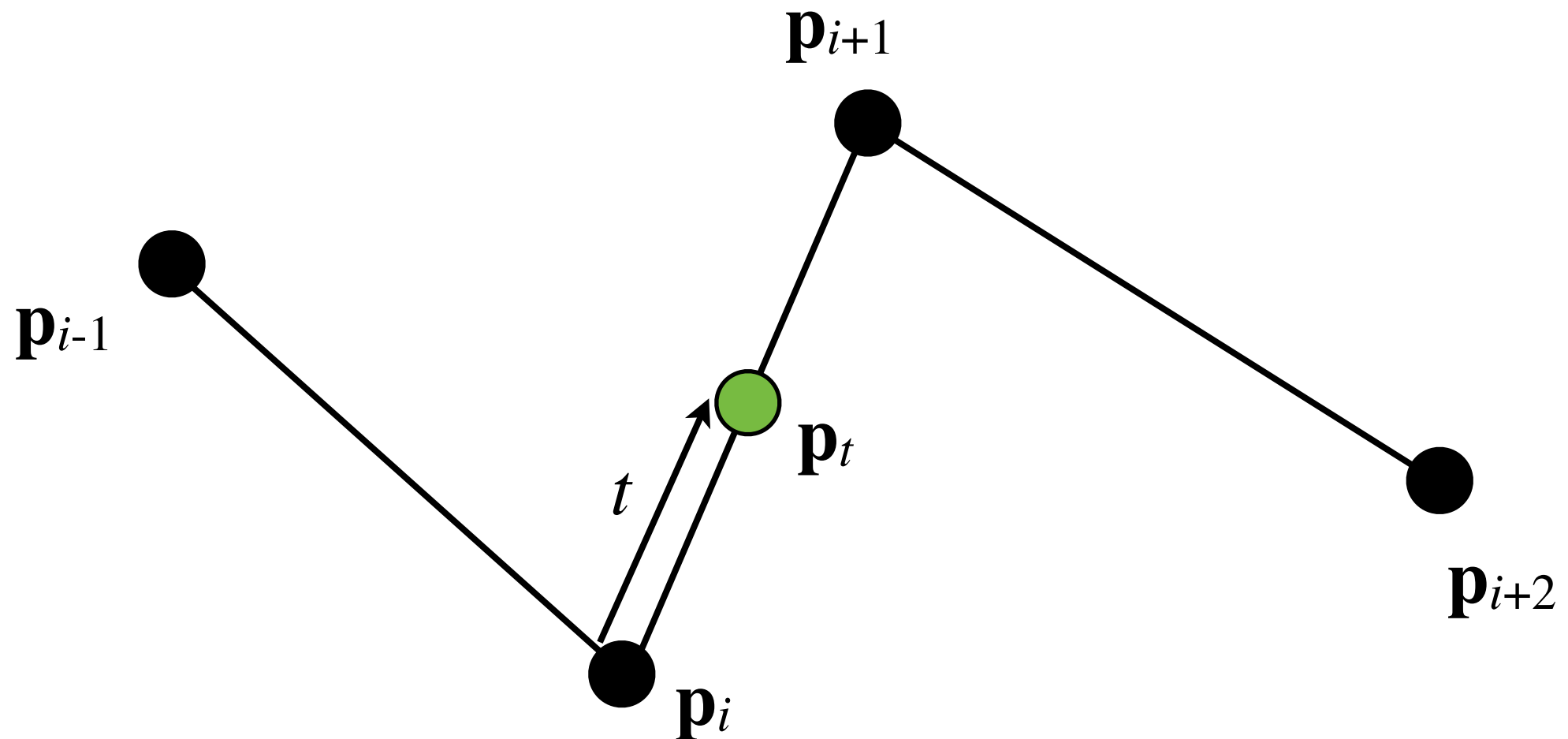
Linear
interpolation



Cubic
interpolation



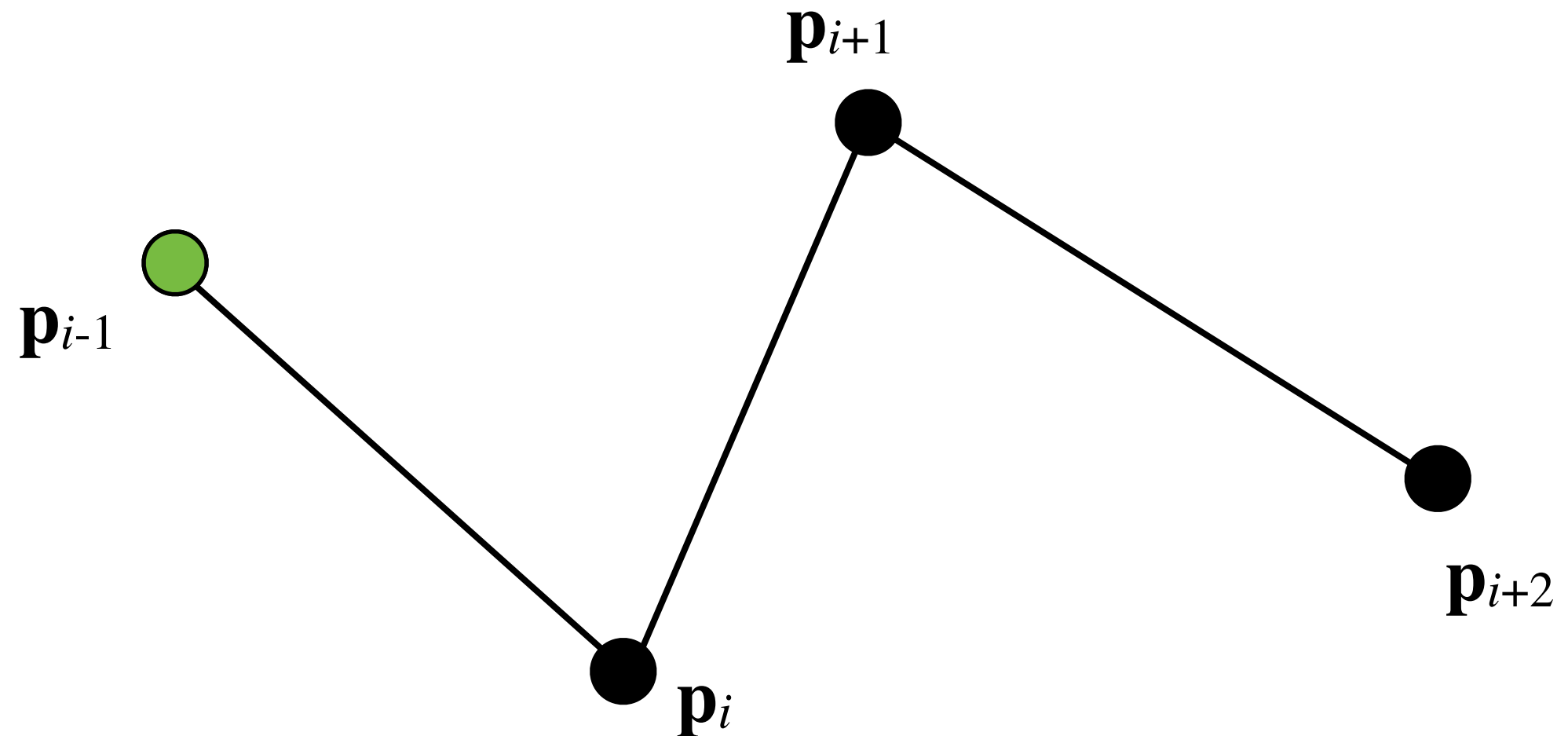
Linear Interpolation



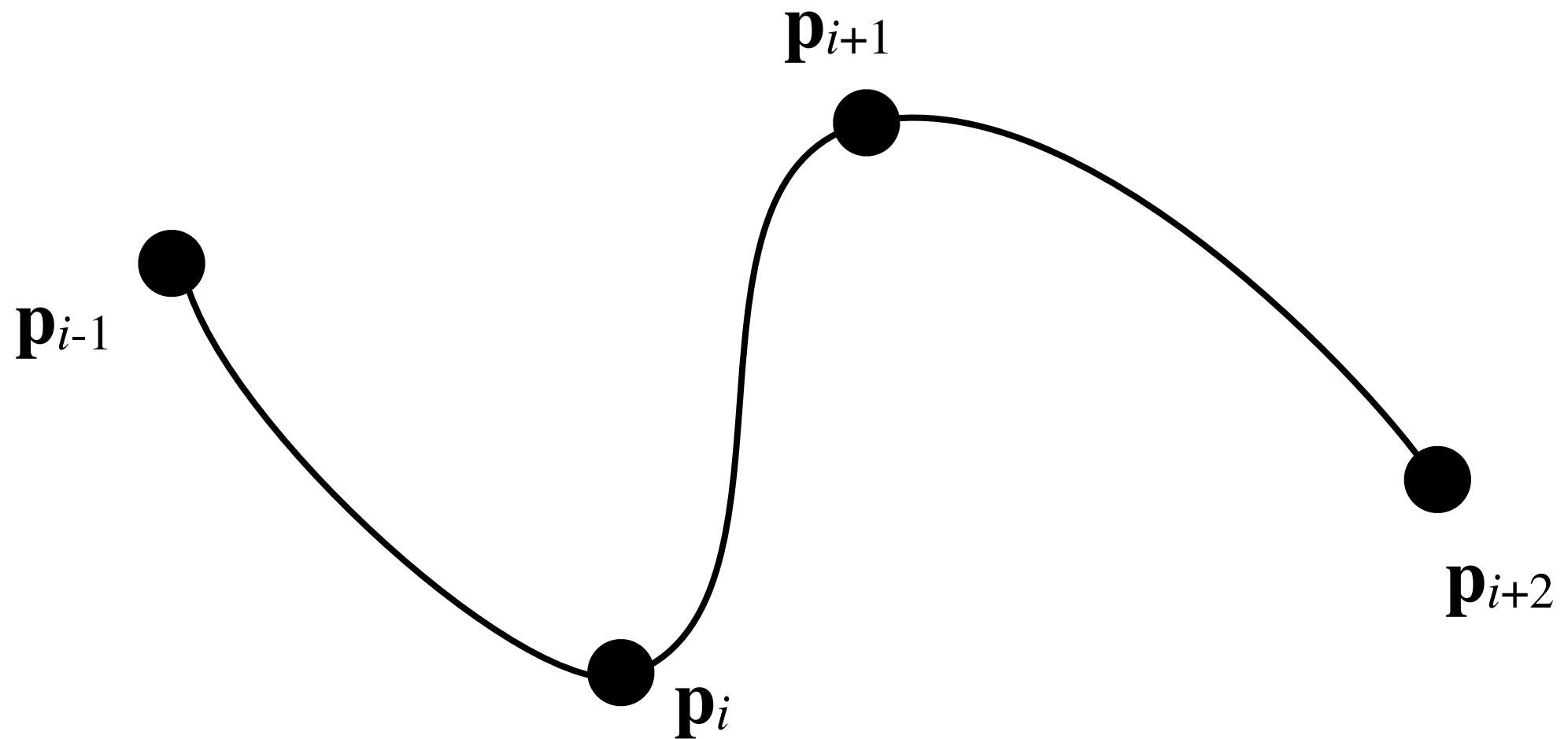
$$\mathbf{p}_t = (1 - t)\mathbf{p}_i + t\mathbf{p}_{i+1}, t \in [0, 1]$$

$$\mathbf{p}_t = \begin{bmatrix} 1 & t \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{p}_i \\ \mathbf{p}_{i+1} \end{bmatrix}$$

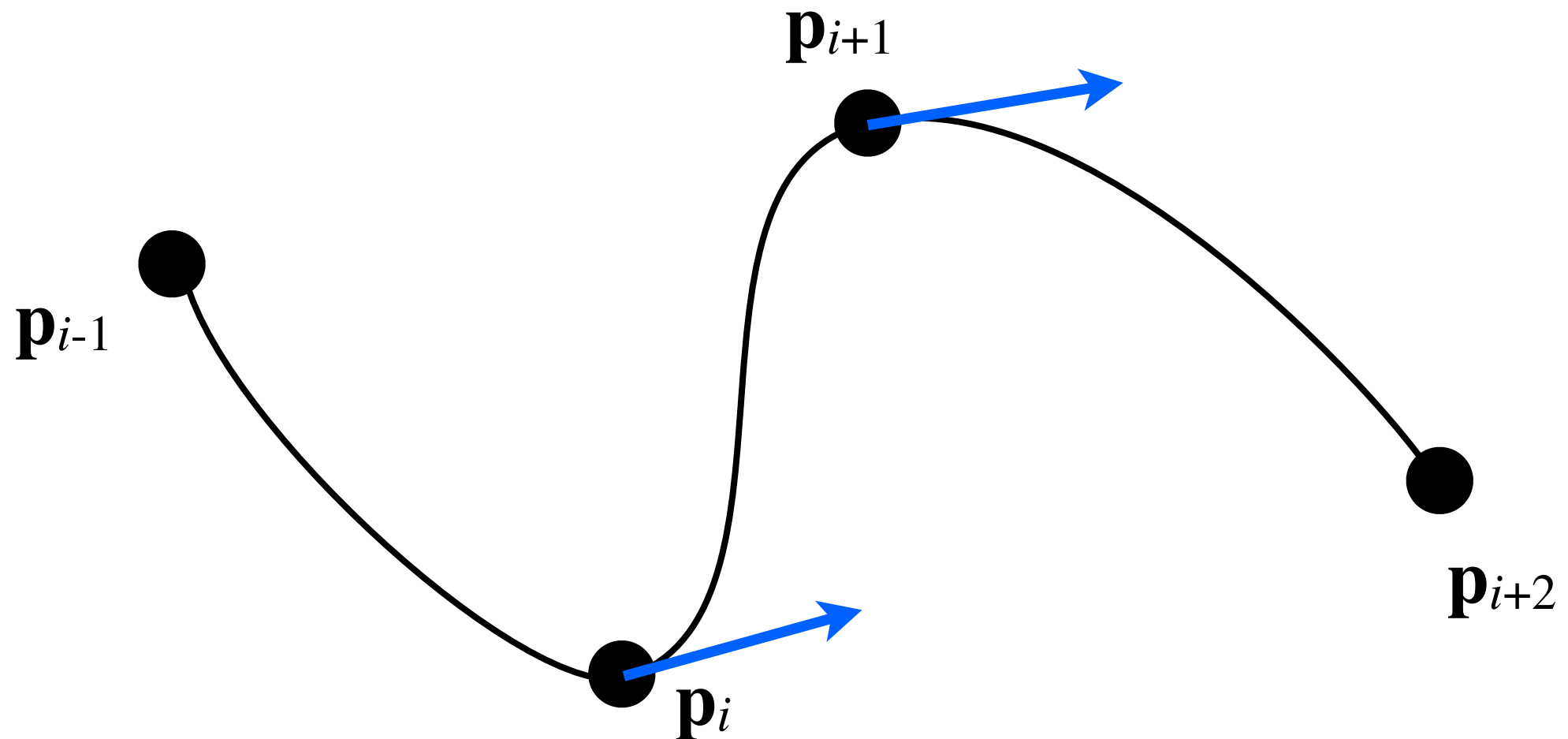
Example (Linear)



Example (Cubic)



Example (Cubic)



**Use tangent information
to specify curve segment**

Cubic Hermite

Interpolation

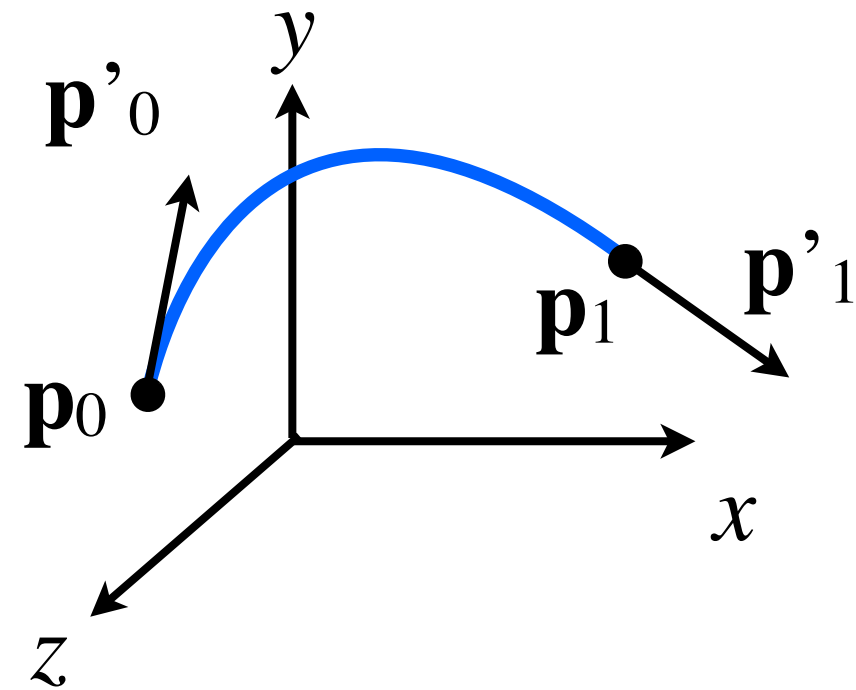
- Given two points and corresponding tangents:

$$\mathbf{p}_0, \mathbf{p}'_0 \text{ and } \mathbf{p}_1, \mathbf{p}'_1$$

we want to find a cubic curve

$$\mathbf{p}(t) = \mathbf{c}_0 + \mathbf{c}_1 t + \mathbf{c}_2 t^2 + \mathbf{c}_3 t^3$$

- Task at hand: Find coefficients \mathbf{c}_i .



Cubic Hermite

Interpolation

- Create system of equations

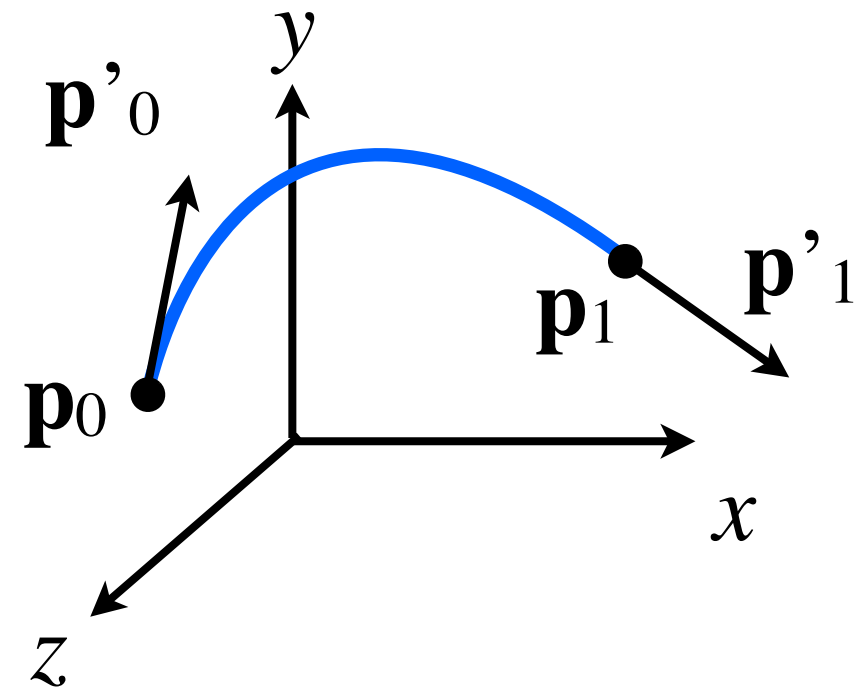
$$\mathbf{p}(t) = \mathbf{c}_0 + \mathbf{c}_1 t + \mathbf{c}_2 t^2 + \mathbf{c}_3 t^3$$

$$\mathbf{p}_0 = \mathbf{p}(0) = \mathbf{c}_0$$

$$\mathbf{p}_1 = \mathbf{p}(1) = \mathbf{c}_0 + \mathbf{c}_1 + \mathbf{c}_2 + \mathbf{c}_3$$

$$\mathbf{p}'_0 = \mathbf{p}'(0) = \mathbf{c}_1$$

$$\mathbf{p}'_1 = \mathbf{p}'(1) = \mathbf{c}_1 + 2\mathbf{c}_2 + 3\mathbf{c}_3$$



Cubic Hermite

Interpolation

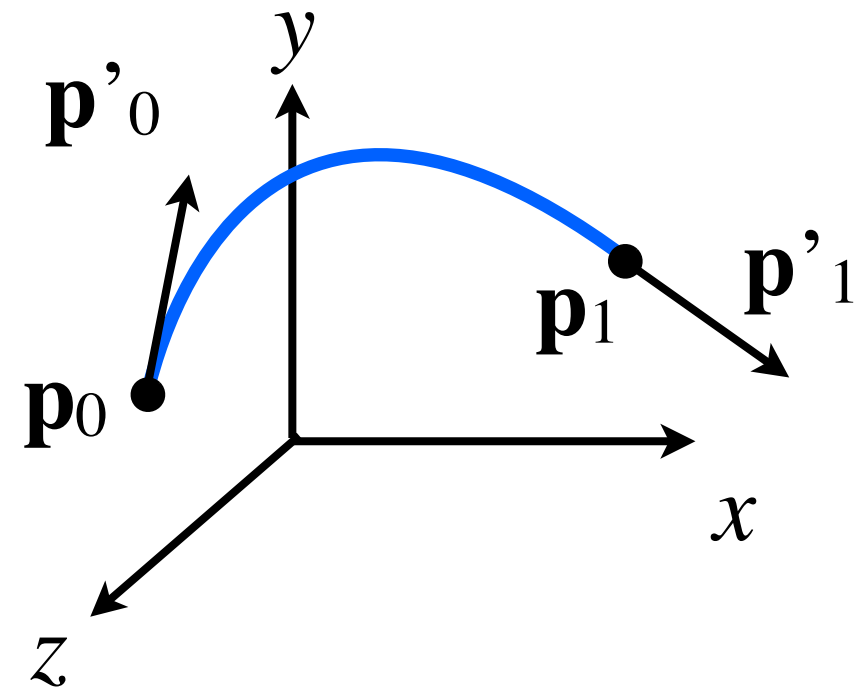
- Write in matrix form

$$\mathbf{p}(t) = \mathbf{c}_0 + \mathbf{c}_1 t + \mathbf{c}_2 t^2 + \mathbf{c}_3 t^3$$

$$\begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}'_0 \\ \mathbf{p}'_1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} \mathbf{c}_0 \\ \mathbf{c}_1 \\ \mathbf{c}_2 \\ \mathbf{c}_3 \end{bmatrix}$$

$$\mathbf{q} = \mathbf{M}\mathbf{c}$$

$$\mathbf{q} = \mathbf{M}\mathbf{c} \Leftrightarrow \mathbf{c} = \mathbf{M}^{-1}\mathbf{q}$$



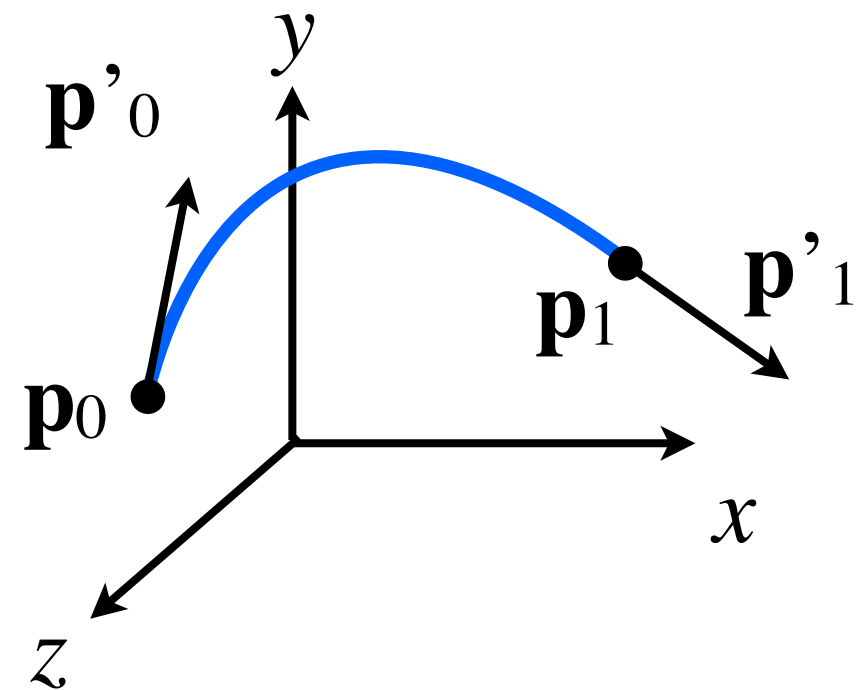
Cubic Hermite

Interpolation

- Write on matrix form

$$\mathbf{p}(t) = \mathbf{c}_0 + \mathbf{c}_1 t + \mathbf{c}_2 t^2 + \mathbf{c}_3 t^3$$

$$\begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}'_0 \\ \mathbf{p}'_1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 2 & 3 \end{bmatrix} \begin{bmatrix} \mathbf{c}_0 \\ \mathbf{c}_1 \\ \mathbf{c}_2 \\ \mathbf{c}_3 \end{bmatrix}$$



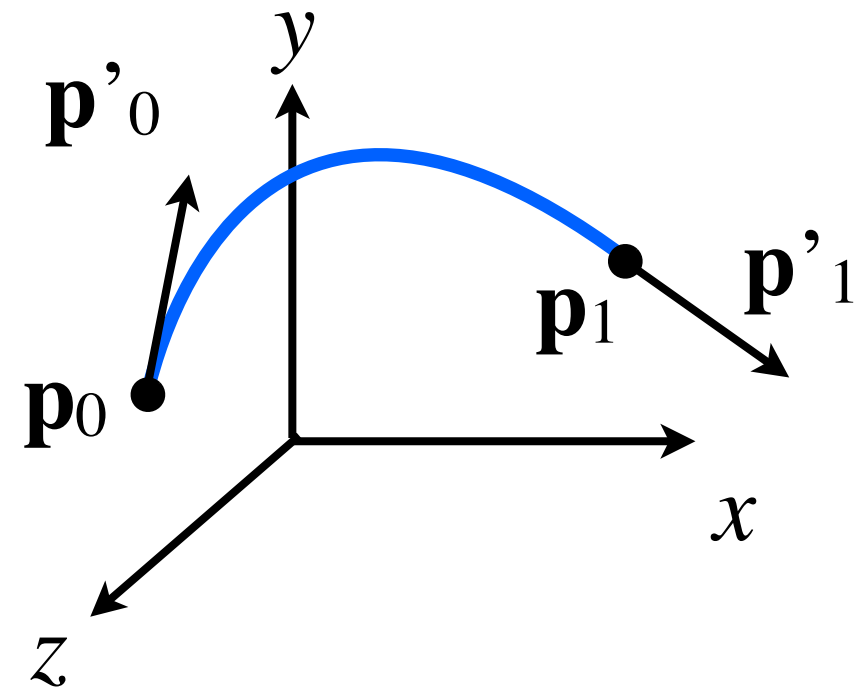
$$\begin{bmatrix} \mathbf{c}_0 \\ \mathbf{c}_1 \\ \mathbf{c}_2 \\ \mathbf{c}_3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 2 & 3 \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}'_0 \\ \mathbf{p}'_1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -3 & 3 & -2 & -1 \\ 2 & -2 & 1 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}'_0 \\ \mathbf{p}'_1 \end{bmatrix}$$

Cubic Hermite

Interpolation

$$\mathbf{p}(t) = \mathbf{c}_0 + \mathbf{c}_1 t + \mathbf{c}_2 t^2 + \mathbf{c}_3 t^3$$

$$\mathbf{p}(t) = [1 \ t \ t^2 \ t^3] \begin{bmatrix} \mathbf{c}_0 \\ \mathbf{c}_1 \\ \mathbf{c}_2 \\ \mathbf{c}_3 \end{bmatrix}$$

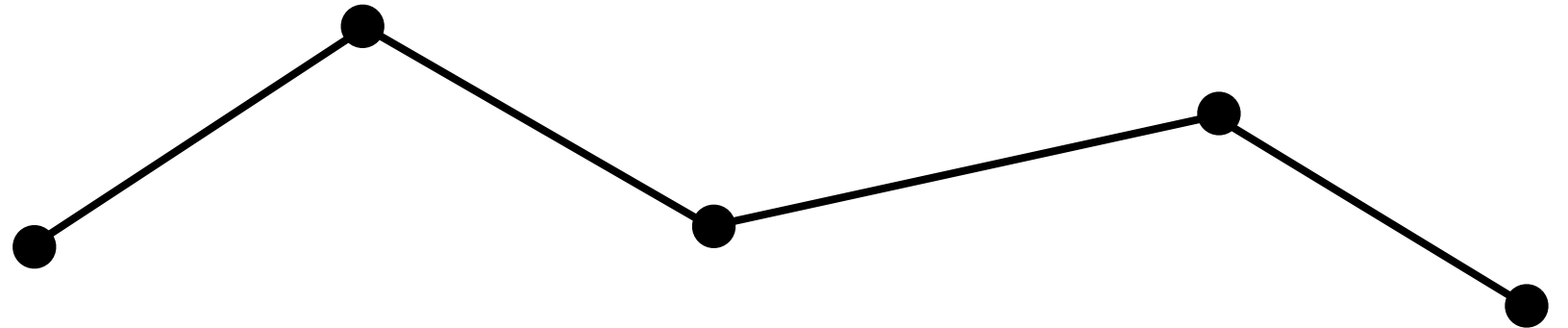


$$\mathbf{p}(t) = [1 \ t \ t^2 \ t^3] \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -3 & 3 & -2 & -1 \\ 2 & -2 & 1 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}'_0 \\ \mathbf{p}'_1 \end{bmatrix}$$

$$\mathbf{p}(t) = \mathbf{t}^T \mathbf{M}^{-1} \mathbf{q}$$

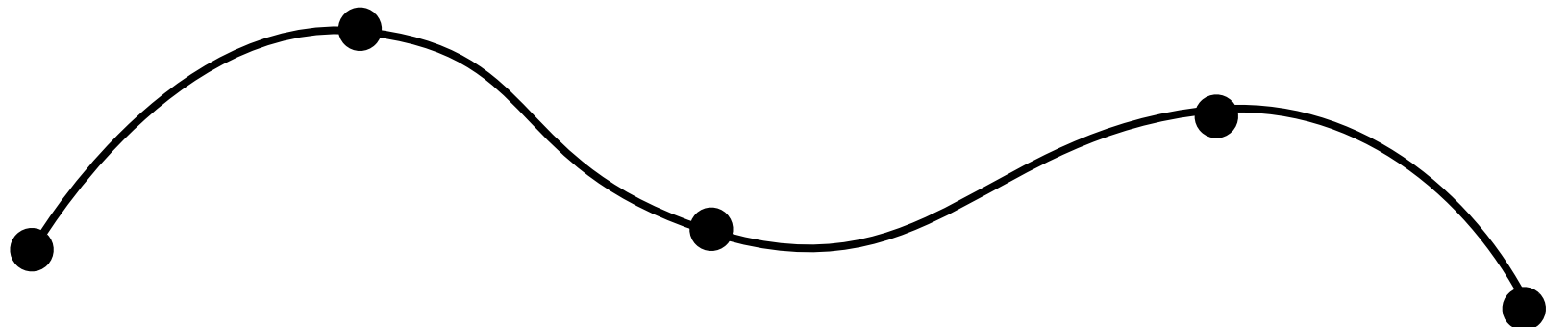
Linear vs Cubic

Linear
interpolation



$$\mathbf{p}_t = \begin{bmatrix} 1 & t \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{p}_i \\ \mathbf{p}_{i+1} \end{bmatrix}$$

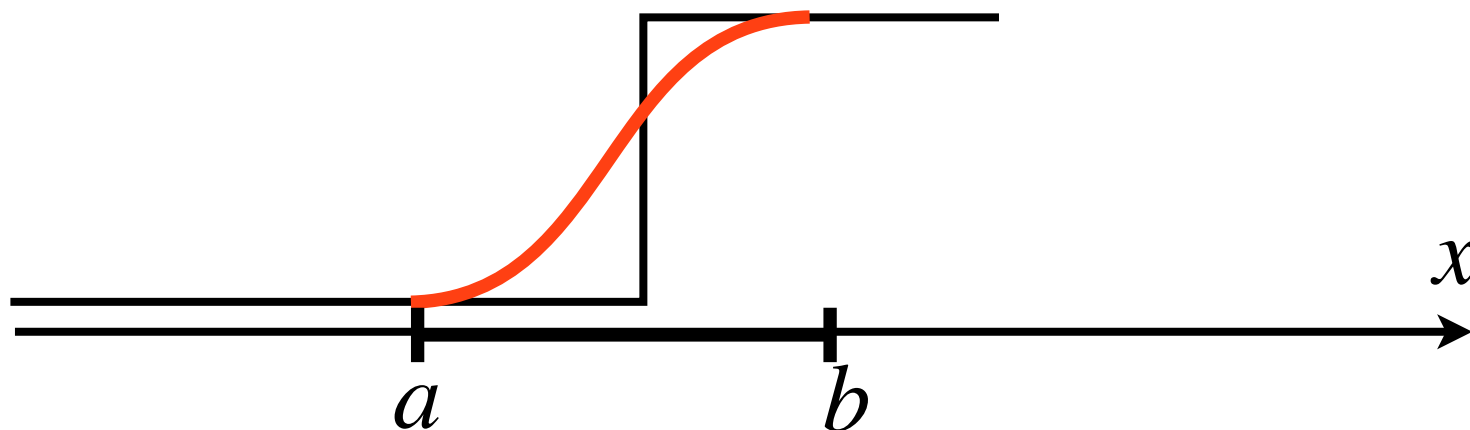
Cubic
interpolation



$$\mathbf{p}(t) = \begin{bmatrix} 1 & t & t^2 & t^3 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -3 & 3 & -2 & -1 \\ 2 & -2 & 1 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}'_0 \\ \mathbf{p}'_1 \end{bmatrix}$$

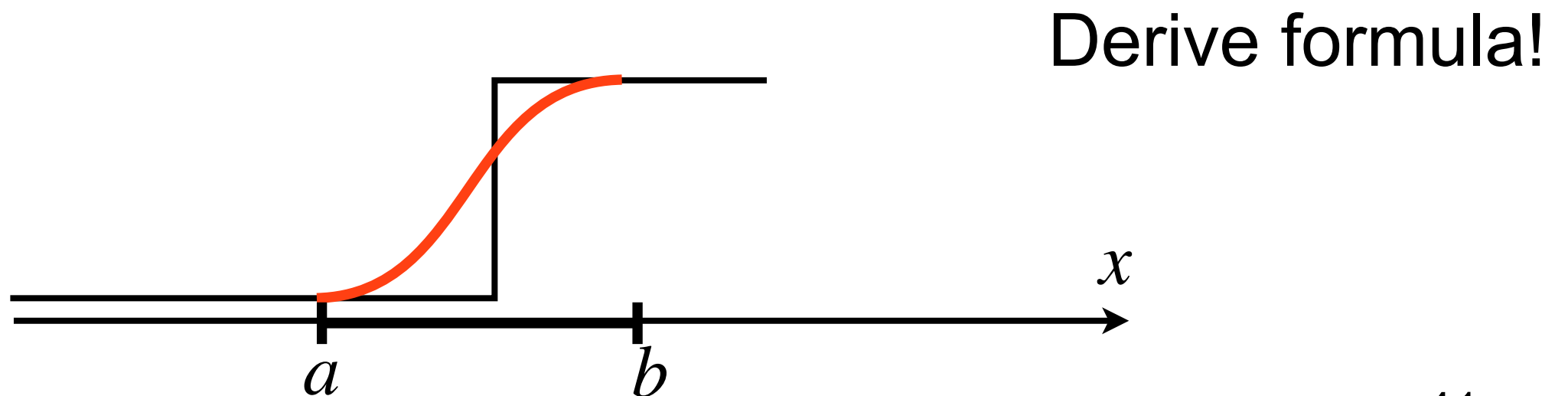
Smoothstep

- Useful special (1D) case of cubic interpolation
 - performs smooth Hermite interpolation between 0 and 1 when $a < x < b$.
 - Useful in cases where a threshold function with a smooth transition is desired.
 - Tangents : 0 at $x=a$ and 0 at $x=b$



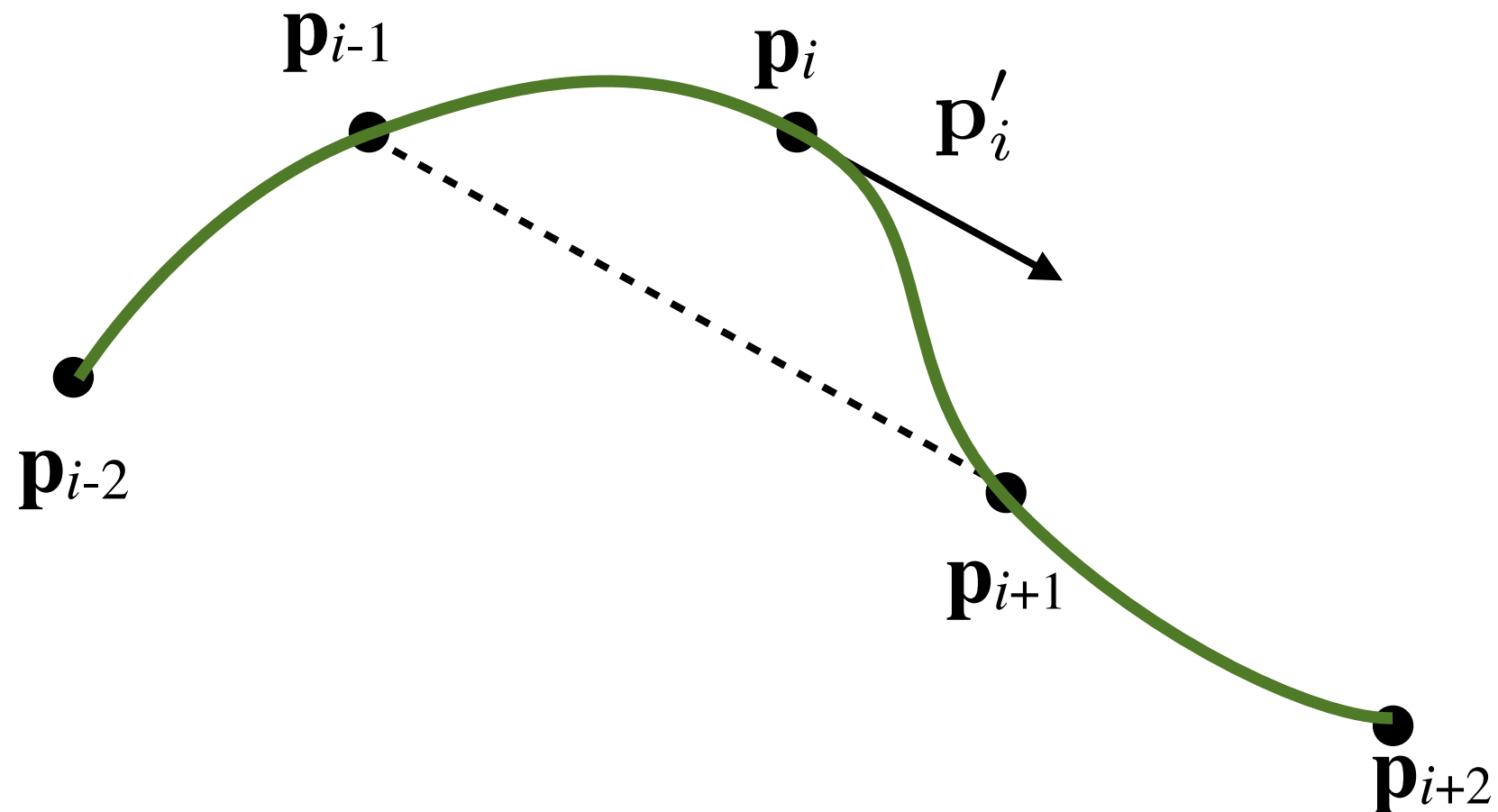
Smoothstep

```
float smoothstep(float x, float a, float b)
{
    float t = clamp((x-a)/(b-a), 0, 1);
    return t*t*(3-2*t);
}
```



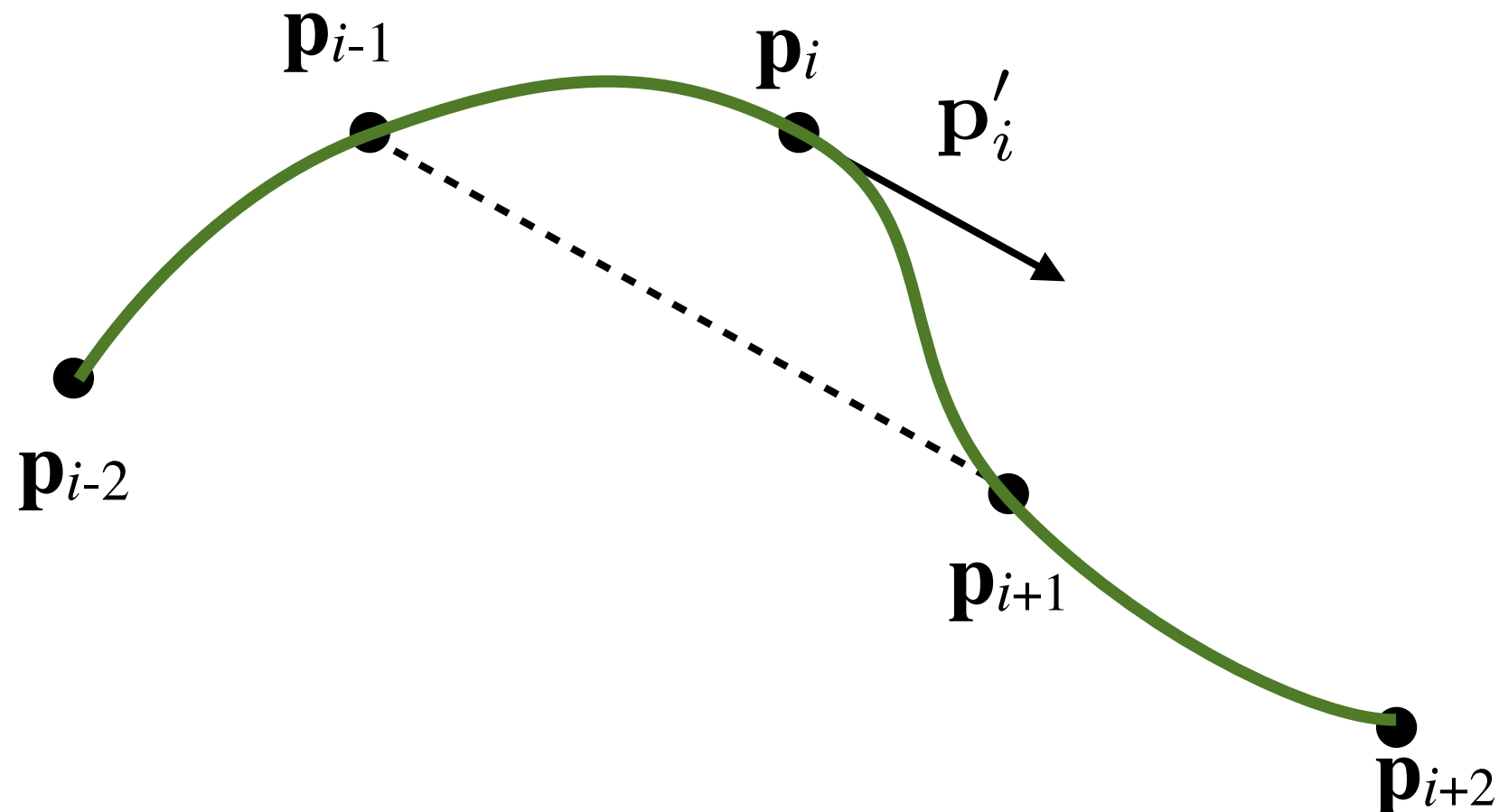
How to specify tangents?

- One simple solution: $\mathbf{p}'_i \approx \frac{\mathbf{p}_{i+1} - \mathbf{p}_{i-1}}{2}$

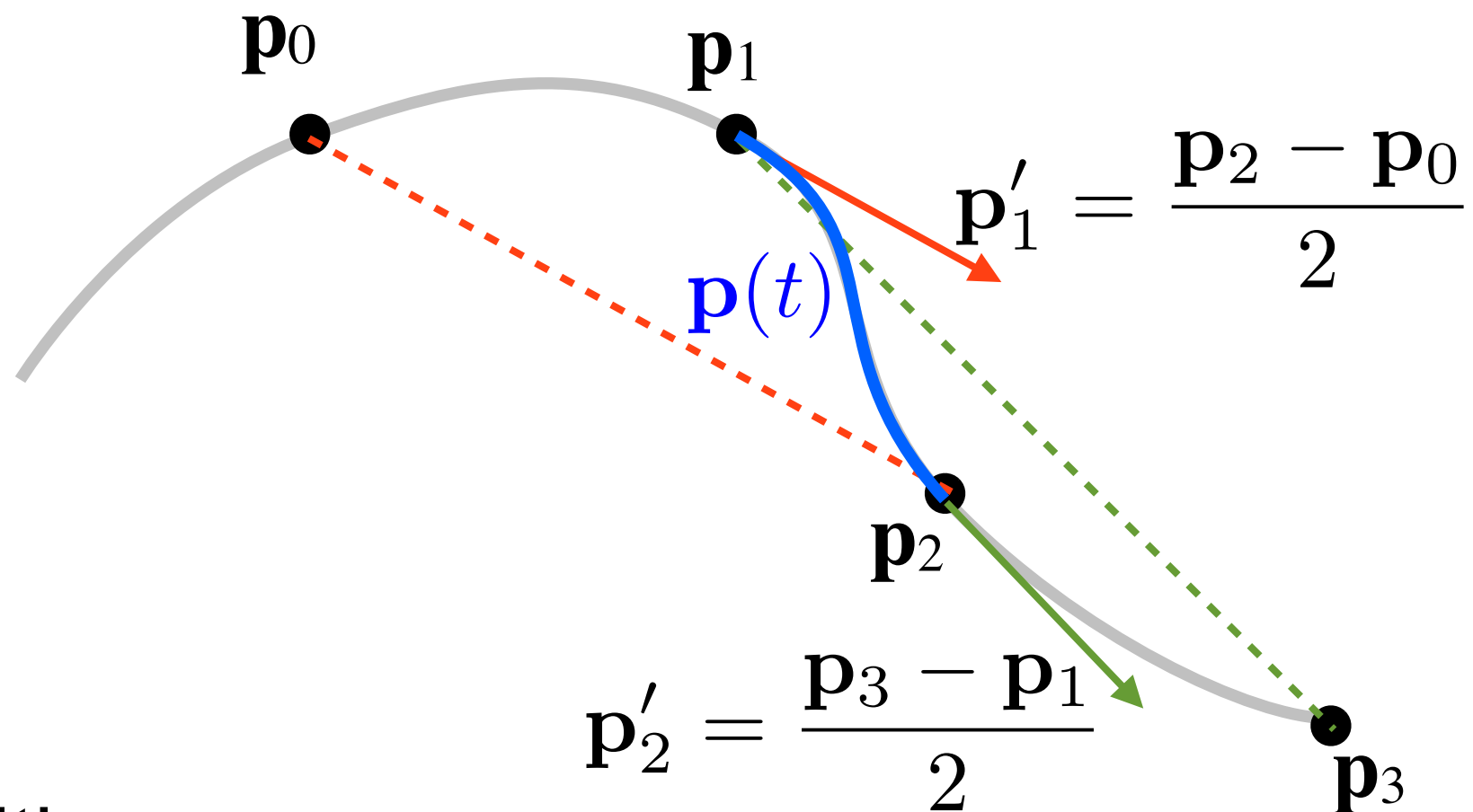


Catmull-Rom Splines

- Given a set of points, define tangents
as: $\mathbf{p}'_i \approx \frac{\mathbf{p}_{i+1} - \mathbf{p}_{i-1}}{2}$
 - Curve segment between \mathbf{p}_i and \mathbf{p}_{i+1} completely defined by \mathbf{p}_{i-1} , \mathbf{p}_i , \mathbf{p}_{i+1} and \mathbf{p}_{i+2}



Catmull-Rom Splines



Four conditions:

$$\begin{bmatrix} \mathbf{p}(0) \\ \mathbf{p}(1) \\ \mathbf{p}'(0) \\ \mathbf{p}'(1) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -1/2 & 0 & 1/2 & 0 \\ 0 & -1/2 & 0 & 1/2 \end{bmatrix} \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{bmatrix}$$

Catmull-Rom Splines

- Tangent def. gives four conditions

$$\begin{bmatrix} \mathbf{p}(0) \\ \mathbf{p}(1) \\ \mathbf{p}'(0) \\ \mathbf{p}'(1) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -1/2 & 0 & 1/2 & 0 \\ 0 & -1/2 & 0 & 1/2 \end{bmatrix} \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \mathbf{p}_2 \\ \mathbf{p}_3 \end{bmatrix}$$

$$\mathbf{q} = \mathbf{N}\mathbf{p}$$

- Insert in Hermite formula:

$$\mathbf{p}(t) = \mathbf{t}^T \mathbf{M}^{-1} \mathbf{q} = \mathbf{t}^T \mathbf{M}^{-1} \mathbf{N} \mathbf{p}$$

$$\mathbf{M}^{-1} \mathbf{N} = \frac{1}{2} \begin{bmatrix} 0 & 2 & 0 & 0 \\ -1 & 0 & 1 & 0 \\ 2 & -5 & 4 & -1 \\ -1 & 3 & -3 & 1 \end{bmatrix}$$

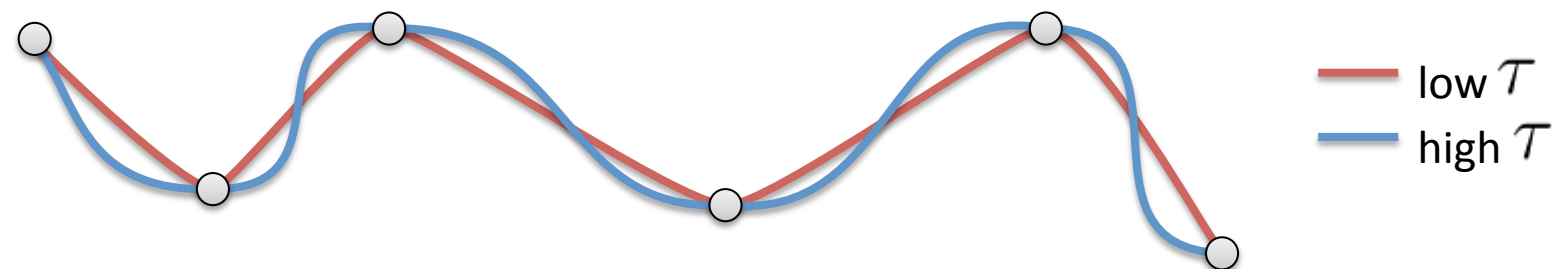
Catmull-Rom Splines

- If we add a tension parameter τ :

$$\mathbf{p}'_i = \tau(\mathbf{p}_{i+1} - \mathbf{p}_{i-1}), \tau \in [0, 1]$$

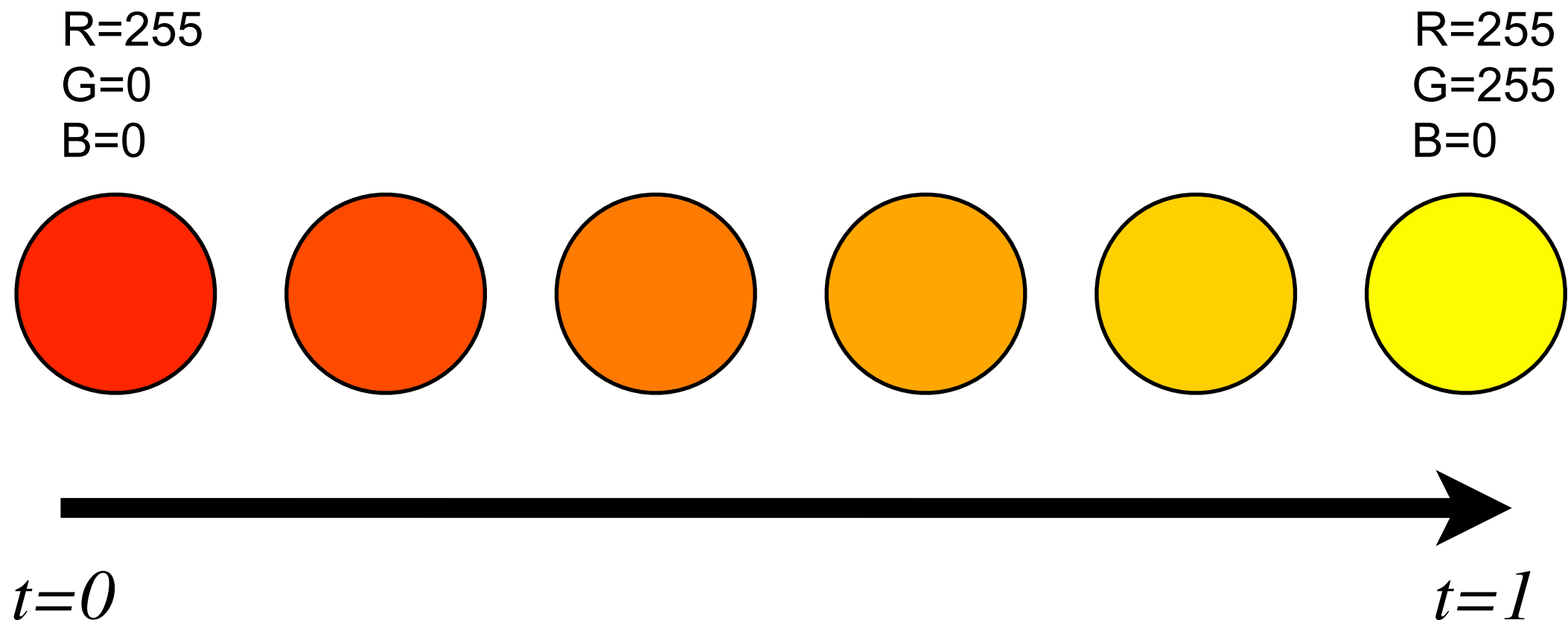
the Catmull-Rom spline can be generalized to:

$$\mathbf{p}(t) = [1 \ t \ t^2 \ t^3] \begin{bmatrix} 0 & 1 & 0 & 0 \\ -\tau & 0 & \tau & 0 \\ 2\tau & \tau - 3 & 3 - 2\tau & -\tau \\ -\tau & 2 - \tau & \tau - 2 & \tau \end{bmatrix} \begin{bmatrix} \mathbf{p}_{i-1} \\ \mathbf{p}_i \\ \mathbf{p}_{i+1} \\ \mathbf{p}_{i+2} \end{bmatrix}$$



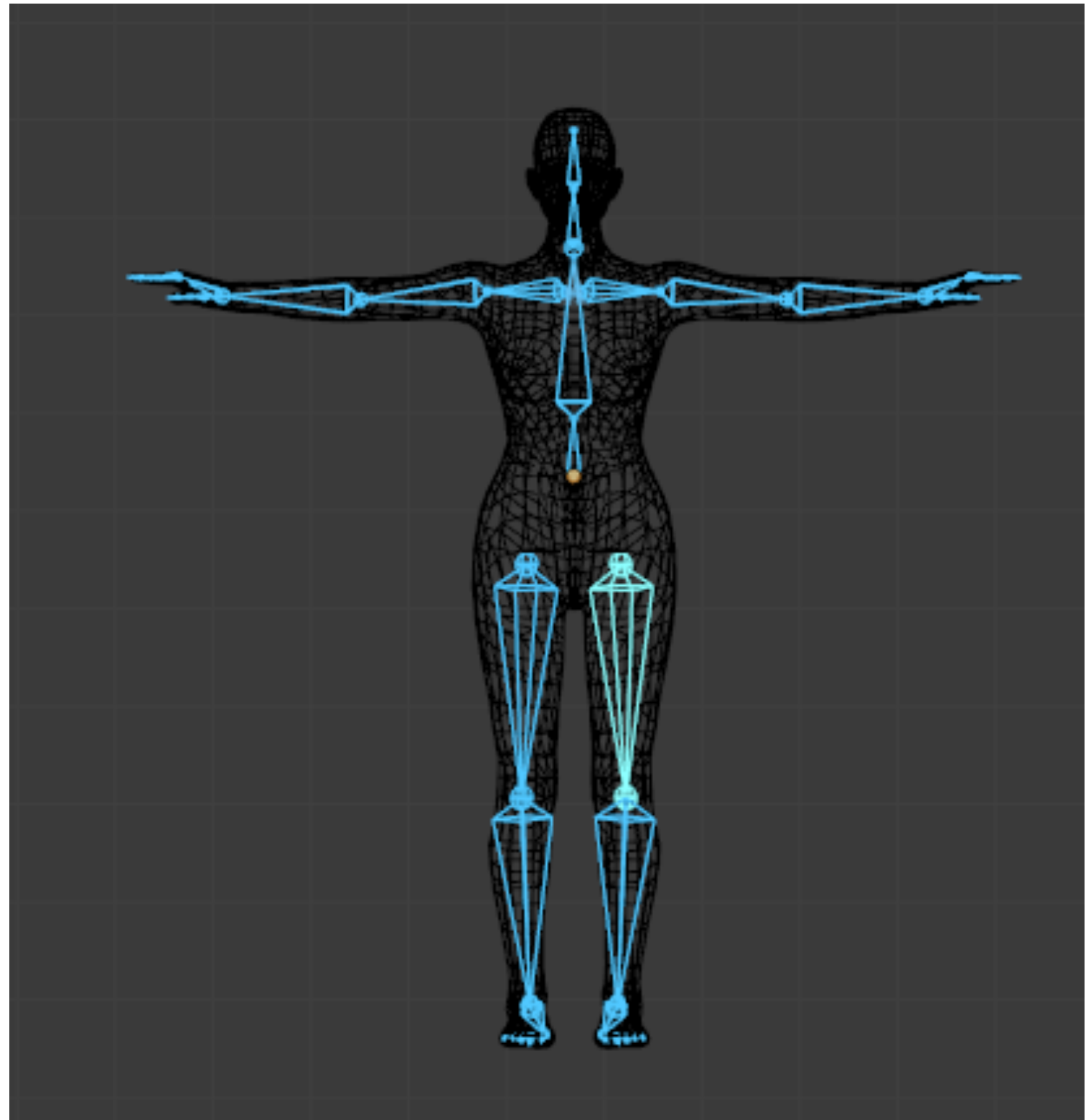
Animate Colors

- Other attributes than position can be animated
 - Example: animate green color channel, $G = 255 * t$



Skinning Characters

- Use a skeleton inside character model (Rigging)
- Connect vertices on the skin to the underlying bones
- Animate bones, skin will follow
- Connect vertex to multiple bones and blend with weights



Extras

- Horizon Zero Dawn animation reel
<https://vimeo.com/212880663>
- Physically-Based Animation
 - Water Simulation
 - Lifelike Fluid Simulations
 - <https://youtu.be/ureGelZPi3o>
- Crowd dynamics
 - <https://youtu.be/daysCqmqd2Y>