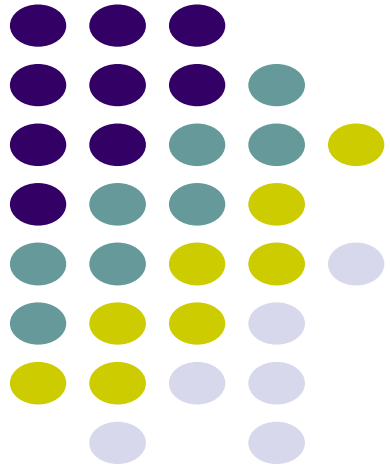
Computer Graphics (CS 4731)
Special Topics

Joshua Cuneo

Computer Science Dept.

Worcester Polytechnic Institute (WPI)



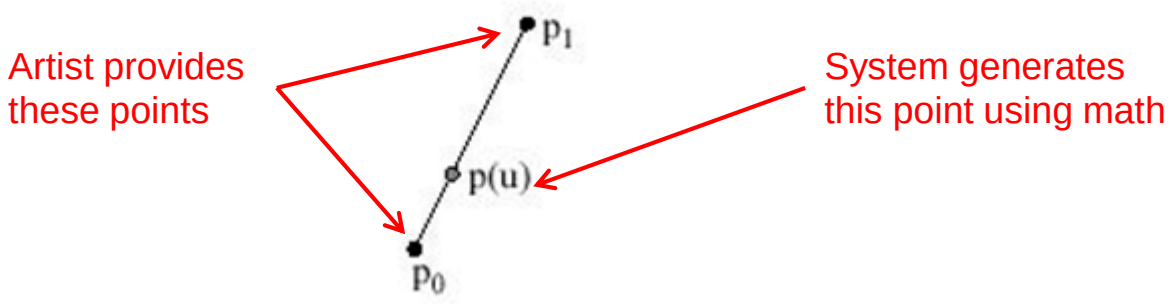
Interactive Curve Design

- 1 approach: curves pass through control points (interpolate)
- Difficulty with this approach:
  - Polynomials always have “wiggles”
  - For straight lines wiggling is a problem
- Our approach: approximate control points (Bezier, B-Splines)

Bezier Curves

- Consider smooth curve that approximates sequence of control points [p0,p1,….]

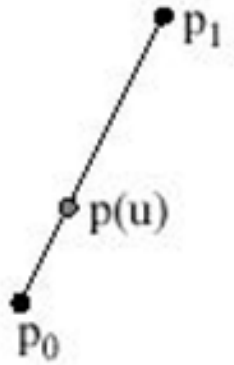
$$p(u) = (1-u)\,p_0 + up_1 \qquad 0 \le u \le 1$$

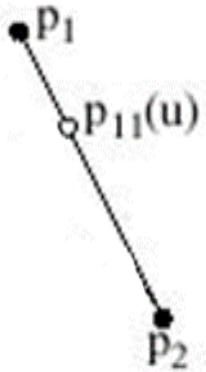


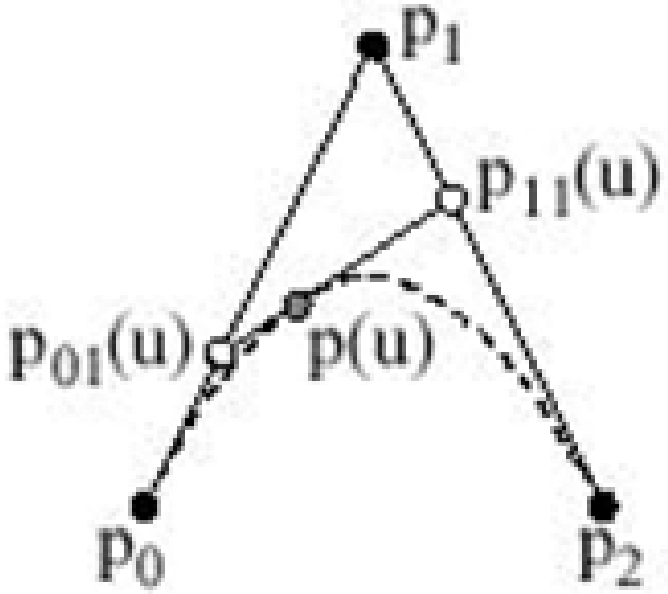
Bezier Curves

- Now consider 3 points
- 2 line segments, P0 to P1 and P1 to P2

$$p_{01}(u) = (1-u)\,p_0 + up_1 \qquad p_{11}(u) = (1-u)\,p_1 + up_2$$







Bezier Curves

Substituting known values of $p_{01}(u)$ and $p_{11}(u)$

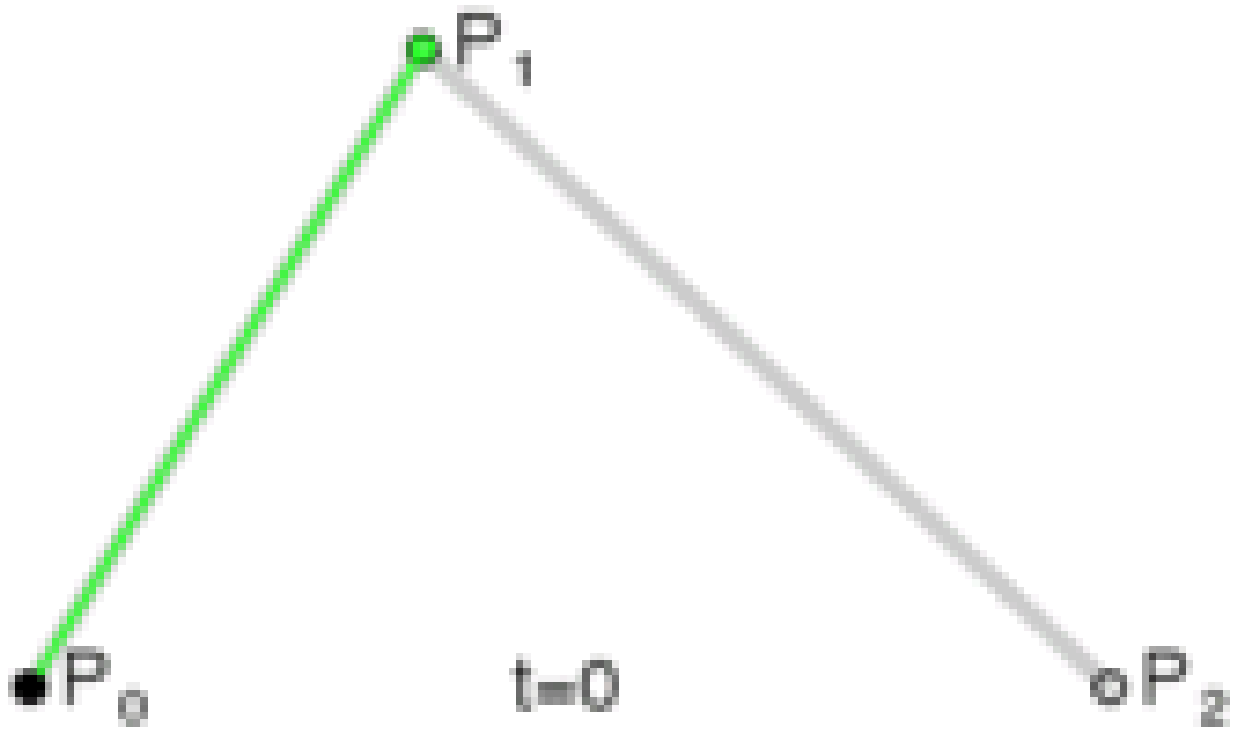
$$p(u) = (1-u)p_{01} + up_{11}(u)$$

$$= (1-u)^2 p_0 + (2u(1-u))p_1 + u^2 p_2$$

$b_{02}(u)$ $b_{12}(u)$ $b_{22}(u)$

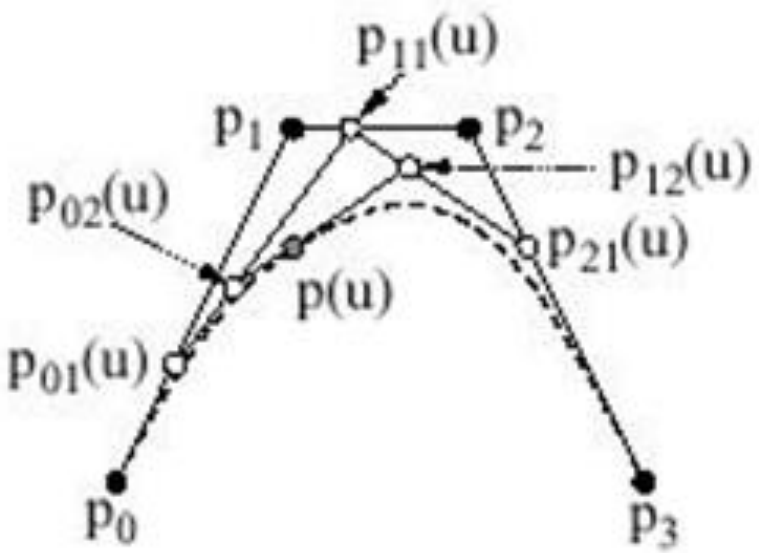
Blending functions for degree 2 Bezier curve

$$b_{02}(u) = (1-u)^2 \qquad b_{12}(u) = 2u(1-u) \qquad b_{22}(u) = u^2$$


P1
P0
t=0
P2

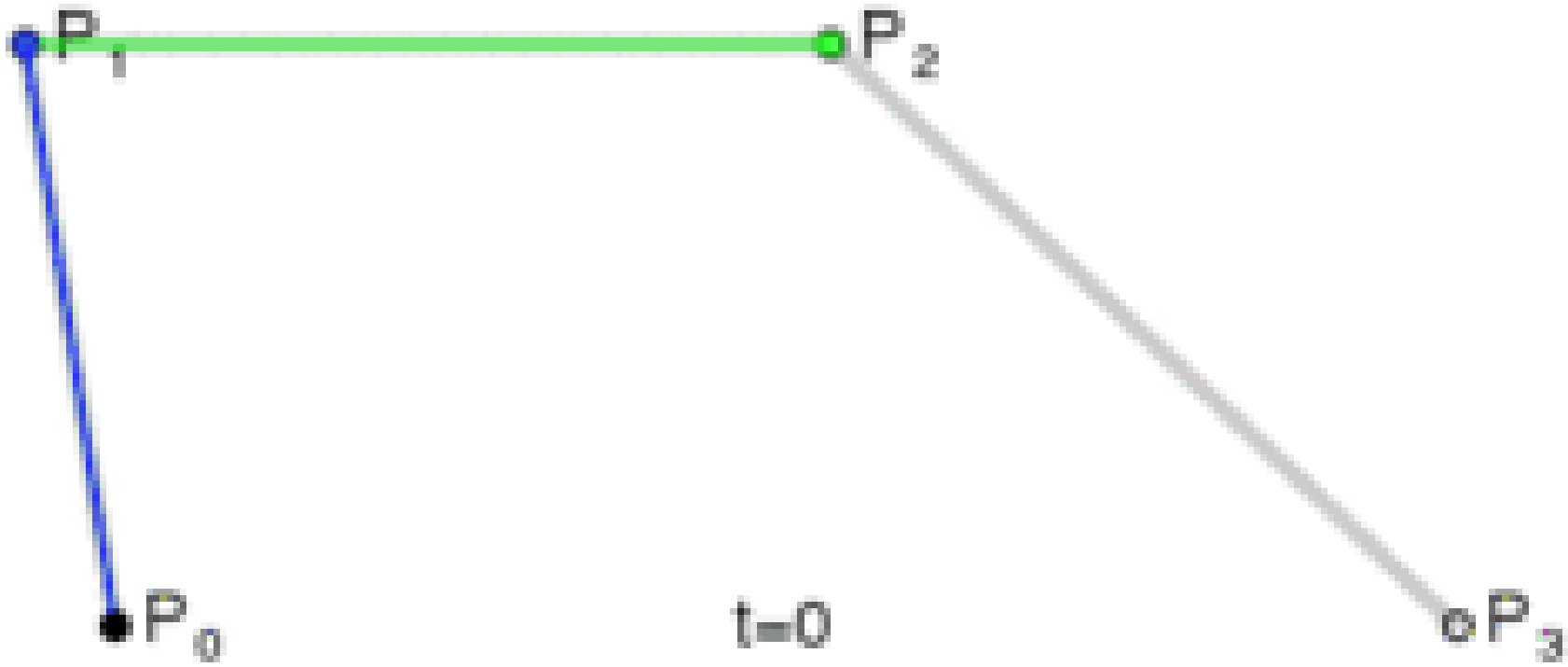
Bezier Curves

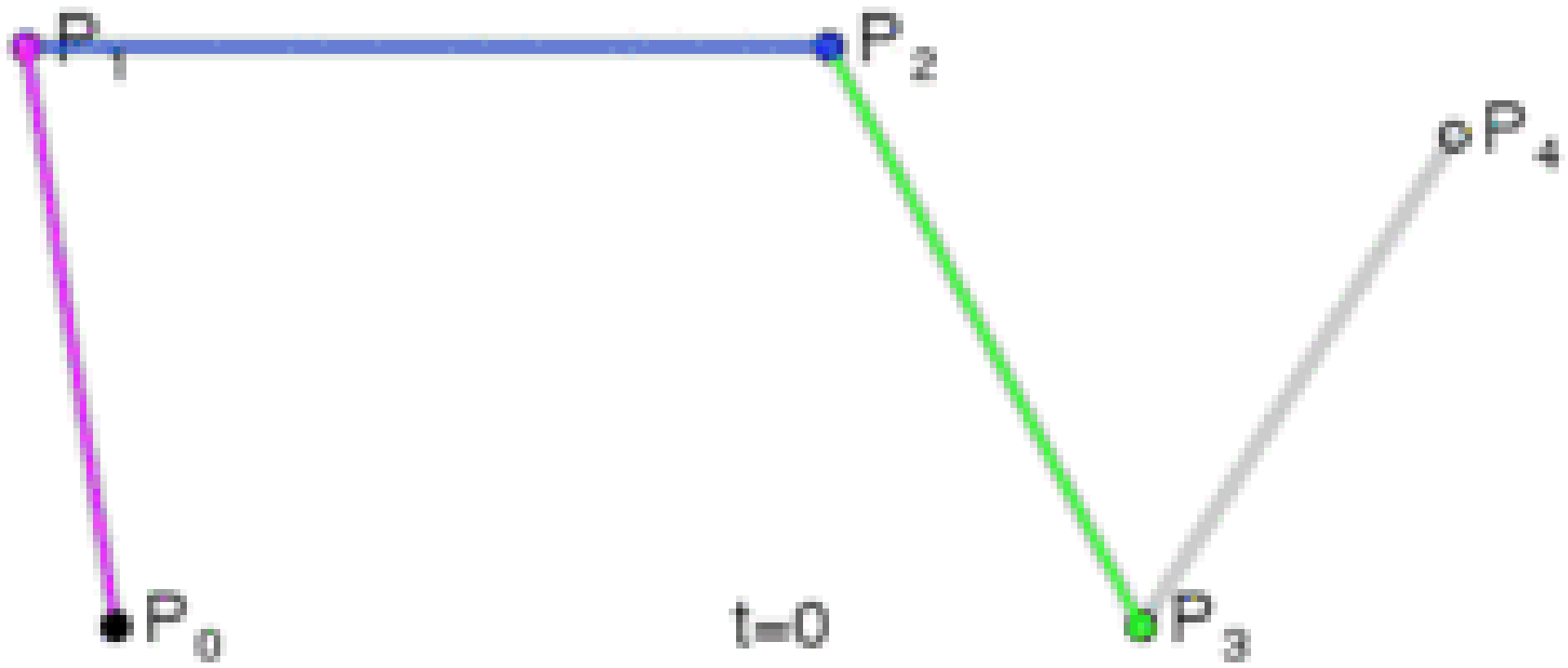
- Extend to 4 control points P0, P1, P2, P3



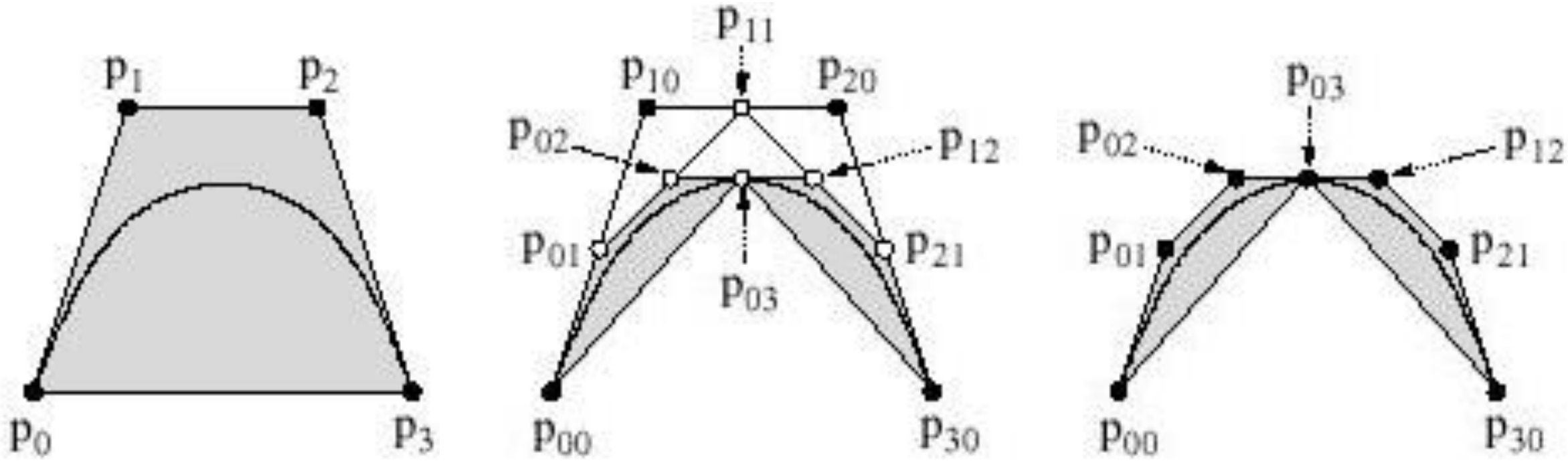
$$p(u) = (1-u)^3 p_0 + (3u(1-u)^2) p_1 + (3u^2(1-u)) p_2 + u^3 p_3$$

$b_{03}(u)$ $b_{13}(u)$ $b_{23}(u)$ $b_{33}(u)$


P1
P2
P0
t=0
P3

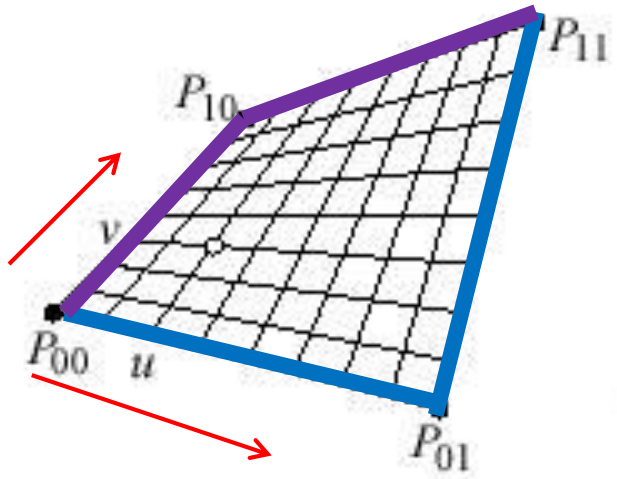

P1
P2
P4
P0
t=0
P3

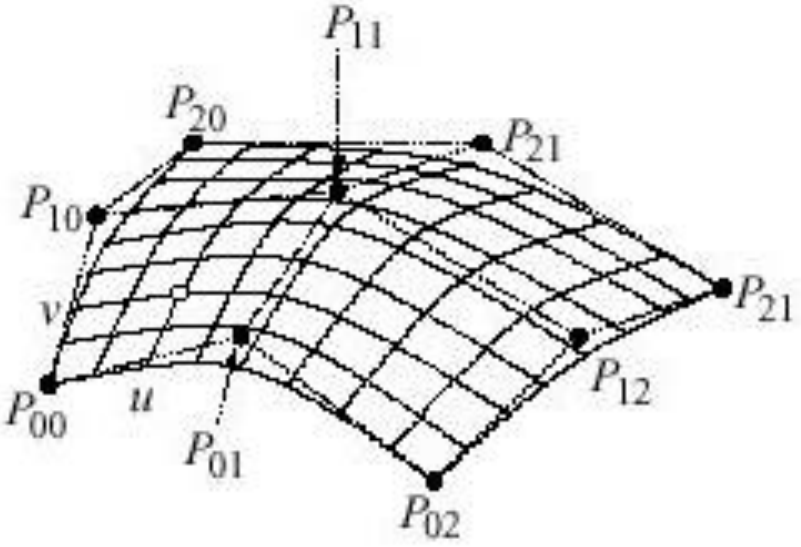
Subdividing Bezier Curves



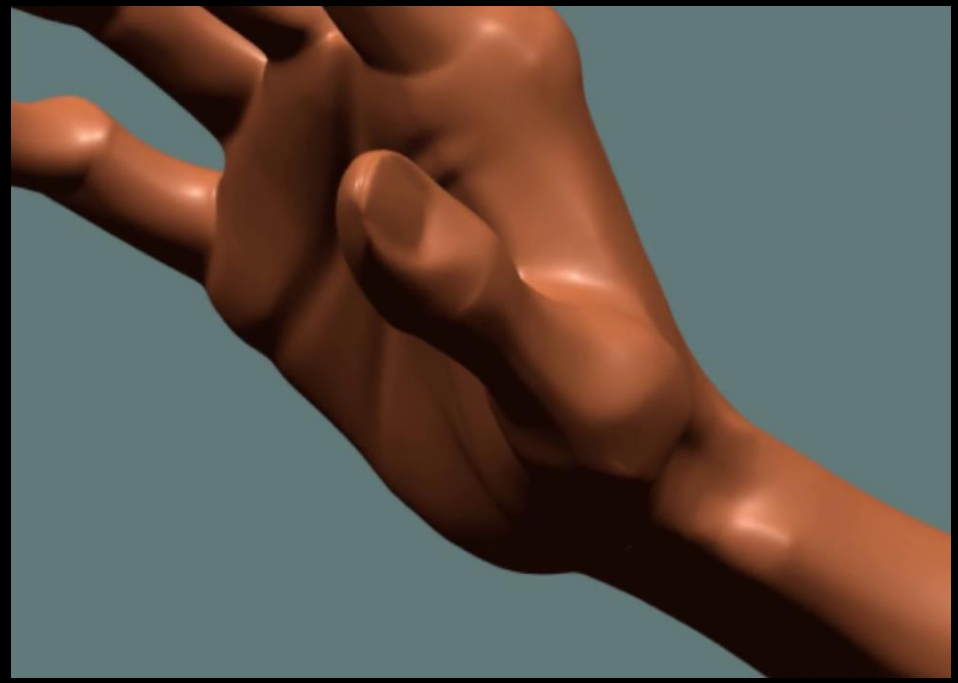
Bezier Surfaces

- Example: 4 control points, P00, P01, P10, P11,
  - 2 parameters **u** and **v**
- Interpolate between
  - P00 and P01 using **u**
  - P10 and P11 using **u**
  - P00 and P10 using **v**
  - P01 and P11 using **v**

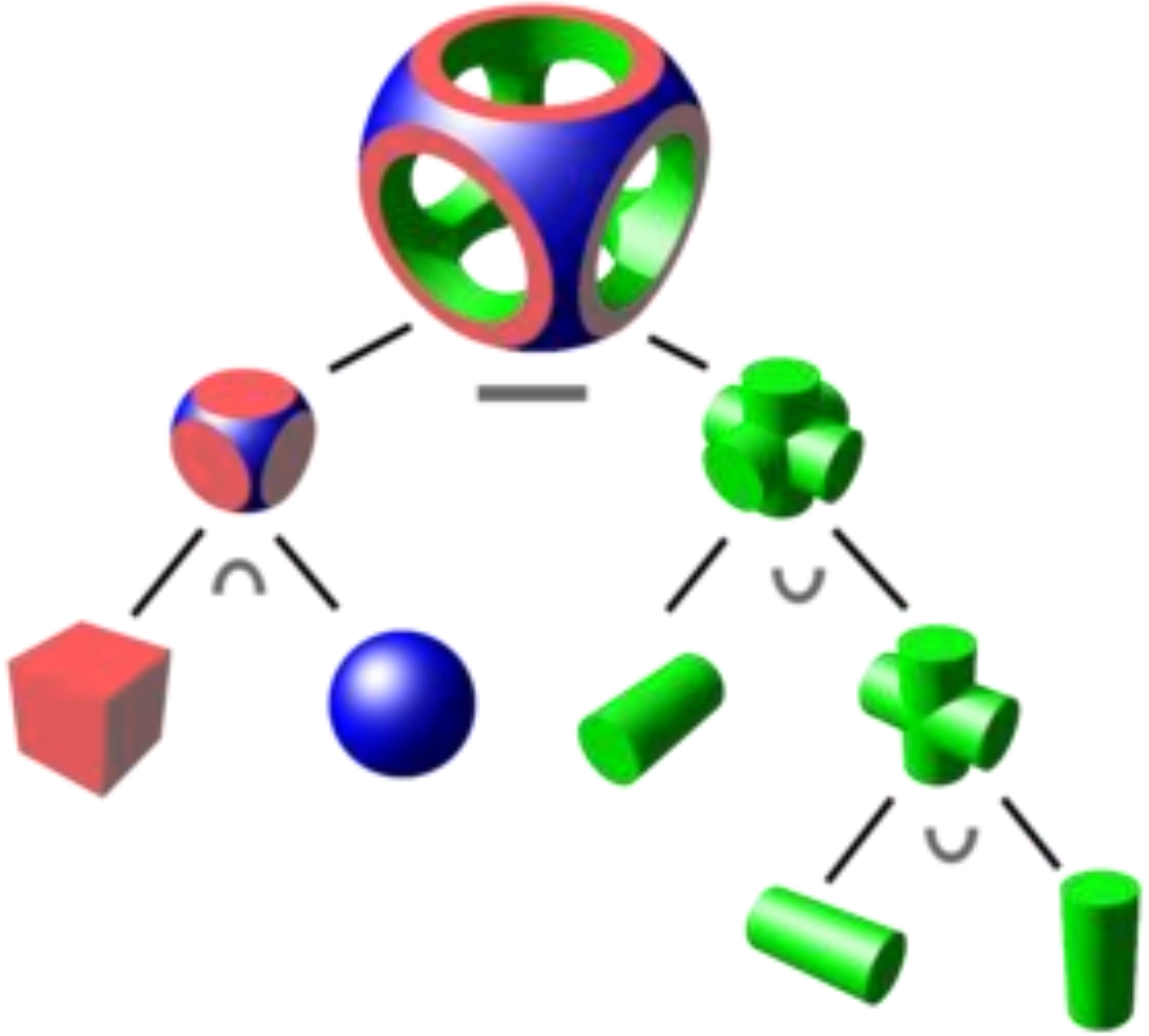




$$p(u,v) = (1-v)((1-u)\,p_{00} + up_{01}) + v((1-u)\,p_{10} + up_{11})$$

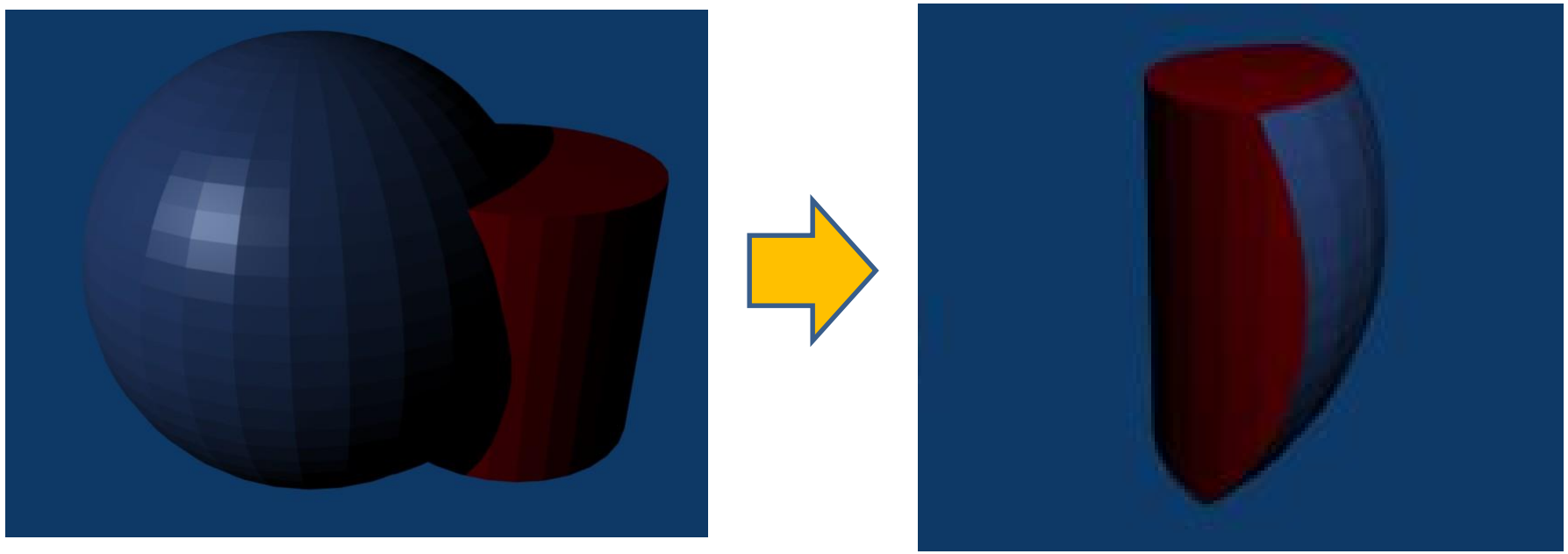


Constructive Solid Geometry



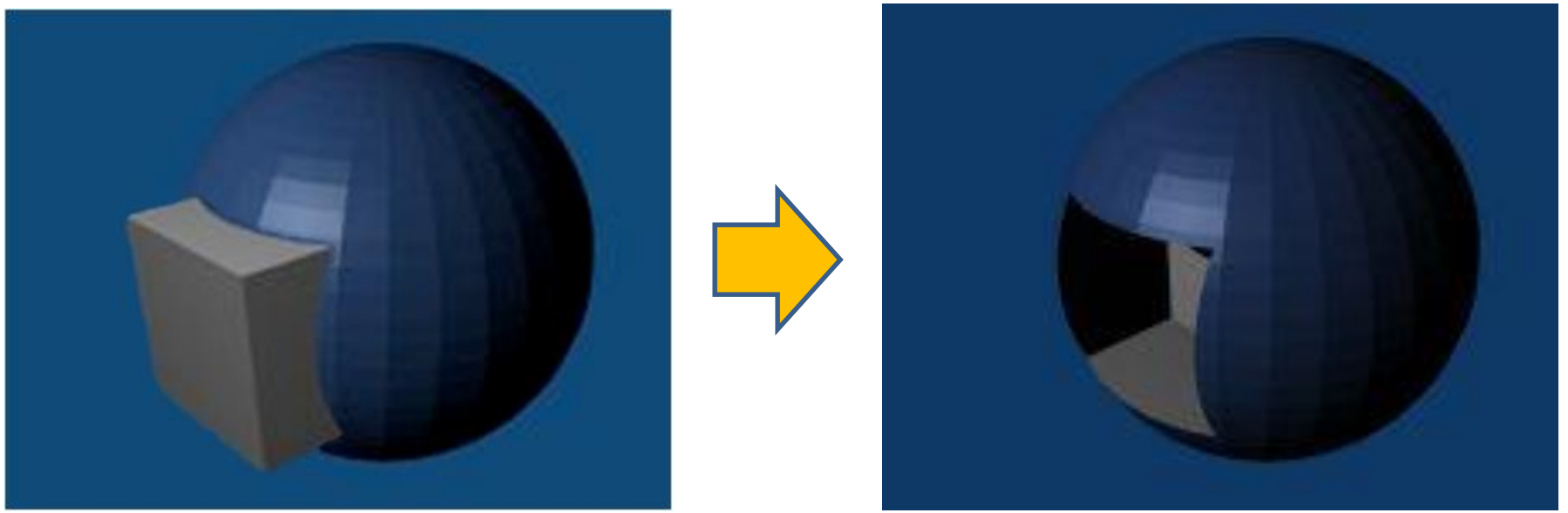
Constructive Solid Geometry

Intersection

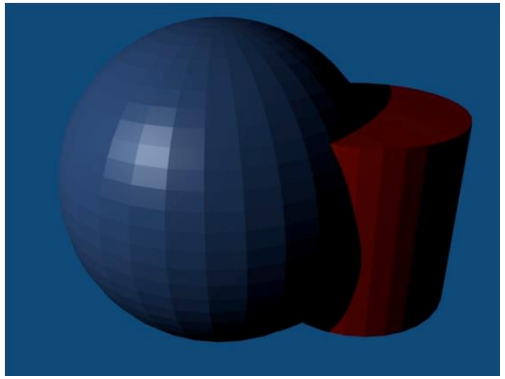


Constructive Solid Geometry

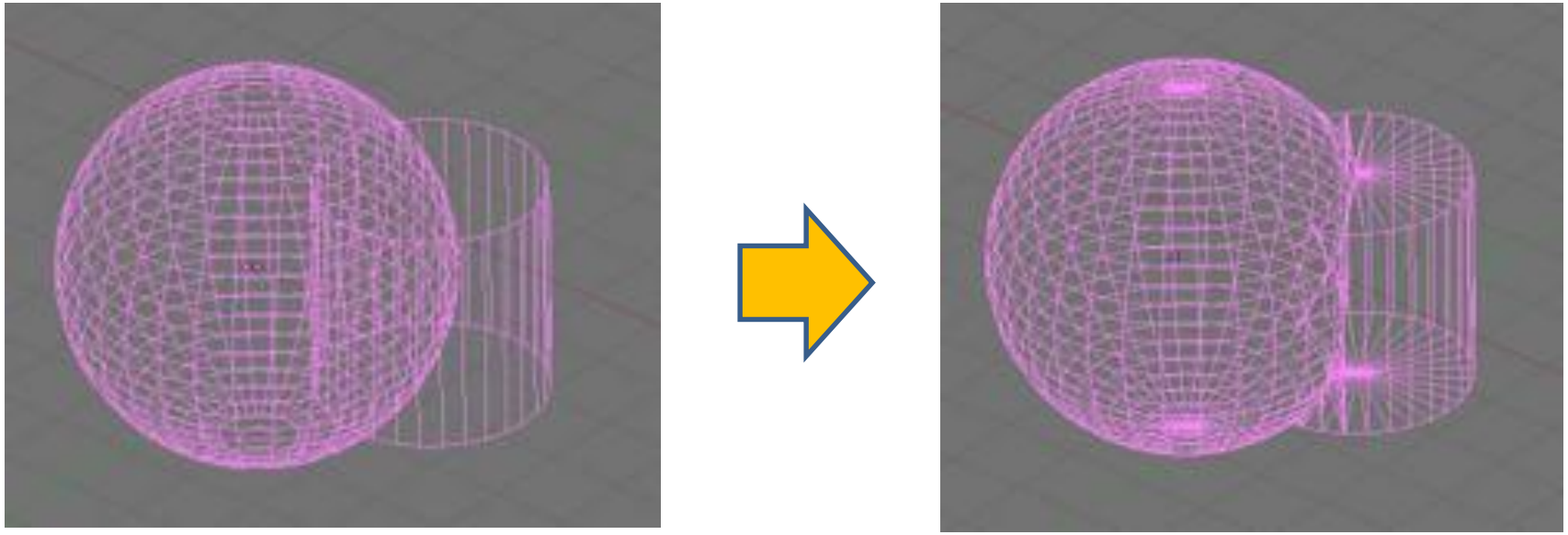
Difference



Constructive Solid Geometry

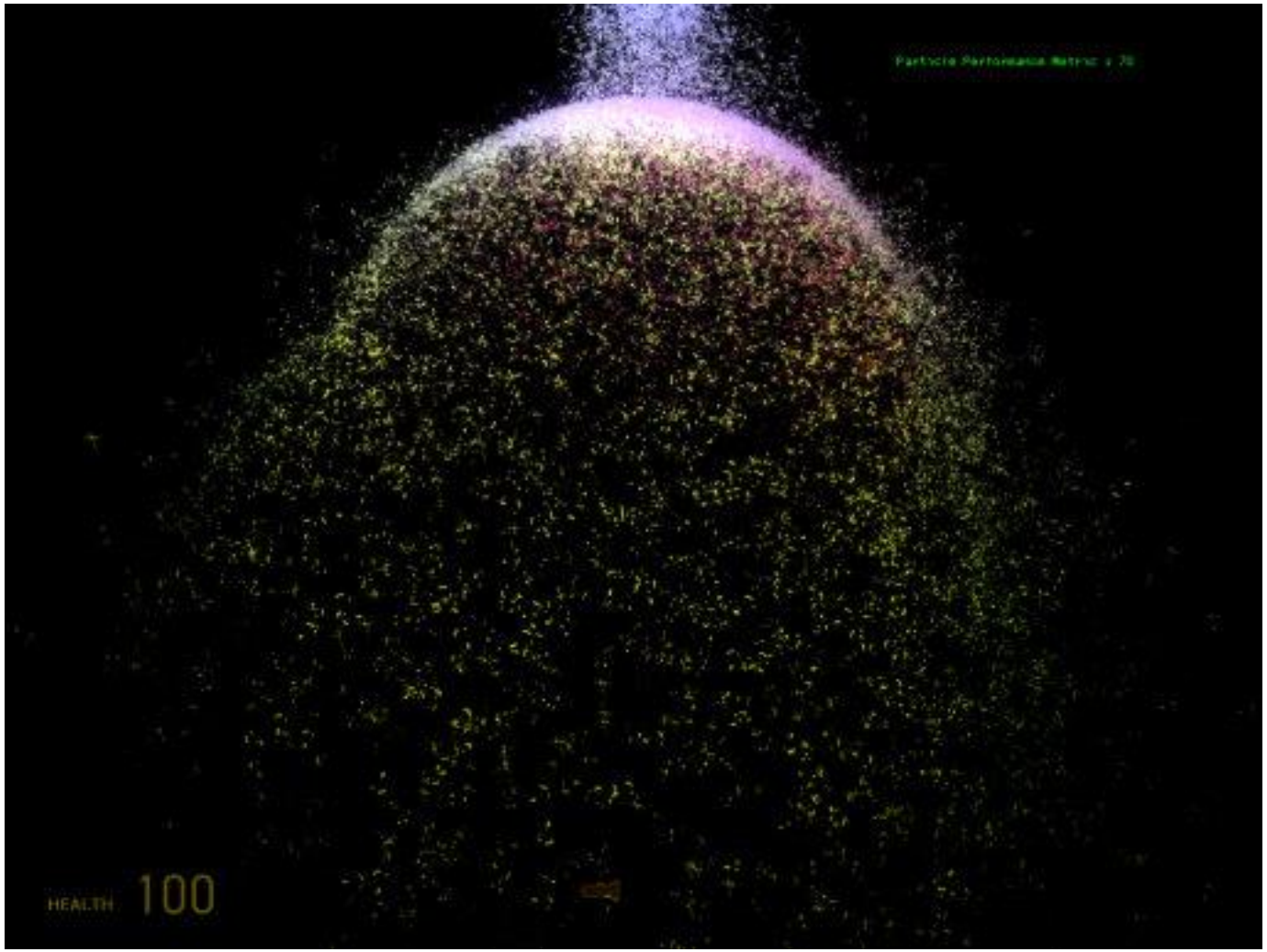


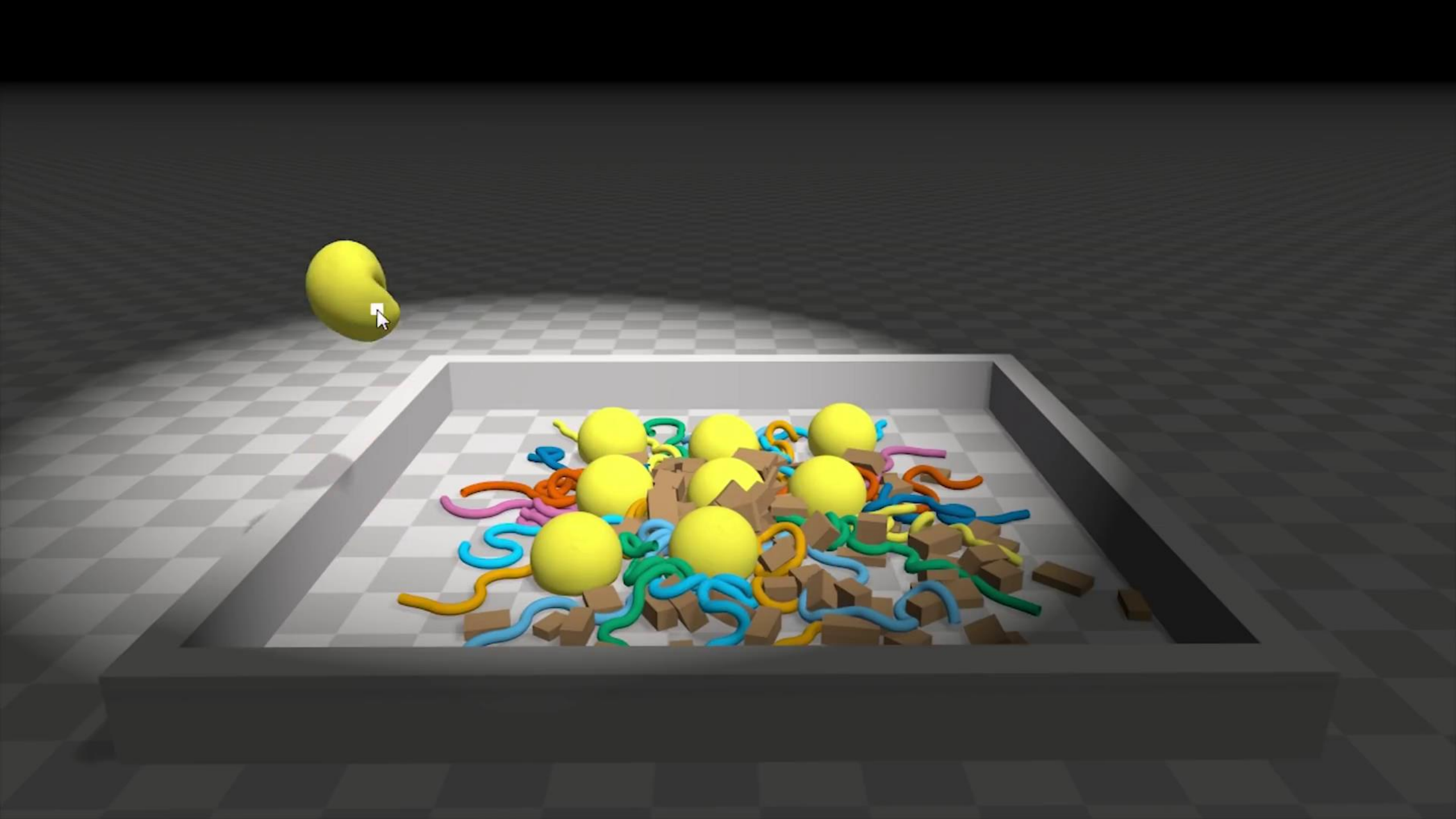
Union




ATOMONTAGE ENGINE

Particle Systems

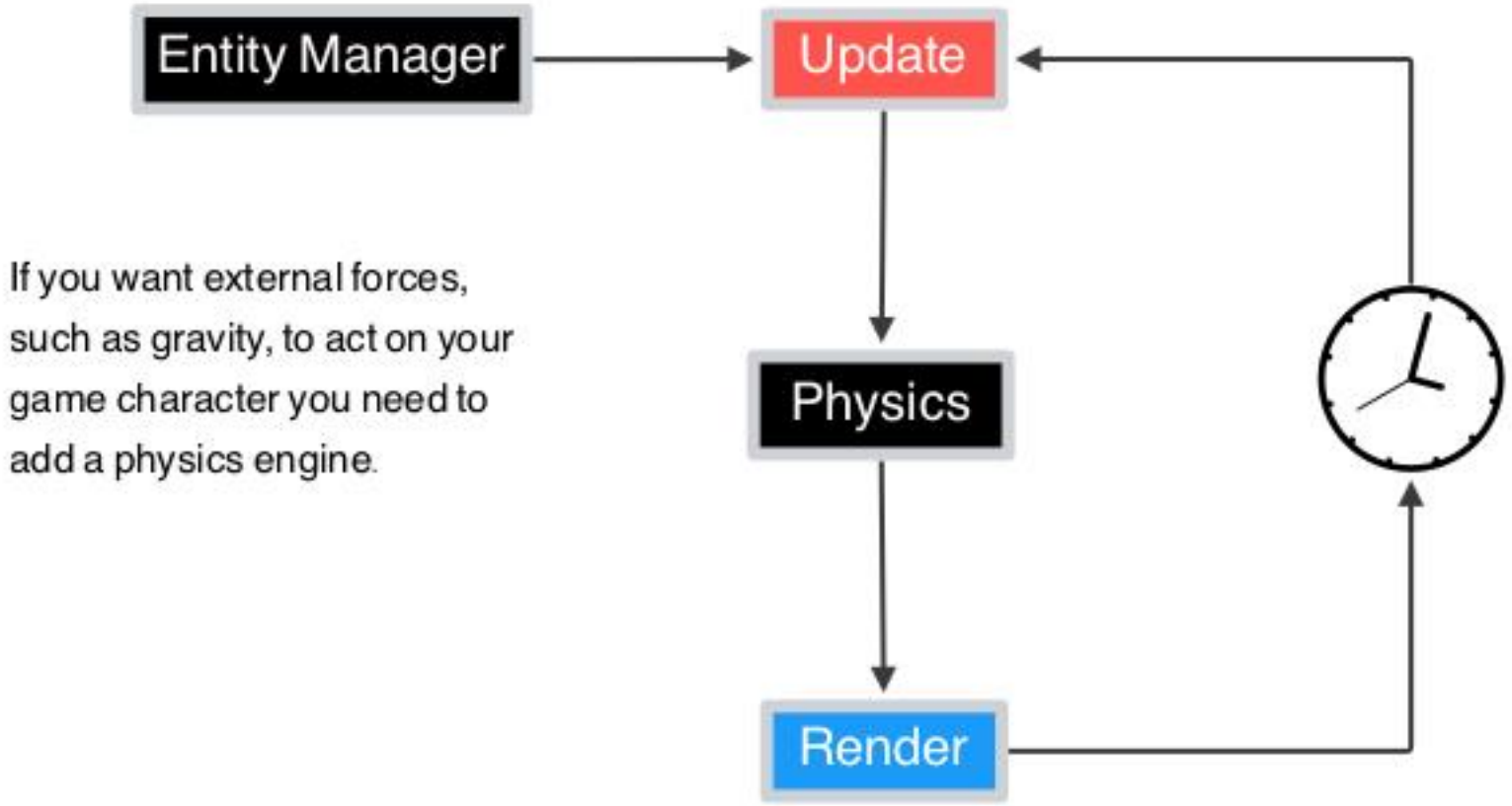




Some Optimization Approaches

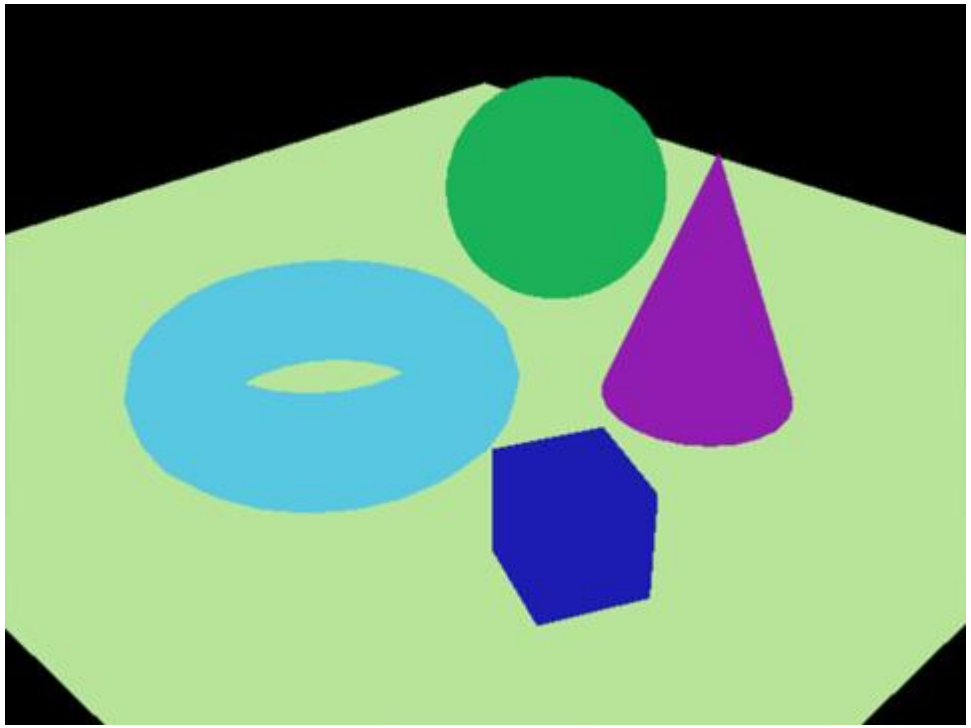
- Pre-process what you can
- Use per-vertex data to encode shading parameters
- Use textures to encode shading parameters
- Write your code tailored to a specific platform to run a specific type of game
- Use triangles, not other types of polygons
- Minimize the number of draw calls wherever possible
- Store data in the CPU cache instead of RAM, where possible
- Concatenate textures into single images

Separate Rendering and Physics

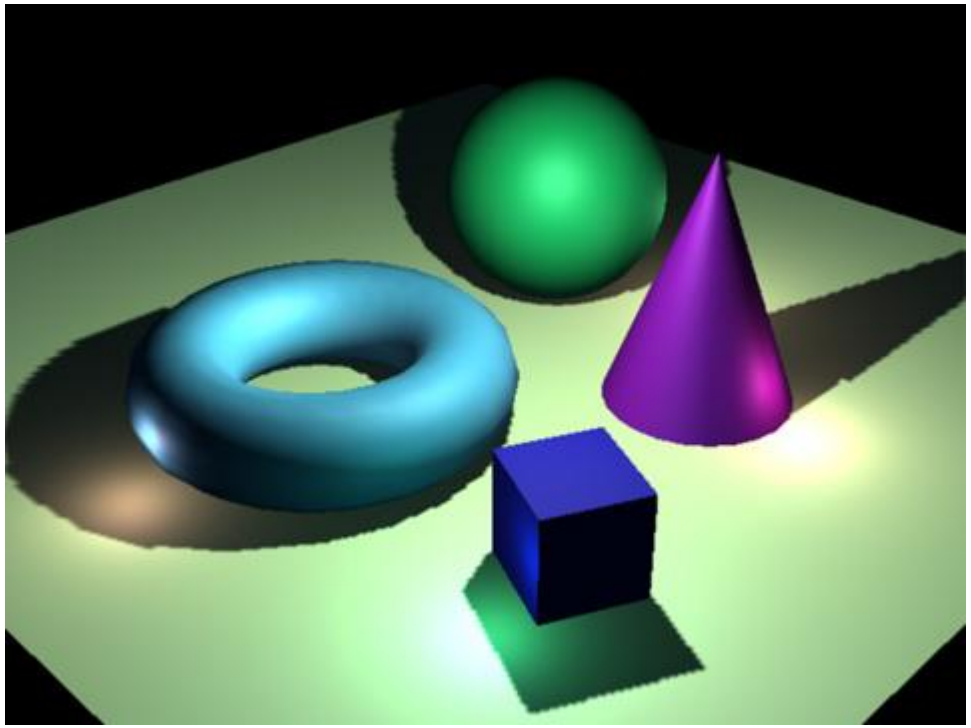


If you want external forces, such as gravity, to act on your game character you need to add a physics engine.

Deferred Rendering

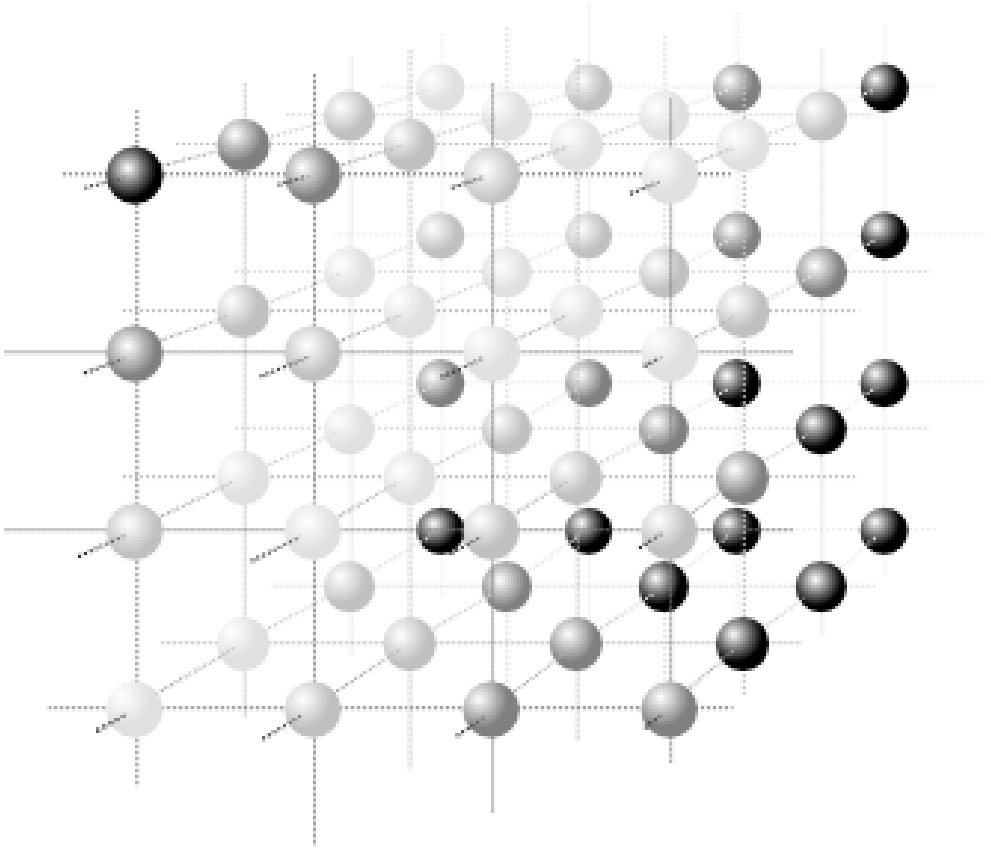


First pass

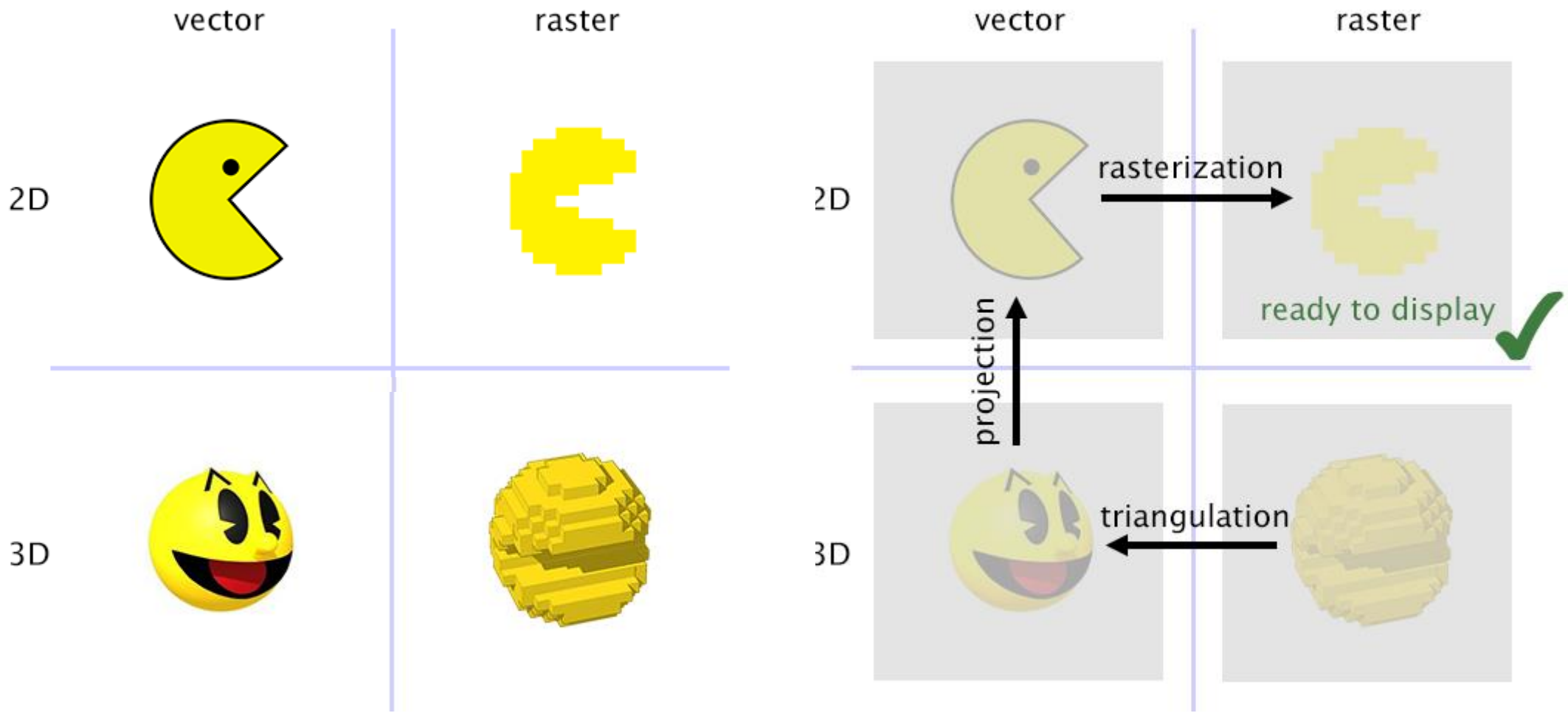


Second pass

Voxels



Voxels


vector
raster
2D
3D
vector
raster
2D
rasterization
ready to display
projection
triangulation
3D

Uncanny Valley